Hybrid Knowledge Architectures for Question Answering

Kaixin Ma

CMU-LTI-23-011

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Thesis Committee:

Eric Nyberg (Chair)
Yonatan Bisk
Maarten Sap
Alessandro Oltramari (Bosch Research)
Hao Cheng (Microsoft Research)

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
In Language and Information Technologies

Abstract

Question answering (QA) is a knowledge-intensive task in natural language processing (NLP) that requires the system to provide answers to user queries expressed in natural language. The types of knowledge a QA system is equipped with broadly fall into two categories, namely *explicit knowledge* and *implicit knowledge*. The explicit knowledge takes formats that are human readable, e.g. raw text, knowledge graphs, structured tables etc, while the implicit knowledge resides in the model parameters that are learned by training on the explicit knowledge. Due to the complementary nature of these two types of knowledge, most recent QA research has tried to leverage both of them for modeling. However, existing work on this front mostly focuses on building customized models for specific datasets, which do not generalize well to other use cases. Moreover, while using end-to-end models to directly learn to fuse different knowledge is a simple solution and often works well, it's hard to interpret the model's reasoning process, leading to untrustworthy predictions. Finally, most systems are designed without considering memory and computation efficiency, which hinders' their application to real-world use cases.

With the aforementioned issues in mind, in this thesis, we present solutions for building generalizable, interpretable, and efficient QA systems. Specifically, we present three solution elements, namely 1) hybrid knowledge fusion, 2) modularized knowledge framework, and 3) modularized knowledge sharing. In the first part, we study various ways of injecting commonsense knowledge into QA systems powered by pretrained language models. Our results show that instance-level late fusion of knowledge subgraphs is promising in a supervised setting and pretraining on transformed knowledge graphs (KGs) provides substantial gains across a diverse set of tasks in a zero-shot setup. These findings show that combining explicit and implicit knowledge is a step towards generalization across different domains of questions. In the second part, we explored two different modularized frameworks for open-domain question answering that bridge the gap across knowledge types and question types. We show that text can serve as a universal knowledge interface for different types of structured knowledge, and decomposing the reasoning process into discrete steps enables a single unified system to solve both single-hop and multi-hop questions. Modularized frameworks not only offer generalization across modalities of knowledge and question types but also bring improved interpretability of the reasoning process. In the third part, we extend the modularized framework from the previous part by allowing implicit knowledge sharing among different modules. Multiple reasoning modules are merged together and learned simultaneously through multi-task learning, and we further add skill-specific specialization for each module to reduce task interference. Such an architecture not only greatly reduced the overall model size but also improved the inference efficiency, therefore achieving all three target properties generalizability, interpretability, and efficiency. Finally, we discuss open challenges and ways forward beyond this thesis.

Acknowledgements

I would like to start by expressing my deepest gratitude to my advisor, Prof. Eric Nyberg, for his unwavering support and guidance in my academic journey and personal growth. Prof. Nyberg's encouragement and care have acted as a reliable beacon, guiding me toward achieving my goals. His understanding and support have provided the necessary resilience to navigate through challenging periods. His faith in my potential has served as a powerful catalyst, propelling me toward greater achievements. I am truly grateful for having him as my advisor.

I am also immensely grateful to the esteemed members of my thesis committee, Prof. Yonatan Bisk, Prof. Maarten Sap, Dr. Alessandro Oltramari and Dr. Hao Cheng, whose invaluable guidance and support have been instrumental in my academic journey. I'm very grateful for Yonatan and Maarten's insightful feedback and comprehensive perspectives, which have not only significantly improved the quality of my work but also spurred me to delve into deeper and more expansive intellectual territories. My heartfelt thanks to Alessandro, whose relentless support has been a steady presence throughout my Ph.D. journey. Demonstrating an exceptional commitment, Alessandro has often traversed the extra mile to assist in the achievement of my goals. His cheerful personality and genuine kindness have consistently infused our interactions with positivity. I'm truly fortunate to have him as my mentor. I'm very grateful to Hao for his exceptional guidance over the past two years. His expertise in research and meticulous attention to detail have been instrumental in shaping my development as a researcher, teaching me the essence of academic rigor and precision.

I am also deeply grateful to all my research collaborators, whose contributions have been invaluable. My sincere gratitude is extended to Dr. Jonathan Francis, who has consistently served as an exceptional mentor. Known for his research versatility and warmheartedness, Jon's insights and assistance were instrumental in fostering my growth during the early stages of my academic pursuit. I am greatly thankful to Prof. Filip Ilievski, whose extraordinary support over the past three years has been pivotal. During a time of uncertainty, Filip extended a hand, leading to vibrant discussions that ignited a number of exciting ideas and ultimately resulted in fruitful research outcomes. His kind personality, combined with meticulous research mentorship, has consistently served as an inspiring model for me. My appreciation extends to Prof. Louis-Philippe Morency, Prof. Jeffrey M Girard, Prof. Florian Metze, Dr. Xiaodong Liu and Dr. Jianfeng Gao, whose insightful feedback and acute research vision significantly elevated the quality of my work. I would like to thank James Route, Xinyu Wang, Satoru Ozaki, Billy Li, Jiarui Zhang and Yifan Jiang. The opportunity to work alongside and learn from these brilliant individuals has been an absolute pleasure and an enriching experience.

I extend heartfelt gratitude to my friends at LTI, who have been integral to the insightful dis-

cussions and the delightful moments we've cherished together. My deep appreciation to my close friend Ruohong Zhang, who is always reliable and an embodiment of inspiring work ethics that have continually motivated me. Unending thanks to my close friend Prakhar Gupta, with whom I can share life experiences, cultivate research ideas and seek advice. The trips and conversations we had together will forever remain in my memory. I am deeply grateful to my close friend Aman Madaan, whose invaluable advice on research and exemplary qualities I constantly aspire to emulate. I would also like to express my gratitude to Hwijeen Ahn, Srijan Bansal, Sang Keun Choe, Zhen Fan, Shengyu Feng, Liangke Gui, Xiaochuang Han, Zhengbao Jiang, Billy Li, Xiaopeng Lu, Sanket Vaibhav Mehta, Satoru Ozaki, Yansen Wang, Zhiruo Wang, Jianing Yang, Donghan Yu, Zhisong Zhang, Shuyan Zhou, Xuhui Zhou and Hao Zhu. Their companionship has infused my journey at LTI with joy and fulfillment.

Lastly, but most importantly, my deepest appreciation to my family. To my parents, Guoqing Ma and Xiaoling Wang, thank you for your unconditional love and for instilling in me the value of hard work and perseverance, and for your unconditional support in all of the decisions I made. To my partner, Zhifei Wu, your encouragement and belief in me have been my source of strength. You have been illuminating my life's journey throughout these years. To my best friends, Shijing Li, Zhiyu Pan, Lichao Tang and Dr. Zitang Wei, the joyful time we spent together has imbued an irreplaceable richness to my life.

This journey would not have been possible without all of you. Thank you.

Contents

1 Introduction **15**
- 1.1 Background and Motivation . . . 15
- 1.2 Concepts and Terminology . . . 17
- 1.3 Thesis Overview . . . 20

2 Towards Generalizable Neuro-Symbolic Systems for Commonsense Question Answering **23**
- 2.1 Introduction . . . 23
- 2.2 Related Work . . . 25
- 2.3 Approach Overview . . . 26
  - 2.3.1 Model architecture . . . 26
  - 2.3.2 Knowledge bases . . . 28
  - 2.3.3 Knowledge elicitation . . . 28
  - 2.3.4 Knowledge injection . . . 30
  - 2.3.5 Knowledge pre-training . . . 31
- 2.4 Experiments . . . 31
  - 2.4.1 Datasets . . . 31
  - 2.4.2 Training details . . . 32
  - 2.4.3 Results . . . 32
- 2.5 Error Analysis . . . 33
- 2.6 Discussion and Conclusion . . . 35

3 Knowledge-driven Data Construction for Zero-shot Evaluation in Commonsense Question Answering **37**
- 3.1 Introduction . . . 37
- 3.2 Related Work . . . 39
  - 3.2.1 Knowledge Injection . . . 39
  - 3.2.2 Generating Commonsense Questions and Answers . . . 39
- 3.3 Zero-Shot QA Framework . . . 40
  - 3.3.1 Synthetic QA Generation . . . 40
  - 3.3.2 Distractor Sampling . . . 42
  - 3.3.3 Language Models . . . 43
  - 3.3.4 Tasks . . . 44

3.4 Experimental Setup . . . 45
3.4.1 Baselines . . . 45
3.4.2 Implementation . . . 45
3.4.3 Hypotheses . . . 45
3.5 Results . . . 46
3.5.1 Main Results . . . 47
3.5.2 Comparison of QA Generation Strategies . . . 47
3.5.3 Comparison of Training Regimes . . . 47
3.5.4 Difficulty of the Synthetic QA Sets . . . 48
3.6 Discussion . . . 48
3.6.1 Towards a Commonsense Service . . . 48
3.6.2 Impact of Knowledge . . . 49
3.6.3 Generating Fair and Informative Questions . . . 49
3.7 Conclusions and limitations . . . 50

4 Open Domain Question Answering with A Unified Knowledge Interface 51
4.1 Introduction . . . 51
4.2 Overview of UDT-QA . . . 53
4.3 Verbalizer: Data-to-text Generation . . . 54
4.3.1 Input Format . . . 54
4.3.2 Improved Data-to-Text Model Training . . . 55
4.4 Experiment Setup . . . 55
4.4.1 Datasets . . . 56
4.4.2 Structured Knowledge Sources . . . 56
4.5 Experiments: Data-to-Text . . . 57
4.6 Experiments: QA over Data and Text . . . 58
4.7 Analysis . . . 59
4.8 Related Work . . . 62
4.9 Discussion and Conclusion . . . 63

5 Open Domain Question Answering via Chain-of-Reasoning over heterogeneous knowledge 65
5.1 Introduction . . . 65
5.2 Overview of the CORE Framework . . . 67
5.3 Intermediary Modules . . . 68
5.3.1 Linker . . . 68
5.3.2 Chainer . . . 70
5.4 Experiments . . . 71
5.4.1 Datasets . . . 71
5.4.2 Experimental Settings . . . 71
5.4.3 Baselines . . . 72
5.4.4 Results . . . 73
5.5 Analysis . . . 74

5.5.1 Ablation Study . . . 74
5.5.2 Impact of Linker & Chainer . . . 75
5.5.3 Alternative Linking Strategy . . . 75
5.5.4 Linker Performance . . . 76
5.5.5 Case Study . . . 77
5.6 Related Work . . . 78
5.7 Discussion and Conclusion . . . 78

6 Chain-of-Skills: A Configurable Model for Open-Domain Question Answering 81
6.1 Introduction . . . 81
6.2 Background . . . 83
6.3 Approach . . . 84
6.3.1 Reasoning Skill Modules . . . 84
6.3.2 Modular Skill Specialization . . . 86
6.3.3 Inference . . . 87
6.4 Experiments . . . 88
6.4.1 Datasets . . . 88
6.4.2 Evaluation Settings . . . 88
6.4.3 Results . . . 89
6.5 Analysis . . . 91
6.5.1 Task Interference . . . 91
6.5.2 Benefit of `Chain-of-Skills` Inference . . . 93
6.5.3 Reduced Model Size and Computation . . . 93
6.5.4 Effect of Pretraining . . . 93
6.5.5 Swapping Experts . . . 94
6.6 Question Answering Experiments . . . 94
6.7 Related Work . . . 95
6.8 Discussion and Conclusions . . . 96

7 Conclusions and Future Work 97
7.1 Summary of Contributions . . . 97
7.2 Beyond This Thesis . . . 98
7.3 Open Challenges . . . 100

A Appendix for Chapter 3 131
A.1 Hyperparameters . . . 131
A.1.1 Model training . . . 131
A.1.2 Tuned parameters . . . 131
A.2 AFLite algorithm . . . 131

B Appendix for Chapter 4 133
B.1 Document Index Statistics . . . 133
B.2 Training Details . . . 133

B.3 Impact of Document Index Size . . . 134
B.4 Comparison betweeh Our Verbalizer and KELM-verbalizer . . . 135
B.5 Case Study on Raw vs Verbalized Tables . . . 135
B.6 Data-to-text Examples . . . 135

C Appendix for Chapter 5 **139**
C.1 Chaining Strategy . . . 139
C.2 Dataset Statistics . . . 139
C.3 Training Details . . . 139
C.3.1 Linker Training . . . 139
C.3.2 Retriever Training . . . 140
C.3.3 Chainer Inference . . . 140
C.3.4 Reader Training . . . 140
C.4 Results and Discussion . . . 141
C.4.1 Impact of Linker & Chainer . . . 141

D Appendix for Chapter 6 **143**
D.1 Inference Pipeline . . . 143
D.2 Experimental Details . . . 144
D.2.1 Data Statistics . . . 144
D.2.2 Training Details . . . 145
D.2.3 Inference Details . . . 147
D.3 Question Answering Results . . . 148
D.3.1 Training Details . . . 148
D.3.2 Results . . . 149
D.4 Computation . . . 150
D.5 Licenses . . . 150

List of Figures

1.1 The Overview of this thesis 19
1.2 Overview of the Aspects of the QA system that this thesis focus on 20

2.1 HyKAS model 26

3.1 An illustration of our question generation pipeline. 41

4.1 An overview of UDT-QA based on the *verbalizer-retriever-reader* pipeline. 53

5.1 Our CORE vs. previous retriever-only methods for evidence discovery. The grey square is the question, the blue dots are 1st-hop passages, the blue triangles are 1st-hop tables, and the purple dots are 2nd-hop passages. 66
5.2 An illustration of CORE for ODQA. Given a question, the *retriever* first finds hop-1 evidence from the entire Wikipedia (orange arrows). Then the *linker* gathers relevant documents for the hop-1 evidence (purple arrows), which are treated as hop-2 evidence. Next, the *chainer* reranks all hop-1 and hop-2 evidence and splices them together into evidence chains (blue arrows). Finally, the *reader* takes in the top-50 chains and produces the answer (black arrows). The gold evidence chain is marked in red. 66

6.1 Comparison of dense retrievers in terms of considered query type and supported skill configuration [a](Karpukhin et al., 2020) [b](Xiong et al., 2020b) [c](Wu et al., 2020). Each box represents a skill (■=*single retrieval*, ■=*expanded retrieval*, ■=*linking*, ■=*reranking*,) and the arrows represent the order of execution. In our case, we can flexibly combine and chain the skills at inference time for different tasks to achieve optimal performance. 82
6.2 Chain-of-Skills (COS) model architecture with three different query types. The left blue box indicates the single retrieval query input. The middle green box is the expanded query retrieval input based on the single retrieval results. The right orange case is the entity-centric query with "deep learning" as the targeted entity. . 83
6.3 Expert configuration for COS at pretraining and fine-tuning. Each numbered box is a skill-specific expert. The lines denote input routing where solid ones also indicate weight initialization mappings. Green lines highlight the expanded query routing which is different for pretraining and fine-tuning. 87

6.4 Top-100 retrieval accuracy on inference strategy: Chain-of-Skills vs retrieval-only. 92
6.5 Comparison on the effect of pretraining using top-100 retrieval accuracy with COS inference. 92

D.1 The reasoning pipeline of Chain-of-Skills (COS). Given a question, COS first identifies salient spans in the question, then the retrieving and linking skills are both used to find first-hop evidence, using the [CLS] token and entity mention representation respectively. Then we merge all the evidence through score alignment and the reranking skill. For top-ranked evidence documents, we concatenate each of them with the question and perform another round of retrieving and linking. Then the second hop evidence are merged and reranked in the same fashion. Finally, the reasoning paths are sorted based on both hops' scores 144

List of Tables

2.1 An example from the DREAM dataset; the asterisk (*) denotes the correct answer. . 24
2.2 An example from the CommonsenseQA dataset; the asterisk (*) denotes the correct answer. . . . 25
2.3 Extracted ConceptNet relations for sample shown in Table 2.2. . . . 29
2.4 Sample generated ATOMIC relations for sample shown in Table 2.1. . . . 29
2.5 Results on DREAM; the asterisk (*) denotes results taken from leaderboard. . . . 31
2.6 Results on CommonsenseQA; the asterisk (*) denotes results taken from leaderboard. . . . 32
2.7 Accuracies for each DREAM question type: **M** means *Matching*, **S** means *Summary*, **L** means *Logic inference*, **C** means *Commonsense inference*, and **A** means *Arithmatic inference*. Numbers beside types denote the number of questions of that type. . . . 33
2.8 Accuracies for each CommonsenseQA question type: **AtLoc.** means *AtLocation*, **Cau.** means Causes, **Cap.** means *CapableOf*, **Ant.** means *Antonym*, **H.Pre.** means *HasPrerequiste*, **H.Sub** means *HasSubevent*, **C.Des.** means *CausesDesire*, and **Des.** means *Desires*. Numbers beside types denote the number of questions of that type. . . . 33

3.1 Generated questions from ATOMIC (top) and CWWV (bottom). (*) denotes the correct answer. . . . 41
3.2 Zero-shot evaluation results with different combinations of models and knowledge sources, across five commonsense tasks. CSKG represent the combination of ATOMIC and CWWV. We run our experiments three times with different seeds and report average accuracy with 95% confidence interval. SMLM (*) used OMCS for CSQA, ROCStories (Mostafazadeh et al., 2016) for aNLI and ATOMIC for SIQA as knowledge resources. . . . 44
3.3 Comparison of different QA generation strategies. . . . 46
3.4 Comparison between MLM and MR training. . . . 48
3.5 LM and human accuracy on our synthetic QA sets. . . . 49

4.1 Intrinsic and extrinsic evaluations of verbalization approaches on DART test and NQ-table-Q (§4.4.1), respectively. "Ans Cov" refers to Answer coverage. All metrics are higher the better except for TER. . . . 55

4.2 End-to-end open-domain QA evaluation of UDT-QA in comparison to recent state-of-the-art models on the test sets of NQ and WebQ. Exact match scores are reported (highest scores shown in **bold**). . . . 58
4.3 Impact of document index size over separately trained retriever-reader models (Top for NQ and bottom for WebQ). All metrics are computed on the corresponding dev set. V stands for Verbalized here and on-wards. . . . 59
4.4 Hot-swap evaluation of raw vs verbalized table using a text-only retriever-reader model on NQ test. . . . 60
4.5 Comparison of verbalized knowledge from our verbalizer and KELM for retriever and reader on WebQ test. Dev results can be found in Table B.3 in section B.4. . . . 61
4.6 Examples of tables/chunks retrieved by our model given the question, where the evidence is bolded. In raw table, | is the row separator and empty is the filler token used by our table parsing heuristic (to make the table in good shape) . . . 61

5.1 End-to-end QA results on OTT-QA. . . . 72
5.2 End-to-end QA results on NQ test. . . . 73
5.3 Chainer ablation on OTT-QA dev. . . . 74
5.4 Reader ablation with different number of documents on OTT-QA dev. . . . 75
5.5 Evidence recall of the joint retriever on OTT-QA dev set, where R@K evaluates gold table chunk recall and AR@K evaluates answer recall. . . . 75
5.6 Answer recall on OTT-QA (top) and NQ (bottom) dev with different linking strategies. . . . 76
5.7 Example evidence chains found by our CORE, where || separates tables and passages. The answer evidence is **bold** and the linked entity mention in the table is *italic*. The first two are from NQ and the latter are from OTT-QA. . . . 76
5.8 Linker variants on OTT-QA Dev set tables. In total, there are 20,064 unique passages attached to these 789 tables . . . 77

6.1 Zero-shot top-k accuracy on test sets for NQ, WebQ and EntityQuestions, and dev set for HotpotQA. . . . 88
6.2 Supervised top-k accuracy on NQ test. . . . 89
6.3 Supervised top-k accuracy on OTT-QA dev. . . . 89
6.4 Supervised passage EM on HotpotQA dev. . . . 89
6.5 Cross-dataset top-k accuracy on test sets. . . . 91
6.6 Ablation results on HotpotQA dev using top-k retrieval accuracy. All models are initialized from BERT-base and trained on HotpotQA only. . . . 91
6.7 Results of feeding the inputs to different experts, where the first two columns represent the query expert id and document expert id. * denotes the proposed setup . . . 94
6.8 End-to-end QA results on OTT-QA. . . . 95

B.1 Statistics of Document Index . . . 134

B.2 Impact of document index size over separately trained retriever-reader models (Top for NQ and bottom for WebQ). All metrics are computed on the corresponding test set. 134
B.3 Dev set results of models trained on WebQ with verbalized WD-graph and KELM . 135
B.4 Error matrix of UDT-QA trained with text+All-tables in raw and verbalized format 135
B.5 Top: examples from DART that are filtered out by our method, the **bold** cells are omitted information from target, and *italic text* from target are likely to bias the model towards hallucination. Bottom: examples from (ID-T), which is generated by our 1st iteration verbalizer 136
B.6 Examples of verbalized table(rows) generated by different verbalizer, where the direct evidences to the answer are marked in **bold** 137

C.1 Statistics of Datasets 140
C.2 Evidence Recall of OTT-QA single retriever on OTT-QA Dev set, where R@K evaluates gold table chunk recall and AR@K evaluates answer recall 141
C.3 Answer Recall on NQ task 141

D.1 Statistics of datasets used in our experiments, columns 2-4 represent the number of questions in each split. The last two columns contain the type of training data and the corresponding number of instances 146
D.2 End-to-end QA Exact Match score on NQ 147
D.3 End-to-end QA results on Hotpot-QA. 148

Chapter 1

Introduction

1.1 Background and Motivation

The task of question answering requires the system to provide a short answer to a user question expressed in natural language. Question answering research has attracted lots of attention due to its broad applications, including commercial search engines, personal assistants and domain-specific FAQs. A typical QA pipeline mainly consists of two parts, the retriever and the reader. The retriever part is responsible for narrowing down the possible answer space from unbounded to a finite number of evidence documents, which is usually less than a few hundred. The reader then further processes the outputs of the retriever to deduce the answer. Like humans, QA systems need the knowledge to provide accurate answers, and this knowledge can be categorized as explicit or implicit. Explicit knowledge usually takes the human-readable format, such as raw text, structured tables, knowledge graphs, etc, whereas implicit knowledge is only stored in model parameters learned from training. Traditional QA systems typically only leverage explicit knowledge to answer the questions (for both retriever and reader), for example, sparse retrieval methods like BM25 (Robertson and Walker, 1994b) or TF-IDF (Chen et al., 2017) are used to find the evidence, and rule-based systems are adopted to analyze the question and derive the answers (Riloff and Thelen, 2000; Ferrucci et al., 2010). Such systems possess transparent reasoning process and efficient computation. However, since they rely on surface forms of the questions and documents (Robertson and Walker, 1994a; Robertson and Zaragoza, 2009), they would fail at capturing the relationship between questions and documents when they do not have lexical overlap.

Owing to the success of the pretrained language models (Devlin et al., 2019; Liu et al., 2019b; Raffel et al., 2019), recent works have shown that a large language model by itself already achieves very strong performance on challenging QA tasks (Brown et al., 2020; Wei et al., 2022b). These models do not require access to an explicit knowledge source, and they directly produce the answer given the question, based on the implicit knowledge encoded in their parameters. In this sense, the QA pipeline also reduces to a reader model only. Although it's good to remove the evidence retrieval stage and have a one-step QA process, these methods only work well when scaling to hundreds of billions of parameters (Chowdhery et al., 2022). Consequently, they incur a large computation and memory cost, making it hard to deploy in real-world settings. Moreover, these

models' reasoning processes are completely black-box to humans, which is also contrary to the users' need for explainable results.

Although the implicit knowledge learned by language models comes from the abstraction of the explicit knowledge in the training corpus, these two types of knowledge are often complementary to each other in practice. Intuitively, language models are good at memorizing the high-frequency patterns in the training data, and they can generalize well to similar cases at inference time. However, they struggle with long-tail facts/knowledge that are rarely observed in the data, i.e. it's hard for the models to encode/store less frequent observations as their implicit knowledge. These cases can be easily addressed by referring directly to the explicit knowledge at tasks' execution time, thus it's often desirable to incorporate both types of knowledge when building QA systems. Due to such complementary nature of explicit knowledge and implicit knowledge, numerous research efforts have been focused on combining both of them for QA (Karpukhin et al., 2020; Lin et al., 2019; Oguz et al., 2020). However, it remains an open question about how to best integrate both types of knowledge in one system because of the versatile formats of explicit knowledge and their interplay with implicit knowledge. Also, most previous work focuses on building customized models for specific datasets without paying much attention to the following three important aspects:

- **Generalization**: Within the scope of this thesis, we consider the term generalization in the following three dimensions. 1) Generalization w.r.t. vocabulary/topic domain. For example, a well-trained model should be able to generalize to both the history domain and the science domain. 2) Generalization w.r.t. format of required knowledge. As previously discussed, explicit knowledge could be presented in the format of unstructured text, tables or knowledge graphs. A single model should be able to handle questions that require reasoning over different formats of knowledge. 3) Generalization w.r.t. questions of different levels of complexity. For example, the model should be able to solve both single-hop questions and multi-hop questions. Ideally, a QA system that has acquired robust reasoning ability should be able to generalize to the aforementioned scenarios. However, several studies have pointed out that models trained on one dataset do not generalize well to other tasks (differed by topic domain). (Pugaliya et al., 2019; Talmor and Berant, 2019) When it comes to the handling of different types of knowledge, previous work usually requires dedicated modules for each knowledge format, e.g. tables-specific encoding modules (Herzig et al., 2021) or KB-specific retriever system (Oguz et al., 2020). These approaches hardly generalize to other modalities of knowledge, combining all of them may lead to an overly complicated system. Finally, previous work builds separate models for solving single-hop questions or multi-hop questions, which do not work well for other question types outside their training coverage (Chen et al., 2020d; Zhong et al., 2022). Such solutions are less applicable in realistic settings because the question types are unknown beforehand.

- **Interpretability**: Regarding interpretability, we only refer to the explicit knowledge part of the system within the scope of this thesis. In particular, we consider the following aspect of interpretability: Being able to trace the model's reasoning process, i.e. understanding how the model arrives at its decision at every step. Ideally, a trustworthy QA system should be able to explain its reasoning step by step much like a human. Interpretability is especially important for solving complex questions, as the model would need multiple steps of reasoning

to infer the answer, i.e. multiple pieces of evidence. Having a transparent reasoning process would enable trustworthy predictions and easy error analysis. Previous works on complex QA did not pay much attention to these aspects of the system. One popular approach is to only rely on language models to encode the question and context document to automatically build up the reasoning chain (Xiong et al., 2020b; Khattab et al., 2021), without considering the relationship across documents. However, it's hard to localize the clue that bridges individual pieces of evidence. This makes it hard to understand the cause of errors when the model makes mistakes and contradicts the users' need for explainable results.

- **Efficiency**: Within this thesis, we consider efficiency in terms of memory, computation and storage cost. Specifically, it's desirable to have a smaller overall model size such that the memory footprint and storage cost can be reduced. Consequently, the model would be able to run in environments with limited computation resources. Moreover, it's desired to have reduced computation, as it could also lead to improved inference speed, making it more applicable to real-world use cases. Previous research in QA mostly focused on getting state-of-the-art accuracy, without concerning too much about the model sizes and computation, (Izacard and Grave, 2021; Sanh et al., 2021), which makes them infeasible to use in many scenarios. Besides the single model's size, when the overall QA system contains multiple modules, these modules are often separately trained and used (Ma et al., 2022b), leading to increased redundancy.

1.2 Concepts and Terminology

We first define and introduce some of the frequently used concepts and terminology in this thesis.

- **Transformers** (Vaswani et al., 2017) is a recently introduced neural network architecture that has revolutionized the NLP field. The basic building block of Transformers is a transformer layer. A transformer layer takes a sequence of encoded token representations as input. It first computes the multi-head self-attention of the inputs, followed by layer normalization (Ba et al., 2016). Then it projects the encoded representations with two feed-forward layers, followed by another layer normalization. Formally, let $\mathbf{h} \in \mathrm{R}^{m \times d}$ be the embedded token sequence where m represent the number of tokens and d represent the hidden dimension:

$$Q = \mathbf{h} \cdot W^q \tag{1.1}$$
$$K = \mathbf{h} \cdot W^k \tag{1.2}$$
$$V = \mathbf{h} \cdot W^v \tag{1.3}$$
$$\text{Attn} = \text{softmax}(\frac{QK^T}{\sqrt{d}})V \tag{1.4}$$
$$\text{SelfOut} = \text{LayerNorm}(h + \text{Attn} \cdot W^c) \tag{1.5}$$
$$\text{Out} = \text{LayerNorm}(\text{SelfOut} + (\text{ACTFN}(\text{SelfOut} \cdot W^1) \cdot W^2)) \tag{1.6}$$

where $W^q, W^k, W^v, W^c \in \mathrm{R}^{d \times d}$, $W^1 \in \mathrm{R}^{d \times 4d}$, $W^2 \in \mathrm{R}^{4d \times d}$ are learnable weights. ACTFN

is a non-linear activation function. In practice, multiple transformer layers are stacked together to form a transformer encoder, and the model is usually trained end-to-end.

- **BERT** (Devlin et al., 2019) is a bi-directional encoder-only transformer model. The model consists of stacked transformer layers (12 layers for base size and 24 layers for large size) and is pretrained on large text corpora with self-supervision only. In particular, the model is pretrained with masked language modeling (MLM) and next-sentence prediction (NSP) objectives. For MLM, 15% of tokens in the input sequence are randomly masked and the model is trained to predict the masked tokens. For NSP, two sentences are randomly sampled and the model is trained to predict if they are consecutive sentences in the original documents. After pretraining, the model can be finetuned for many NLP tasks by simply adding a task-specific output layer on top.

- **Bi-Encoder** is a popular architecture adopted by many recent works on dense retrievers (Karpukhin et al., 2020; Wu et al., 2020). The overall model usually consists of two BERT-style encoders, one for encoding questions and the other for encoding documents. The model is trained with a contrastive learning objective, where the model learns to map the positive documents closer to the question and push away the negative documents. Formally,

$$L = -\frac{\exp(\text{sim}(\mathbf{q}, \mathbf{d}^+))}{\sum_{\mathbf{d}' \in \mathcal{D} \cup \{\mathbf{d}^+\}} \exp(\text{sim}(\mathbf{q}, \mathbf{d}'))} \tag{1.7}$$

  where $\mathbf{q} \in \mathrm{R}^d$ is the encoded question vector, $\mathbf{d}^+, \mathbf{d}' \in \mathrm{R}^d$ are the encoded positive document vector and negative document vector respectively, $\mathcal{D}$ is the document sets (usually all documents within the same batch). In practice, the randomly sampled negatives are easy to learn and distinguish, so harder negatives that provide more signals Xiong et al. (2020a) are mined to train the model.

- **Reader** is the model responsible for predicting the final answer given the question and provided evidence documents. There are two classes of reader models, namely extractive reader and generative reader. For the extractive reader, the model extracts a span of text from the evidence document as the answer. In practice, the model consists of an encoder to process the input question and documents, which can be a BERT model. Then two linear layers are added on top of the encoder to predict the start and end token indices of the answer span. Formally,

$$P_{Start} = \text{Softmax}(C \cdot W^s) \tag{1.8}$$

$$P_{End} = \text{Softmax}(C \cdot W^e), \tag{1.9}$$

  where $W^s, W^e \in \mathrm{R}^d$ are learnable weights and $C \in \mathrm{R}^{m \times d}$ is the output sequence from the encoder. For the generative reader, the model similarly encodes the input question and documents with a BERT-like model. Then the model generative the answer token-by-token in an autoregressive fashion. The extractive reader possesses the advantage of always grounding the predicted answer to input documents, thus preventing hallucination by design. However,

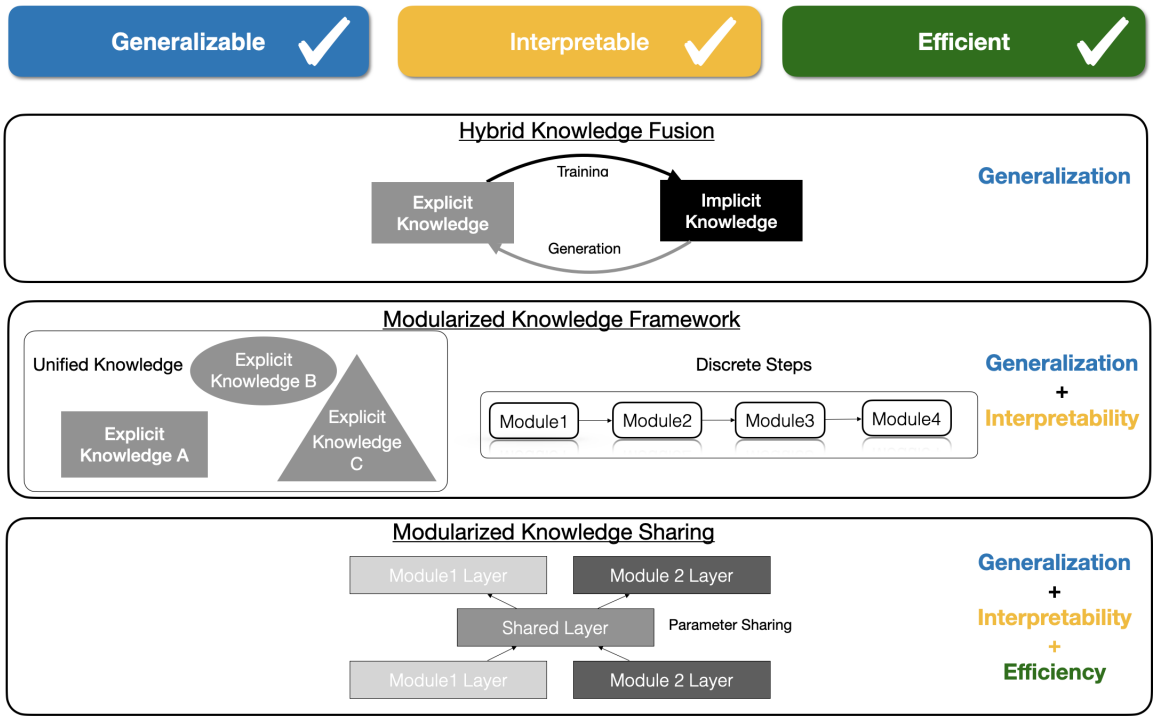

Figure 1.1: The Overview of this thesis

it cannot handle cases where no span of the input text can answer the question, i.e. the information needs to be gathered and fused from different places. On the other hand, the generative reader is able to generate novel answers not found within the documents, which offers greater flexibility and addresses the shortcomings of the extractive reader. However, they suffer from hallucinations. Thus, efforts have been made to combine the advantage of both kinds of reader models by ensemble Cheng et al. (2021b).

- **Close-book** and **Open-book** are two popular settings studied by recent QA literature. In the close-book setting, the model only has the reader component, and all the necessary knowledge required to answer the question is assumed to be given or already learned, similar to a human close-book test. A popular kind of close-book task setup is also called **Reading Comprehension**. In this setting, the model is given a context paragraph along with the question. Thus the main challenge lies in understanding the context. For open-book settings, the model has both retriever and reader components. The model is only given the question, and it has access to external corpora where it can find evidence documents that help answer the question, similar to a human open-book test. In this thesis, we study both close-book settings and open-book settings, and we put more focus on open-book settings due to their proximity to real-world scenarios.

	Explicit Knowledge	Implicit Knowledge	Generalizability	Interpretability	Efficiency
Retriever	✅	✅	?	?	?
Reader	✅	✅	?	?	?

Figure 1.2: Overview of the Aspects of the QA system that this thesis focus on

1.3 Thesis Overview

With the aforementioned issues in mind, in this thesis, we introduce solutions for building generalizable, interpretable, and efficient QA systems. We show that *modularized systems that incorporate diverse explicit knowledge and allow implicit knowledge sharing are able to generalize across different domains, knowledge types, and question types, offer interpretable reasoning processes, and achieve improved modeling efficiency*. To this end, we propose three solution elements towards this goal, namely hybrid knowledge fusion, modularized knowledge framework, and modularized knowledge sharing. As illustrated in Figure 1.1, each of these elements builds on top of the previous ones and makes a step forward towards encompassing all aspects of the ideal system. In Figure 1.2, we show the grid view of the components of this thesis. As previously discussed, a QA system mainly consists of the retriever part and the reader part. For the boxes with check marks, we assume that the foundations we built upon have already reached the target state, i.e. both the retriever and the reader model have already gained implicit knowledge from pretraining, and the retriever has gained access to a corpus of explicit knowledge. Then we aim to turn the remaining question marks in the grid into check marks, by proposing the three solution elements.

Next, we present the overview of these three solution elements.

- **Hybrid Knowledge Fusion**: In this part, we study the interplay between explicit knowledge and implicit knowledge, in the context of multiple-choice commonsense QA tasks. In particular, we present various methods for injecting the symbolic knowledge from the commonsense knowledge graphs into pretrained language models, with a focus on improving the generalizability of the reader component across different domains.
  - In chapter 2, We proposed different ways of knowledge elicitation based on the characteristics of the knowledge graph, including both heuristic-based extraction and free-form generation (HyKAS). We focused on the supervised setting and experimented with an attention-based late-fusion method and pretraining the model on flattened KG. Our results show that attention injection is more robust when the alignment between the QA tasks and the knowledge graphs is unknown, and naive pretraining could lead to catastrophic forgetting. This work is published in the COIN workshop at EMNLP 2019 (Ma et al., 2019).
  - In chapter 3, we proposed a zero-shot QA framework that transforms a consolidated commonsense knowledge graph into synthetic QA sets, and trains language models on the synthetic QA data (HyKAS-CSKG). By covering a variety of commonsense

knowledge graphs, we show that the resulting model can generalize to a diverse set of tasks focusing on different aspects of commonsense and achieve significant gains over previous baselines in a zero-shot setting. This work is published at AAAI 2021 (Ma et al., 2020).

Overall these two chapters contribute methods that effectively combine the explicit knowledge graphs and implicit knowledge from language models, which improved the reader's generalizability on commonsense QA tasks across different domains. However, the first part of the thesis has only focused on commonsense QA tasks that contain mostly single-hop questions and leveraged knowledge graphs that are limited in scale. Moreover, the retriever component, which is essential for interpretable explicit knowledge, is understudied. These observations motivate us to dive into the next solution element of the thesis.

- **Modularized Knowledge Framework**: In this part, we present two modularized frameworks for solving open-domain question answering (ODQA) tasks, UDT-QA and CORE. One for building a unified knowledge interface for different kinds of web-scale structured knowledge, and another for decomposing the QA reasoning process into discrete steps such that one single system can handle both single-hop and multi-hop questions. Here, we improve the generalizability of both the retriever and the reader by enabling them to handle heterogeneous structured knowledge and questions of different complexity, and we further enhance the interpretability of the retriever with discrete reasoning operations.

  - In chapter 4, we proposed a verbalizer-retriever-reader framework for ODQA (UDT-QA). We proposed to treat text as a universal knowledge interface and introduced a novel verbalizer model to convert structured tables and knowledge base subgraphs into text. Consequently, we can use one single retriever-reader system to answer the questions without relying on knowledge type-specific modules. Our results show that our verbalizer works well for both tables and KB and the verbalized knowledge brings consistent gains over the raw knowledge format. This work is published at ACL 2022 (Ma et al., 2022b).
  - In chapter 5, we proposed a chain-of-reasoning framework (CORE) that decomposes the overall QA reasoning process into discrete operations: retrieve, link, rerank, and read. This formulation naturally covers both simple questions and complex questions' reasoning pipelines. Within this framework, each reasoning step can be implemented by a separate module, and each module operates on the outputs of previous modules which offers transparent intermediate results. We show that our framework can solve both single-hop questions and multi-hop questions using one single model. Our framework also offers better interpretability by enabling the trace of the expanding entity mentions in the tables. This work is published at EMNLP 2022 (Findings) (Ma et al., 2022a).

Overall, the proposed method from these two chapters led to generalization across different modalities of knowledge and different question types for both the retriever and reader, and we also gained improved interpretability through reasoning decomposition and knowledge

verbalization. However, one problem arises that our framework contains many individual modules, which are completely separated. Consequently, storing and running all of these modules may incur a large computation burden. Motivated by this observation, we proceed to the next solution element of this thesis where we try to improve the efficiency of the system.

- **Modularized Knowledge Sharing**: In this part, we build QA systems that retain the advantages mentioned in previous chapters, while achieving better modeling efficiency. We observe that some components of the QA system proposed in the previous chapter share similar model architectures or objectives, thus it presents an opportunity to merge them together. In this way, we are able to improve the retriever components' memory and computation efficiency while keeping its generalization and interpretability.

  - In chapter 6, we proposed a chain-of-skills framework (COS) that learns multiple reasoning skills jointly within one shared encoder, including single-retrieval, expanded-retrieval, span-detection, entity linking and reranking skills. This essentially merged individual modules from the CORE framework in Chapter 5 into one model, therefore significantly reducing the model size. On top of this, we introduced novel ways of reformulating the task training objectives and added skill-specific specialization layers to reduce task interference. The model is trained with multi-task learning. Once trained, we can flexibly configure the inference pipeline for solving seen and unseen tasks, i.e. by chaining the skills in different order or combination. Therefore, the model achieves greater generalization to different tasks, and the chain-of-skill inference naturally offers interpretable intermediate results. Moreover, merging the modules enables one forward pass to produce representations used for multiple skills, which greatly improves computation efficiency. This work is published at ACL 2023 (Ma et al., 2023).

Chapter 2

Towards Generalizable Neuro-Symbolic Systems for Commonsense Question Answering

Language models like BERT (Devlin et al., 2019) are pretrained on large corpora of text, and they have acquired a decent amount of implicit knowledge from the pretraining process. Consequently, they serve as good bases for building task-specific models. However, previous studies have pointed out that these models still lack appropriate knowledge for solving certain reasoning tasks, e.g. commonsense question answering, which hinders their applicability. In this chapter, we study methods that incorporate additional explicit knowledge into these models for building more generalizable QA systems. In particular, we proposed different methods for eliciting suitable knowledge from different commonsense knowledge graphs. We also experimented with various ways of fusing this explicit knowledge with the model's implicit knowledge (hybrid knowledge). Our results show that the attention-based late-fusion method is a more robust method for knowledge fusion compared to naive pretraining.

2.1 Introduction

With the recent success of large pre-trained language models (Devlin et al., 2019; Radford et al., 2019; Yang et al., 2019; Liu et al., 2019b), model performance has reached or surpassed human-level capability on many previous question-answering (QA) benchmarks (Hermann et al., 2015; Rajpurkar et al., 2016; Lai et al., 2017). However, these benchmarks do not directly challenge model reasoning capability, as they require only marginal use of external knowledge to select the correct answer, i.e., all the evidence required to solve questions in these benchmarks is explicit in the context lexical space. Efforts have been made towards building more challenging datasets that, by design, require models to synthesize external commonsense knowledge and leverage more sophisticated reasoning mechanisms (Zhang et al., 2018; Ostermann et al., 2018), showing that the previous state-of-the-art models often struggle to solve these newer tasks reliably. As a result, commonsense has received a lot of attention in other areas as well, such as natu-

ral language inference (Zellers et al., 2018b, 2019) and visual question answering (Zellers et al., 2018a). Despite the importance of commonsense knowledge, however, previous work on QA methods takes a coarse-grained view of commonsense, without considering the subtle differences across the various knowledge types and resources. Such differences have been discussed at length in AI by philosophers, computational linguists, cognitive psychologists (see for instance (Davis, 2014)): at the high level, we can identify *declarative commonsense*, whose scope encompasses factual knowledge, e.g., 'the sky is blue', 'Paris is in France'; *taxonomic knowledge*, e.g., 'football players are athletes', 'cats are mammals'; *relational knowledge*, e.g., 'the nose is part of the skull', 'handwriting requires a hand and a writing instrument'; *procedural commonsense*, which includes prescriptive knowledge, e.g., 'one needs an oven before baking cakes', 'the electricity should be off while the switch is being repaired' (Hobbs et al., 1987); *sentiment knowledge*, e.g., 'rushing to the hospital makes people worried', 'being in vacation makes people relaxed'; and *metaphorical knowledge* (e.g., 'time flies', 'raining cats and dogs'). We believe that it is important to identify the most appropriate commonsense knowledge type required for specific tasks, in order to get better downstream performance. Once the knowledge type is identified, we can then select the appropriate knowledge base(s), and the suitable neural integration mechanisms (e.g., attention-based injection, pre-training, or auxiliary training objectives).

Accordingly, in this work we conduct a comparison study of different knowledge bases and knowledge integration methods, and we evaluate model performance on two multiple-choice QA datasets that explicitly require commonsense reasoning. In particular, we used ConceptNet (Speer et al., 2017) and the recently-introduced ATOMIC (Sap et al., 2019a) knowledge resources, integrating them with the *Option Comparison Network* model (OCN; Ran et al. (2019)), a recent state-of-the-art model for multiple choice QA tasks. We evaluate our models on the `DREAM` (Sun et al., 2019b) and `CommonsenseQA` (Talmor et al., 2019) datasets. An example from `DREAM` that requires commonsense is shown in Table 2.1, and an example from `CommonsenseQA` is shown in Table 2.2. Our experimental results and analysis suggest that attention-based injection is preferable for knowledge integration and that the degree of domain overlap, between knowledge-base and dataset, is vital to model success.[1]

Dialogue:
M: I hear you drive a long way to work every day.
W: Oh, yes. it's about sixty miles. but it doesn't seem that far, the road is not bad, and there's not much traffic.
Question:
How does the woman feel about driving to work?
Answer choices:
A. She doesn't mind it as the road conditions are good.*
B. She is unhappy to drive such a long way everyday.
C. She is tired of driving in heavy traffic.

Table 2.1: An example from the `DREAM` dataset; the asterisk (*) denotes the correct answer.

[1]From a terminological standpoint, 'domain overlap' here must be interpreted as the overlap between question types in the targeted datasets, and types of commonsense represented in the knowledge bases under consideration.

Question: A revolving door is convenient for two direction travel, but it also serves as a security measure at a what? **Answer choices:** A. Bank*; B. Library; C. Department Store; D. Mall; E. New York

Table 2.2: An example from the `CommonsenseQA` dataset; the asterisk (*) denotes the correct answer.

2.2 Related Work

It has been recognized that many recent QA tasks require external knowledge or commonsense to solve, and numerous efforts have been made in injecting commonsense into neural models. Bauer et al. (2018) introduced a pipeline for extracting grounded multi-hop commonsense relation paths from ConceptNet and proposed to inject commonsense knowledge into neural models' intermediate representations, using attention. Similarly, Mihaylov and Frank (2018) also proposed to extract relevant knowledge triples from ConceptNet and use Key-Value Retrieval (Miller et al., 2016) to gather information from knowledge to enhance the neural representation. Zhong et al. (2018) proposed to pre-train a scoring function using knowledge triples from ConceptNet, to model the direct and indirect relation between concepts. This scoring function was then fused with QA models to make the final prediction. Pan et al. (2019) introduced an entity discovery and linking system to identify the most salient entities in the question and answer options. Wikipedia abstracts of these entities are then extracted and appended to the reference documents to provide additional information. Weissenborn et al. (2018) proposed a strategy of dynamically refining word embeddings by reading input text as well as external knowledge, such as ConceptNet and Wikipedia abstracts. More recently, Lin et al. (2019) proposed to extract sub-graphs from ConceptNet and embed the knowledge using Graph Convolutional Networks (Kipf and Welling, 2016). Then the knowledge representation is integrated with word representation through an LSTM layer and hierarchical attention mechanism. Lv et al. (2019) introduced graph-based reasoning modules that take both ConceptNet knowledge triples and Wikipedia text as inputs to refine word representations from a pretrained language model and make predictions.

Commonsense knowledge integration has also received a lot of attention on many other tasks. Tandon et al. (2018) proposed to use commonsense knowledge as hard/soft constraints to bias the neural model's prediction on a procedural text comprehension task. Ma et al. (2018) proposed to use embedded affective commonsense knowledge inside the LSTM cell to control the information flow in each gate for the sentiment analysis task. Li and Srikumar (2019) presented a framework to convert declarative knowledge into first-order logic that enhances neural networks' training and prediction. Peters et al. (2019) and Levine et al. (2020) both tried to inject knowledge into language models by pretraining on knowledge bases.

Previous works only focus on using external knowledge sources to improve model performance on certain tasks, disregarding the type of commonsense knowledge and how the domain of the knowledge resource affects results on downstream tasks. In this paper, we examine the roles of the

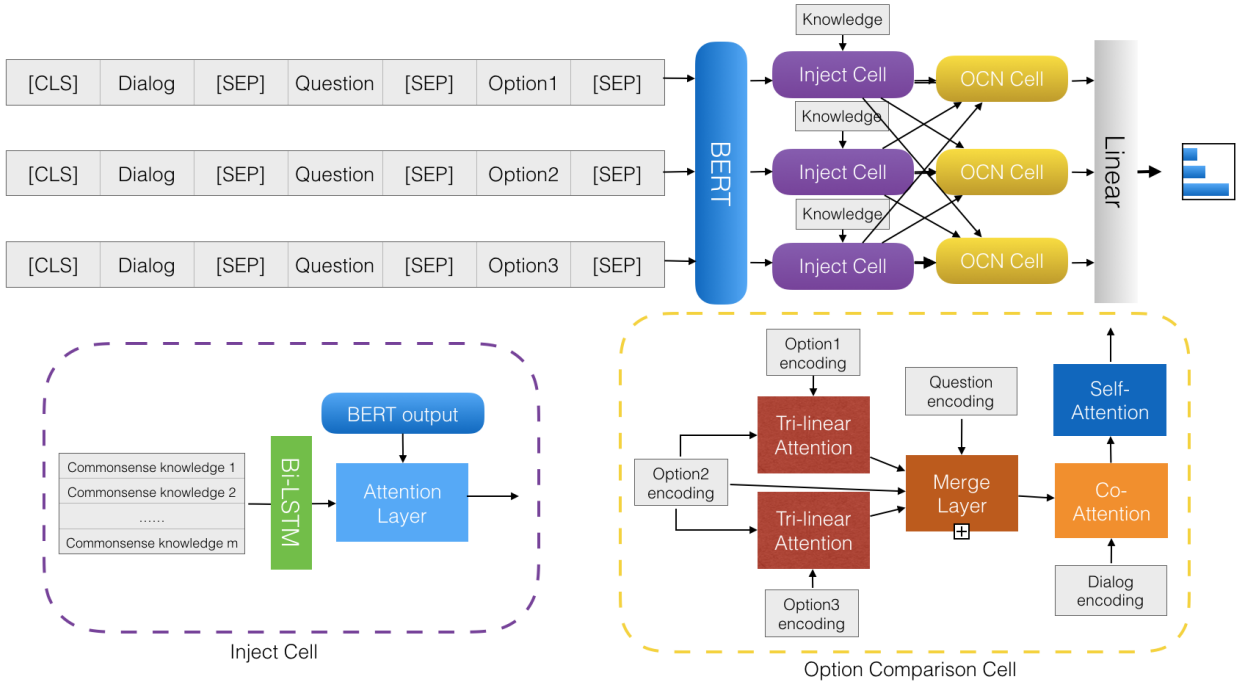

Figure 2.1: Option Comparison Network with Knowledge Injection

knowledge-base domain and specific integration mechanisms on model performance.

2.3 Approach Overview

In this section, we describe the model architecture used in our experiments. Next, we introduce two popular knowledge resources, we define our knowledge-extraction method, then we illustrate various neural knowledge-integration mechanisms.

2.3.1 Model architecture

The BERT model (Devlin et al., 2019) has been applied to numerous QA tasks and has achieved very promising performance, particularly on the `DREAM` and `CommonsenseQA` datasets. When utilizing BERT on multiple-choice QA tasks, the standard approach is to concatenate the dialogue context and the question with each answer-option, in order to generate a list of tokens which is then fed into the BERT encoder; a linear layer is added on top, in order to predict the answer. One aspect of this strategy is that each answer option is encoded independently: from a cognitive perspective, this aspect contradicts how humans typically solve multiple-choice QA tasks, namely by *weighing* each option to find correlations within them, in addition to correlations with respect to the question. To address this issue, Ran et al. (2019) introduced the *Option Comparison Network* (OCN) that explicitly models pairwise answer-option interactions, making OCN better-suited for multiple-choice QA task structures. We re-implemented OCN while keeping BERT as its upstream

encoder.[2] Specifically, given a dialogue D, a question Q, and an answer-option O_k, we concatenate them and encode with BERT to get hidden representation $T_{enc} \in \mathbb{R}^{n \times d}$:

$$T_{enc} = \text{BERT}(D; Q; O_k) \tag{2.1}$$

Where d is the size of BERT's hidden representation and n is the total number of words. Next, the dialogue encoding $D_{enc} \in \mathbb{R}^{n_d \times d}$, question encoding $Q_{enc} \in \mathbb{R}^{n_q \times d}$, and answer-option encoding $O_{k,enc} \in \mathbb{R}^{n_o \times d}$ are separated from T_{enc}. Here, option-encoding consists both of question and option, i.e. $Q_{enc} \subseteq O_{k,enc}$ and $n_d + n_o = n$, as suggested by Ran et al. (2019). Given a set of options O_k ($k = 1, 2, ...$), these options are compared, pairwise, using standard tri-linear attention (Seo et al., 2016):

$$\text{Att}(u, v) = W_1 \cdot u + W_2 \cdot v + (W_3 \circ v) \cdot u \tag{2.2}$$

Where, $W_1, W_2, W_3 \in \mathbb{R}^d$ are trainable weights and $u \in \mathbb{R}^{x \times d}$, $v \in \mathbb{R}^{y \times d}$ are input matrices; x and y here are generic placeholder for input lengths; matrix multiplication and elementwise multiplication are denoted by $(\cdot)$ and $(\circ)$, respectively. Next, we gather information from all other options, to form a new option representation $O_{k,new} \in \mathbb{R}^{n_o \times d}$. Formally, given option $O_{k,enc}$ and another option $O_{l,enc} \in \mathbb{R}^{n_l \times d}$, $O_{k,new}$ is computed as follows:

$$O_k^l = O_{l,enc} \cdot \text{Att}(O_{l,enc}, O_{k,enc}) \tag{2.3}$$

$$\widetilde{O_k^l} = [O_{k,enc} - O_k^l; O_{k,enc} \circ O_k^l] \tag{2.4}$$

$$O_{k,new} = \tanh(W_c \cdot [O_{k,enc}; \{\widetilde{O_k^l}\}_{l \neq k}]) \tag{2.5}$$

Where, $W_c \in \mathbb{R}^{(d+2d(|O|-1)) \times d}$, $|O|$ denotes total number of options and n_l denotes the number of words in the compared option. Then, a gating mechanism is used to fuse the option-wise correlation information $O_{k,new}$ with the current option-encoding $O_{k,enc}$. Gating values are computed as:

$$G = \text{sigmoid}(W_g[O_{k,enc}; O_{k,new}; \widetilde{Q}]) \tag{2.6}$$

$$\widetilde{Q} = Q_{enc} \cdot \text{softmax}(Q_{enc} \cdot V_a)^T \tag{2.7}$$

$$O_{fuse} = G \circ O_{k,enc} + (1 - G) \circ O_{k,new} \tag{2.8}$$

[2]Because the newly-released XLNet has outperformed BERT on various tasks, we considered using XLNet as the OCN's encoder. However, from our initial experiments, XLNet is very unstable, in that it easily provides degenerate solutions—a problem noted by Devlin et al. (2019) for small datasets. We found BERT to be more stable in our study.

Here, $W_g \in \mathbb{R}^{3d \times d}$ and $V_a \in \mathbb{R}^{d \times 1}$. Co-attention (Xiong et al., 2016) is applied to re-read the dialogue, given the fused option-correlation features:

$$A_{do} = \text{Att}(D_{enc}, O_{fuse}) \tag{2.9}$$
$$A_{od} = \text{Att}(O_{fuse}, D_{enc}) \tag{2.10}$$
$$O_d = A_{od} \cdot [D_{enc}; A_{do} \cdot O_{fuse}] \tag{2.11}$$
$$\widetilde{O_d} = \text{ReLU}(W_p([O_d; O_{fuse}])) \tag{2.12}$$

Here, $W_p \in \mathbb{R}^{3d \times d}$. Finally, self-attention (Wang et al., 2017) is used to compute final option representation $\widetilde{O_f} \in \mathbb{R}^{n_o \times d}$:

$$O_s = \widetilde{O_d} \cdot \text{Att}(\widetilde{O_d}, \widetilde{O_d}) \tag{2.13}$$
$$O_f = [\widetilde{O_d}; O_s, \widetilde{O_d} - O_s; \widetilde{O_d} \circ O_s] \tag{2.14}$$
$$\widetilde{O_f} = \text{ReLU}(W_f \cdot O_f) \tag{2.15}$$

Unlike the vanilla BERT model, which takes the first token to predict the answer, max-pooling is applied on the sequence dimension of $\widetilde{O_f} \in \mathbb{R}^{n_o \times d}$, in order to generate the final prediction.

2.3.2 Knowledge bases

The first knowledge base we consider for our experiments is ConceptNet (Speer et al., 2017). ConceptNet contains over 21 million edges and 8 million nodes (1.5 million nodes in the partition for the English vocabulary), generating triples of the form $(C1, r, C2)$: the natural-language concepts $C1$ and $C2$ are associated by commonsense relation r, e.g., *(dinner, AtLocation, restaurant)*. Thanks to its coverage, ConceptNet is one of the most popular semantic networks for commonsense. ATOMIC (Sap et al., 2019a) is a new knowledge base that focuses on procedural knowledge. Triples are of the form *(Event, r, {Effect|Persona|Mental-state})*, where head and tail are short sentences or verb phrases and r represents an *if-then* relation type. An example would be: *(X compliments Y, xIntent, X wants to be nice)*. Since both `DREAM` and `CommonsenseQA` datasets are open-domain and require general commonsense, we think these knowledge bases are most appropriate for our investigation.

2.3.3 Knowledge elicitation

ConceptNet. For the `DREAM` dataset, we find ConceptNet relations that connect dialogues and questions to the answer options. The intuition is that these relation paths would provide explicit evidence that would help the model find the answer. Formally, given a dialogue D, a question Q, and an answer-option O, we find all ConceptNet relations *(C1, r, C2)*, such that $C1 \in (D + Q)$ and $C2 \in O$, or vice versa. This rule works well for single-word concepts. However, a large number of concepts in ConceptNet are actually phrases, and finding exactly matching phrases in $D/Q/O$ is much harder. To fully utilize phrase-based ConceptNet relations, we relaxed the exact-match

constraint to the following:

$$\frac{\text{\# words in C} \cap \text{S}}{\text{\# words in C}} > 0.5 \tag{2.16}$$

Here, S represents $D/Q/O$, depending on which sequence we try to match the concept C to. Additionally, when the part-of-speech (POS) tag for a concept is available, we make sure it matches the POS tag of the corresponding word in $D/Q/O$. For `CommonsenseQA`, we use the same procedure to find ConceptNet relations for each answer option, except that only Q is present and used. Table 2.3 shows the extracted ConceptNet triples for the `CommonsenseQA` example in Table 2.2. It is worth noting that we are able to extract the original ConceptNet sub-graph that was used to create the question, along with some extra triples. Although not perfect, the bold ConceptNet triple does provide some clue that could help the model resolve the correct answer.

Options	Extracted ConceptNet triples
Bank	(revolving door *AtLocation* bank) **(bank RelatedTo security)**
Library	(revolving door *AtLocation* library)
Department Store	(revolving door *AtLocation* store) (security IsA department)
Mall	(revolving door *AtLocation* mall)
New York	(revolving door *AtLocation* New York)

Table 2.3: Extracted ConceptNet relations for sample shown in Table 2.2.

Input sentence	Generated ATOMIC relations
Utterance 1	(xAttr dedicated) (xWant to get to work)
Utterance 2	(xAttr far) **(xReact happy)** (xWant to get to their destination)
Option A	**(xAttr calm)** (xWant to avoid the road)
Option B	**(xAttr careless)** (xReact annoyed) (xEffect get tired)
Option C	**(xAttr frustrated)** (xEffect get tired) (xWant to get out of car)

Table 2.4: Sample generated ATOMIC relations for sample shown in Table 2.1.

ATOMIC. We observe that many questions in `DREAM` inquire about the agent's opinion and feelings. Superficially, this particular question type seems well-suited for ATOMIC, whose focus is on folk psychology and related general implications; we could frame our goal as evaluating whether ATOMIC can provide relevant knowledge to help answer these questions. However, one challenge to this strategy is that heads and tails of knowledge triples in ATOMIC are short sentences or verb phrases, while rare words and person-references are reduced to blanks and PersonX/PersonY, respectively. This calls for a new matching procedure, different from the ConceptNet extraction strategy, for eliciting ATOMIC-specific relations: we rely on the recently-published COMET model (Bosselut et al., 2019) to generate new ATOMIC relations, with intermediate phrasal resolutions. In particular, we first segmented all dialogues, questions, and answer options into sentences. We further segment long sentences into sub-sentences, using commas as separators. Because only verb phrases satisfy the definition of an "event" in ATOMIC (i.e., relations are only invoked by

verbs), we remove all sentences/sub-sentences that do not contain any verb. Next, we use a pre-trained COMET model (Bosselut et al., 2019) to generate all possible ATOMIC relations, for all candidate sentences/sub-sentences and we use greedy-decoding to take the 1-best sequences. Table 2.4 shows the sample ATOMIC relations, generated using the DREAM example in Table 2.1. It is interesting to note that the reaction for the woman agent (second utterance) is identified as *happy*, since she said that 'the road is not bad.' If we compare the identified attributes for answer options, the one from the correct answer seems to be semantically closer than the other two.

2.3.4 Knowledge injection

Given previously extracted/generated knowledge triples, we need to integrate them with the OCN model. Inspired by Bauer et al. (2018), we propose to use attention-based injection. For ConceptNet knowledge triples, we first convert concept-relation tokens into regular tokens, in order to generate a pseudo-sentence. For example, "*(book, AtLocation, library)*" would be converted to "book at location library." Next, we use the BERT embedding layer to generate an embedding of this pseudo-sentence, with C denoting a ConceptNet relation:

$$H_C = \text{BiLSTM}(C) \tag{2.17}$$

If we let $H_C \in \mathbb{R}^{1\times 2l}$ be the concatenation of the final hidden states and l be the number of hidden units in the LSTM layer, then m ConceptNet relations would yield the commonsense knowledge matrix $H_M \in \mathbb{R}^{m\times 2l}$. We adopt the attention mechanism used in QAnet (Yu et al., 2018b) to model the interaction between H_M and the BERT encoding output T_{enc} (from Equation 2.1):

$$\widetilde{H}_M = H_M \cdot W_{proj} \tag{2.18}$$
$$\mathcal{S} = \text{Att}(H_M, T_{enc}) \tag{2.19}$$
$$A_m = \text{softmax}(\mathcal{S}) \cdot \widetilde{H}_M \tag{2.20}$$
$$A_t = \text{softmax}(\mathcal{S}) \cdot \text{softmax}(\mathcal{S}^T) \cdot T_{enc} \tag{2.21}$$
$$T_C = [T_{enc}; A_m; T_{enc} \circ A_m; T_{enc} \circ A_t] \tag{2.22}$$
$$T_{out} = \text{ReLU}(T_C \cdot W_a) \tag{2.23}$$

Specifically, H_M is first projected into the same dimension as T_{enc}, using $W_{proj} \in \mathbb{R}^{2l\times d}$. Then, the similarity matrix $\mathcal{S} \in \mathbb{R}^{n\times m}$ is computed using tri-linear attention, as in Equation 2.2. We then use $\mathcal{S}$ to compute text-to-knowledge attention $A_m \in \mathbb{R}^{n\times d}$ and knowledge-to-text attention $A_t \in \mathbb{R}^{n\times d}$. Finally, the knowledge-aware textual representation $T_{out} \in \mathbb{R}^{n\times d}$ is computed, where $W_a \in \mathbb{R}^{4d\times d}$. T_{out} is fed to subsequent layers (in place of T_{enc}), in order to generate the prediction. The model structure with knowledge injection is summarized in Figure 2.1.

For ATOMIC knowledge triples, the injection method is slightly different. Because heads of these knowledge triples are sentences/utterances and the tails contain attributes of the persons (i.e., subject and object of the sentence), it is not possible to directly inject the knowledge triples, as-is. We replace the heads of the ATOMIC knowledge triples with the corresponding speaker for dialogues and leave them as blank for the answer-options. Next, we convert the special rela-

tion tokens into regular tokens, e.g., “xIntent”⇒“intent" and “oEffect"⇒ “others effect”, to make pseudo-sentences. As a result, an ATOMIC relation “*(the road is not bad, xReact, happy)*" would be converted to “*(W, react, happy)*." Moreover, as the ATOMIC knowledge triples are associated with dialogues and answer options, independently, we inject option relations into $O_{enc} \in \mathbb{R}^{n_o \times d}$ and dialogue relations into D_{enc}, respectively, using the injection method described above.

2.3.5 Knowledge pre-training

Pre-training large-capacity models (e.g., BERT, GPT (Radford et al., 2019), XLNet (Yang et al., 2019)) on large corpora, then fine-tuning on more domain-specific information, has led to performance improvements on various tasks. Inspired by this, our goal in this section is to observe the effect of pre-training BERT on commonsense knowledge and refine the model on task-specific content from our DREAM and CommonsenseQA corpora. Essentially, we would like to test if pre-training on our external knowledge resources can help the model acquire commonsense. For the ConceptNet pre-training procedure, pre-training BERT on pseudo-sentences formulated from ConceptNet knowledge triples does not provide much gain on performance. Instead, we trained BERT on the *Open Mind Common Sense* (OMCS) corpus (Singh et al., 2002), the original corpus that was used to create ConceptNet. We extracted about 930K English sentences from OMCS and randomly masked out 15% of the tokens; we then fine-tuned BERT, using a masked language model objective. Then we load this fine-tuned model into OCN and train on DREAM and CommonsenseQA tasks. As for pre-training on ATOMIC, we again use COMET to convert ATOMIC knowledge triples into sentences; we created special tokens for 9 types of relations as well as blanks. Next, we randomly masked out 15% of the tokens, only masking out tail-tokens. We use the same OMCS pre-training procedure.

Models	Dev Acc	Test Acc
BERT Large(*)	66.0	66.8
XLNet(*)	-	**72.0**
OCN	70.0	69.8
OCN + CN injection	**70.5**	69.6
OCN + AT injection	69.6	**70.1**
OCN + OMCS pre-train	64.0	62.6
OCN + ATOMIC pre-train	60.3	58.8

Table 2.5: Results on DREAM; the asterisk (*) denotes results taken from leaderboard.

2.4 Experiments

2.4.1 Datasets

We choose to evaluate our hypotheses using the DREAM and CommonsenseQA datasets, because some / all questions require commonsense reasoning and because there remains a large gap between state-of-the-art models and human performance.

Models	Dev Acc
BERT + OMCS pre-train(*)	68.8
RoBERTa + CSPT(*)	**76.2**
OCN	64.1
OCN + CN injection	67.3
OCN + OMCS pre-train	65.2
OCN + ATOMIC pre-train	61.2
OCN + OMCS pre-train + CN inject	**69.0**

Table 2.6: Results on CommonsenseQA; the asterisk (*) denotes results taken from leaderboard.

DREAM is a dialogue-based multiple-choice QA dataset, introduced by Sun et al. (2019b). It was collected from English-as-a-foreign-language examinations, designed by human experts. The dataset contains 10,197 questions for 6,444 dialogues in total, and each question is associated with 3 answer-options. The authors point out that 34% of questions require commonsense knowledge to answer, which includes social implication, speaker's intention, or general world knowledge.

CommonsenseQA is a multiple-choice QA dataset that specifically measure commonsense reasoning (Talmor et al., 2019). This dataset is constructed based on ConceptNet (Speer et al., 2017). Specifically, a source concept is first extracted from ConceptNet, along with 3 target concepts that are connected to the source concept, i.e., a sub-graph. Crowd-workers are then asked to generate questions, using the source concept, such that only one of the target concepts can correctly answer the question. Additionally, 2 more distractor concepts are selected by crowd-workers so that each question is associated with 5 answer-options. In total, the dataset contains 12,247 questions. For CommonsenseQA, we evaluate models on the development-set only, since test-set answers are not publicly available.

2.4.2 Training details

For ease of comparison, we borrow hyperparameter settings from Pan et al. (2019); we used the BERT Whole-Word Masking Uncased model (Devlin et al., 2019) for all experiments. For DREAM experiments, we used a max sequence length of 512, batch size of 24, learning rate of $1e^{-5}$, and we trained the model for 16 epochs. For CommonsenseQA, we used a max sequence length of 60, batch-size of 32, learning rate of $1e^{-5}$, and trained for 8 epochs. For pre-training on OMCS, we used max sequence length of 35, batch size of 32, learning rate of $3e^{-5}$, and trained for 3 epochs. For pre-training on ATOMIC, the max sequence length is changed to 45, other hyperparameters remain the same, and we only use the ATOMIC training set. When using OCN on CommonsenseQA, since there is no dialogue, we compute co-attention with Q_{enc}, in place of D_{enc}, in order to keep the model structure consistent.

2.4.3 Results

DREAM results are shown in Table 2.5, and CommonsenseQA results are shown in Table 2.6. For all of our experiments, we run 3 trials with different random seeds and we report average

scores in the tables. Evaluated on DREAM, our OCN model got a significant performance boost (+3.0%), compared to BERT-large from previous work. We think the reasons are that OCN is better suited for the task and that we used the BERT Whole-Word Masking Uncased model. OCN with ConceptNet knowledge injection achieves slightly better results on the development set, while ATOMIC knowledge injection helps achieve a small improvement on the test set. However, we recognize that these improvements are very limited; to our surprise, OCN pre-trained on OMCS or ATOMIC got significantly lower performance.

As for results on CommonsenseQA, ConceptNet knowledge injection provides a significant performance boost (+2.8%), compared to the OCN baseline, suggesting that explicit links from question to answer-options help the model find the correct answer. Pre-training on OMCS also provides a small performance boost to the OCN baseline. Since both ConceptNet knowledge injection and OMCS pre-training are helpful, we combine both approaches with OCN and we are able to achieve further improvement (+4.9%). Finally, similar to the results on DREAM, OCN pre-trained on ATOMIC yields a significant performance drop.

Models	M(54)	S(15)	A+L(11)	L(228)	C+L(122)	C(14)	C+S(60)
OCN	88.9	86.7	27.3	75.9	60.7	71.4	70.0
OCN + CN injection	83.3(-5.6)	86.7(+0.0)	18.2(-9.2)	76.8(+0.9)	59.8(-0.9)	64.3(-7.1)	78.3(**+8.3**)
OCN + AT injection	88.9(+0.0)	80.0(-6.7)	27.3(+0.0)	75.9(+0.0)	66.4(**+5.7**)	71.4(+0.0)	75(**+5.0**)
OCN + OMCS pre-train	70.4(-18.5)	73.3(-13.4)	45.4(+18.1)	69.7(-6.2)	48.4(-12.3)	57.1(-14.3)	68.3(**-1.7**)
OCN + ATOMIC pre-train	66.6(-22.3)	86.7(+0.0)	18.2(-9.2)	64.0(-11.9)	51.6(-9.1)	42.9(-28.5)	70.0(**+0.0**)

Table 2.7: Accuracies for each DREAM question type: **M** means *Matching*, **S** means *Summary*, **L** means *Logic inference*, **C** means *Commonsense inference*, and **A** means *Arithmatic inference*. Numbers beside types denote the number of questions of that type.

Models	AtLoc.(596)	Cau.(194)	Cap.(109)	Ant.(92)	H.Pre.(46)	H.Sub.(39)	C.Des.(28)	Des.(27)
OCN	64.9	66.5	65.1	55.4	69.6	64.1	57.1	66.7
+CN inj,	67.4(+2.5)	70.6(+4.1)	66.1(+1.0)	60.9(+5.5)	73.9(+4.3)	66.7(+2.6)	64.3(+7.2)	77.8(+11.1)
+OMCS	68.8(+3.9)	63.9(-2.6)	62.4(-2.7)	60.9(+5.5)	71.7(+2.1)	59.0(-5.1)	64.3(+7.2)	74.1(+7.4)
+ATOMIC	62.8(-2.1)	66.0(**-0.5**)	60.6(-4.5)	52.2(-3.2)	63.0(-6.6)	56.4(-7.7)	60.7(**+3.6**)	74.1(**+7.4**)
+OMCS+CN	71.6(+6.7)	71.6(+5.1)	64.2(+0.9)	59.8(+4.4)	69.6(+0.0)	69.2(+5.1)	75.0(+17.9)	70.4(+3.7)

Table 2.8: Accuracies for each CommonsenseQA question type: **AtLoc.** means *AtLocation*, **Cau.** means Causes, **Cap.** means *CapableOf*, **Ant.** means *Antonym*, **H.Pre.** means *HasPrerequiste*, **H.Sub** means *HasSubevent*, **C.Des.** means *CausesDesire*, and **Des.** means *Desires*. Numbers beside types denote the number of questions of that type.

2.5 Error Analysis

To better understand when a model performs better or worse with knowledge integration, we analyzed model predictions. DREAM dataset provides annotations for about 1000 questions: 500 questions in the development-set and 500 in the test-set. Specifically, questions are manually classified

into 5 categories: Matching, Summary, Logic inference, Commonsense inference, and Arithmetic inference; and each question can be classified under multiple categories. We refer readers to Sun et al. (2019b) for additional category information. We extracted model predictions for these annotated questions in the test set and grouped them by type. The accuracies for each question group are shown in Table 2.7. Note that we omitted 2 categories that have less than 10 questions. For the ConceptNet and the ATOMIC knowledge-injection models, we can see that they did better on questions that involve commonsense (the last 3 columns in the table), and the performance on other types is about the same or slightly worse, compared to baseline OCN. As for models pre-trained on OMCS corpus or ATOMIC knowledge base, we already saw that these model performances drop, compared to the baseline. When we look at the performance difference in each question type, it is clear that some categories account for the performance drop more than others. For example, for both the OMCS pre-trained model and the ATOMIC pre-trained model, performance drops significantly for Matching questions, in particular. On the other hand, for questions that require both commonsense inference and summarization, both models' performances only dropped slightly or did not change. Based on these results, we infer that commonsense knowledge injection with attention is making an impact on models' weight distributions. The model is able to do better on questions that require commonsense but is losing performance on other types, suggesting a direction for future research in developing more robust (e.g., conditional) injection methods. Moreover, pre-training on knowledge bases seems to have a larger impact on models' weight distributions, resulting in inferior performance. This weight distribution shift also favors of commonsense, as we see that commonsense types are not affected as much as other types. We also conducted a similar analysis for `CommonsenseQA`. Since all questions in `CommonsenseQA` require commonsense reasoning, we classify questions based on the ConceptNet relation between the question concept and the correct answer concept. The intuition is that the model needs to capture this relation in order to answer the question. The accuracies for each question type are shown in Table 2.8. Note that we have omitted question types that have less than 25 questions. We can see that with ConceptNet relation-injection, all question types got performance boosts, for both OCN model and OCN pre-trained on OMCS, suggesting that knowledge is indeed helpful for the task. In the case of OCN pre-trained on ATOMIC, although the overall performance is much lower than OCN baseline, it is interesting to see that performance for the "Causes" type is not significantly affected. Moreover, performance for "CausesDesire" and "Desires" types actually got much better. As noted by (Sap et al., 2019a), "Causes" in ConceptNet is similar to "Effects" and "Reactions" in ATOMIC; and "CausesDesire" in ConceptNet is similar to "Wants" in ATOMIC. This result also correlates with our findings from our analysis on `DREAM`, wherein we found that models with knowledge pre-training perform better on questions that fit the knowledge domain but perform worse on others. In this case, pre-training on ATOMIC helps the model do better on questions that are similar to ATOMIC relations, even though overall performance is inferior. Finally, we noticed that questions of type "Antonym" appear to be the hardest ones. Many questions that fall into this category contain negations, and we hypothesize that the models still lack the ability to reason over negation sentences, suggesting another direction for future improvement.

2.6 Discussion and Conclusion

Based on our experimental results and error analysis, we see that external knowledge is only helpful when there is alignment between questions and knowledge-base types. Thus, it is crucial to identify the question type and apply the best-suited knowledge. In terms of knowledge-integration methods, attention-based injection seems to be the better choice for pre-trained language models such as BERT. Even when alignment between the knowledge base and dataset is sub-optimal, the performance would not degrade. On the other hand, pre-training on knowledge bases would shift the language model's weight distribution toward its own domain, greatly. If the task domain does not fit knowledge-base well, model performance is likely to drop. When the domain of the knowledge-base aligns with that of the dataset perfectly, both knowledge-integration methods bring performance boosts and a combination of them could bring further gain.

Overall we observe positive signals for building a more generalizable QA system by incorporating additional explicit knowledge graphs. However, the scope of this chapter is limited, because we have only investigated two knowledge graphs and two QA benchmarks. Moreover, we have only experimented with supervised settings. These limitations motivate us to delve deeper into the direction of hybrid knowledge fusion and pursue the model's generalizability in a broader scope.

Chapter 3

Knowledge-driven Data Construction for Zero-shot Evaluation in Commonsense Question Answering

As discussed in the previous chapter, fusing additional explicit knowledge with the language model's internal implicit knowledge improves the model's performance on commonsense QA tasks (better in-domain generalization). However, training data with human annotations are expensive/hard to collect in many real-world scenarios, hence it's more important to enhance the model's generalization in unseen tasks, e.g. unsupervised settings. Motivated by this observation, in this chapter, we propose a novel neuro-symbolic framework for zero-shot question answering across commonsense tasks. Our framework transforms various existing knowledge graphs into synthetic QA pairs, which can be used to adapt pretrained language models. Once trained, the models can be directly applied to solve different commonsense QA tasks. Our results show that adapting the language models on synthetic data populated from consolidated commonsense knowledge graphs leads to significant performance improvement across a diverse set of QA tasks focusing on different aspects of commonsense. In other words, incorporating additional explicit knowledge through carefully designed data transformation and training leads to better generalization across different domains of questions.

3.1 Introduction

Common sense is key to efficient communication in everyday situations, as it enables natural language understanding through contextual reasoning. Machine question answering (QA) benchmarks, like `SocialIQA` (Sap et al., 2019b) and `PhysicalIQA` (Bisk et al., 2020), are effective *behavioral* tests of commonsense reasoning in machines, each focusing on different capabilities. Answering a question in `SocialIQA` might require the knowledge that readers typically prefer heroes over villains in fantasy novels; whereas, in `PhysicalIQA`, the knowledge that metal stools can break windows, because windows are made of glass and metal is a more enduring material than glass.

Although such tasks had been traditionally difficult for machines, recent developments in pretrained neural language modeling have led to leaps in accuracylclosing the gap between human

and machine performance to single-digit percentage points.[1] However, due to increasing concern that large-capacity neural systems are modeling individual datasets, rather than learning how to perform logical reasoning or to utilize external knowledge effectively (Mitra et al., 2019), the focus is shifting to alternative training and evaluation strategies. In particular, *zero-shot evaluation* shows promise as an efficient measure of model generalisability across tasks (Shwartz et al., 2020; Li et al., 2020b). Here, models are trained and validated on task $\mathbf{A}$, and tested on a different task $\mathbf{B}$, without access to $\mathbf{B}$'s training data or labels. This leads state-of-the-art models from individual tasks to falter, sometimes by as much as a 50% decrease in performance (Shwartz et al., 2020). Repositories of commonsense knowledge, like `ConceptNet` (Speer et al., 2017) and `ATOMIC` (Sap et al., 2019a), can be beneficial for commonsense QA, especially when little or no training data is available.

Enriching the training data with `ConceptNet` and `ATOMIC` has been shown (Ma et al., 2019; Mitra et al., 2019) to improve accuracy on datasets *derived* from these graphs: `CommonSenseQA` (Talmor et al., 2019) and `SocialIQA`. Knowledge bases (KBs) can be used to generate question-answer pairs and distractors automatically, in order to test a model's reasoning ability (Petroni et al., 2019; Richardson and Sabharwal, 2020) or provide additional supervision (Ye et al., 2019; Yang et al., 2020). While KBs have been shown to help in a zero-shot transfer setting recently (Banerjee and Baral, 2020), no comprehensive study exists on the relation between various knowledge, its usage method, and neural models for zero-shot transfer across commonsense tasks. Moreover, while adversarial filtering techniques (Bras et al., 2020) improve the quality of a manually created question set, their impact on automatically generated questions from a variety of KBs has not been investigated yet.

In this paper, (1) we compile a set of hypotheses and design a novel neuro-symbolic framework that investigates the dependency between knowledge sources, question generation techniques, language model (LM) variants, and tasks. Our framework leverages a wide range of KBs, covering visual, social, and concept-based knowledge, to pre-train LMs for zero-shot evaluation on multiple-choice commonsense QA tasks. (2) Recognizing that the aspect of question generation is especially understudied, we expand on prior work to devise and test four distractor-sampling strategies for effective question generation. We analyze their impact on model performance across tasks, conditioned on model class and (pre-)training regime, and show that generating questions that are simultaneously fair and informative is difficult but beneficial for LM pre-training. (3) We determine which combination of knowledge graphs (KGs), data construction/training, and architectures is most effective and can utilize appropriately rich contexts across five tasks. We observe that diversifying knowledge generally improves performance, under the condition of it being aligned with the task, and that preserving the structure of the task is desired. (4) We make our code and resulting datasets available to the community to facilitate future research in this direction.[2]

[1] For example (accessed 4 August, 2020): `https://leaderboard.allenai.org/socialiqa/submissions/public`

[2] `https://github.com/Mayer123/HyKAS-CSKG`

3.2 Related Work

3.2.1 Knowledge Injection

Strong performance on standard multiple-choice QA benchmarks, like SocialIQA and PhysicalIQA, has been achieved by fine-tuning a task-specific prediction layer, placed atop pre-trained LMs, such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019b), and GPT (Radford et al., 2019). As shown by Ma et al. (2019) and Mitra et al. (2019), combining neural methods with structured background knowledge from ConceptNet, WordNet (Miller, 1995), and ATOMIC works well for commonsense datasets that have been partially derived from these resources, such as SocialIQA and CommonSenseQA. Here, the structured knowledge, formalized as lexicalized task-targeted evidence paths, is injected into an LM, either via an attention mechanism (Bauer et al., 2018) or through an auxiliary training objective (Xia et al., 2019). Graph and relation networks can also be used to score answer candidates, by informing the graph structure with data from LMs (Lin et al., 2019; Zhong et al., 2019). Finally, complete KGs can be incorporated directly in training by introducing additional modeling objectives, to teach a model about general commonsense regardless of the task at hand (Peters et al., 2019; Levine et al., 2020; Liu et al., 2020; Zhang et al., 2019; Talmor et al., 2020). This line of work resembles our approach of including background knowledge in a general, task-agnostic way; however, it still relies on the task training data and has generally not been tested in a zero-shot regime.

3.2.2 Generating Commonsense Questions and Answers

Richardson and Sabharwal (2020) use links in WordNet to generate question-answer pairs, then leverage the resulting dataset to evaluate language models. Petroni et al. (2019) prompt the skills of language models by sentences instead of questions, generated from sources like ConceptNet and SQuAD (Rajpurkar et al., 2016). Previous works have generated synthetic QA sets to complement existing training data. Ye et al. (2019) proposed an 'align-mask-select' method to generate questions using ConceptNet and Wikipedia. Kocijan et al. (2019) constructed a large set of pronoun resolution questions using Wikipedia sentences. Yang et al. (2020) generate QA pair and distractors using generative models. Regarding zero-shot evaluation, the Self-Talk model of (Shwartz et al., 2020) generates clarification prompts based on a template prefix, which are leveraged to elicit knowledge from another LM, which is used jointly with the original context and question to score each answer candidate. Given a task context, one can use COMET (Bosselut and Choi, 2019), a generative model trained on commonsense KGs, to generate background knowledge statements, and to compute scores for each answer candidate based on the context, question, and generated knowledge. Banerjee and Baral (2020) pre-train the LM with three representation learning functions which aim to complete a knowledge triple given two of its elements. These functions jointly compute the distance for each answer candidate. The ambition of this paper is to provide a comprehensive framework for such prior efforts on zero-shot QA with KGs. By covering a wider set of KGs, question generation techniques, and tasks, we can systematically investigate the effect of using different KGs, generation methods, and techniques across tasks.

3.3 Zero-Shot QA Framework

Given a natural language question Q, and n possible answers $A_1, ..., A_n$, the task is to select the most probable single answer A. We refer to the remaining $n - 1$ possible answers: $D_1, ..., D_{n-1}$ as distractors. In a zero-shot QA evaluation mode, the system has no access to the task training or development data. We assume a setup where the system is pre-trained once and then applied across different tasks in a zero-shot manner. Our zero-shot evaluation framework addresses this task by variants of pre-training an LM on an artificial QA set, created from KG data. Next, we describe its covered tasks, sources of knowledge, question generation strategies, LM techniques, and training regimes, in turn.

3.3.1 Synthetic QA Generation

We generate questions, answers, and distractor options from five KGs: `ATOMIC`, `ConceptNet`, `WordNet`, `VisualGenome` (Krishna et al., 2017), and `Wikidata` (Vrandečić and Krötzsch, 2014), found in the `Commonsense Knowledge Graph` (`CSKG`) (Ilievski et al., 2020). Notably, `ATOMIC` differs from the other KGs in two ways: 1) its relations have a different focus than those of the other sources; and 2) its node labels are longer and formalized as templates. Due to these considerations, we prepare two sets of QA sets: one based on `ATOMIC` and one based on the remaining four knowledge sources. Figure 3.1 illustrates our question generation pipeline.

Data partitions

`ATOMIC` expresses pre- and post-states for events and their participants with nine relations. Its head nodes are events, whereas the tail nodes are either events or attributes. Its nodes have two particularities: 1) irrelevant parts of the node text are replaced with blanks ('_'); and 2) references to fictional agents are indicated with special tokens (e.g., *PersonX*). We follow the `SocialIQA`'s `ATOMIC` train/dev/test splits, to ensure that the facts of the dev and test partitions are excluded in training.

Our second partition, `CWWV`, covers three other KGs in `CSKG` that express commonsense facts between concepts: `ConceptNet`, `WordNet`, and `Wikidata`. We use them jointly to generate questions, and we enrich them with additional distractors from `VisualGenome`. Treating these four sources as a single one is enabled by their `CSKG` mapping to a single set of relations, defined by `ConceptNet`. We focus on 14 semantic relations that are grounded on strong psycholinguistic and pragmatic evidence (Murphy, 2003), like `/r/Causes` and `/r/HasPrerequisite`. Since there is no pre-defined train/dev/test split for `CSKG`, we randomly sample 5% of generated questions as development set, while the other 95% are used for training, to maximize the coverage of the knowledge.

Generating questions and answers

If a triple (h, r, t) has an associated sentence, we directly employ it for question generation; otherwise, we generate a sentence in a lexicalization step, using a set of pre-defined templates.

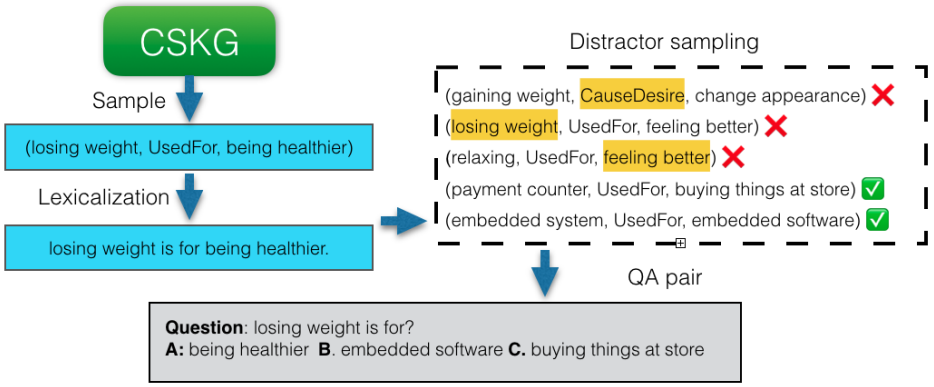

Figure 3.1: An illustration of our question generation pipeline.

Table 3.1: Generated questions from ATOMIC (top) and CWWV (bottom). (*) denotes the correct answer.

Question: Robin takes the fifth. As a result, Robin wanted to
A1: go to the cinema.
A2: withhold information. (*)
A3: hear what they think.
Question: losing weight is for
A1: being healthier. (*)
A2: embedded software.
A3: buying things in store.

Next, we generate the question Q by removing the tail of the sentence, and extract this tail as the correct answer, A.

Here, we ensure that there is no token overlap between the head and the correct answer. For ATOMIC, we: 1) compare the keyword tokens instead of all tokens, in order to avoid stopwords; and 2) the agent templates (e.g., 'PersonX') are replaced with randomly sampled gender-neutral names from a pre-defined set. For CWWV, we filter out questions where either the head or the tail are not common concepts or they are named entities. We use corpus frequency as a proxy for commonness,[3] while named entities are filtered by removing all concepts whose labels start with a capital letter.

Generating negative samples (distractors)

We seek to generate distractor options that satisfy two criteria: *informativeness* and *fairness*. Namely, a good distractor has semantic relatedness with the context (informative), while being relatively easy to discriminate from the correct answer (fair). We create the pool of distractors D

[3] https://pypi.org/project/wordfreq/ (Accessed 9 Sept. 2020)

for every sample as follows:

1. The distractor candidates are the tails of knowledge triples (h', r', t') with the same relation $r' = r$, randomly sampled from the KGs. This would ensure that the distractors can fill the same semantic role as the correct answer.
2. The head h' of the sampled triples does not have non-stop word overlap with h.
3. The distractor tail t' is not part of the correct answer set, i.e., there exist no triples, (h, r, t')

. Considering the example in Figure 3.1, the triple *(gaining weight, CausesDesire, change appearance)* will be filtered out by rule (1), *(losing weight, UsedFor, feeling better)* will be ruled out by both (2) and (3), and *(relaxing, UsedFor, feeling better)* will be ruled out by (3). Here, we replace any references to fictional ATOMIC agents in the distractors with the same names used in the question. We then randomly select two distractors (D_1, D_2) from D. We refer to this distractor pooling strategy as *random*, and propose three alternative strategies in the next Section.

Example questions with each partition are shown in Table 3.1. For ATOMIC, this procedure generates 535K QA pairs for training and 60K for development. For CWWV, the training set contains 157K and the dev set has 8K QA pairs.

3.3.2 Distractor Sampling

Existing data generation procedures are likely to introduce annotation artifacts in datasets (Zellers et al., 2019; Sakaguchi et al., 2019). Models may exploit these artifacts to achieve spuriously strong performance during training, at the expense of degradation in robustness. To generate more challenging QA pairs from KGs and to alleviate potential biases in our synthetic sets, we test two other distractor sampling strategies in addition to the *random* strategy: 1) we select distractors that are as similar as possible to the answer, while being under a certain threshold (*adv-answer*); and 2) we select distractors that are as similar as possible to the question, while being under a certain threshold (*adv-question*). Here we define the similarity of two nodes to be their proximity in the embedding space, measured by cosine similarity. The intuition is that, by generating more challenging QA pairs for the models, we could achieve better generalization across tasks. We use the RoBERTa sentence embedding model (Reimers and Gurevych, 2020) to compute embeddings for all KG nodes. For these two strategies, we set an upper bound on the similarity score to avoid unfair distractors, i.e., paraphrases of the correct answer. Based on manual observations, we set their distractor similarity upper bound to be 0.6 for CWWV and 0.4 for ATOMIC.

Sample filtering

Besides these distractor sampling strategies, we test another condition (3), where we select the distractors randomly, but only keep the questions whose distractors are sufficiently challenging at training time (*adv-filter*). The intuition is that QA pairs generated using the aforementioned methods might still be too easy for the models, thus we would like to only keep the most challenging subset to train our models. We employ the AFLite algorithm (Sakaguchi et al., 2019) for our purpose. Given a *train* and *dev* split of our synthetic QA set, we use 5% of the *train* set to finetune a RoBERTa model with a classification head (4% training, 1% validation). These 5% are discarded

from *train* after this step. We then compute the fixed embeddings for the remaining 95% of *train* and the entire *dev*, denoted as *Trn* and *Dev*. Next, we feed *Trn* and *Dev* along with their labels to the AFLite algorithm, which iteratively filters out easy examples using an ensemble of linear classifiers. Finally, we retain (101K training, 11K dev) samples for ATOMIC and (29K training, 1.5K dev) samples for CWWV subset. The details of AFLite can be found in the appendix.

3.3.3 Language Models

We consider 2 types of language models: auto-regressive language models and masked language models (MLM). Specifically, we use GPT-2 and RoBERTa to select the best answer candidate. Given a context C, a question Q, and a list of answer options ($A_1, A_2...$), we concatenate C and Q with each answer option to build input sequences ($T_1, T_2...$). We also use templates to convert a sequence T into a natural language sentence following (Shwartz et al., 2020). For example, we transform the sequence: *[C] What will X want to do next? [A_i]* into: *[C], as a result, X want to [A_i]*. The score S for the resulting sequence using an auto-regressive LM is computed as follows:

$$\mathrm{S_{LM}}\left(T\right) = -\frac{1}{n}\sum_{i=1}^{n}\log P\left(t_i \mid t_1 \ldots t_{i-1}\right) \tag{3.1}$$

where n is the number of tokens in the sequence and P is the conditional probability provided by the LM. To evaluate MLMs, we mask out one token at a time and compute its loss (Zhou et al., 2020). We repeat this process for every token in the sequence. The final MLM score is:

$$\mathrm{S_{MLM}}\left(T\right) = -\frac{1}{n}\sum_{i=1}^{n}\log P\left(t_i \mid \ldots t_{i-1}, t_{t+1} \ldots\right) \tag{3.2}$$

The predicted option is the one with the lowest score.

LM Finetuning

In the typical model architecture for finetuning LM for multiple-choice tasks, a linear layer is added on top of the LM encoder to predict the answer. The model inputs are separated by a model-specific delimiter. However, as this architecture introduces randomly initialized parameters, it may not be able to fully utilize the pre-trained weights (Tamborrino et al., 2020). Instead, we re-use the GPT-2 and RoBERTa with LM head for finetuning.

By keeping the model intact, we can reuse the same converting templates and scoring functions. To train the model, given the scores computed for each answer candidate $S_1, S_2, ...S_m$, we use the marginal ranking (MR) loss defined as:

$$\mathcal{L} = \frac{1}{m}\sum_{\substack{i=1 \\ i\neq y}}^{m}\max\left(0, \eta - S_y + S_i\right) \tag{3.3}$$

Here, η represents the margin and y is the index of the correct answer. For an MLM model, the computation cost for the scoring function scales quadratically with the input length. To make the training more efficient, we only mask out non-stop tokens in the head and tail nodes.

Table 3.2: Zero-shot evaluation results with different combinations of models and knowledge sources, across five commonsense tasks. CSKG represent the combination of ATOMIC and CWWV. We run our experiments three times with different seeds and report average accuracy with 95% confidence interval. SMLM (*) used OMCS for CSQA, ROCStories (Mostafazadeh et al., 2016) for aNLI and ATOMIC for SIQA as knowledge resources.

Model		KG	aNLI	CSQA	PIQA	SIQA	WG
Majority		-	50.8	20.9	50.5	33.6	50.4
GPT2-L		-	56.5	41.4	68.9	44.6	53.2
RoBERTa-L		-	65.5	45.0	67.6	47.3	57.5
Self-talk	2020	-	-	32.4	70.2	46.2	54.7
COMET-DynaGen	2019	ATOMIC	-	-	-	50.1	-
SMLM	2020	$\star$	65.3	38.8	-	48.5	-
GPT2-L (MR)		ATOMIC	$59.2(\pm0.3)$	$48.0(\pm0.9)$	$67.5(\pm0.7)$	$53.5(\pm0.4)$	$54.7(\pm0.6)$
GPT2-L (MR)		CWWV	$58.3(\pm0.4)$	$46.2(\pm1.0)$	$68.6(\pm0.7)$	$48.0(\pm0.7)$	$52.8(\pm0.9)$
GPT2-L (MR)		CSKG	$59.0(\pm0.5)$	$48.6(\pm1.0)$	$68.6(\pm0.9)$	$53.3(\pm0.5)$	$54.1(\pm0.5)$
RoBERTa-L (MR)		ATOMIC	$\mathbf{70.8}(\pm\mathbf{1.2})$	$64.2(\pm0.7)$	$72.1(\pm0.5)$	$63.1(\pm1.5)$	$59.6(\pm0.3)$
RoBERTa-L (MR)		CWWV	$70.0(\pm0.3)$	$\mathbf{67.9}(\pm\mathbf{0.8})$	$72.0(\pm0.7)$	$54.8(\pm1.2)$	$59.4(\pm0.5)$
RoBERTa-L (MR)		CSKG	$70.5(\pm0.2)$	$67.4(\pm0.8)$	$\mathbf{72.4}(\pm\mathbf{0.4})$	$\mathbf{63.2}(\pm\mathbf{0.7})$	$\mathbf{60.9}(\pm\mathbf{0.8})$
RoBERTa-L (supervised)		-	85.6	78.5	79.2	76.6	79.3
Human		-	91.4	88.9	94.9	86.9	94.1

Training Regimes

In order to disentangle the contribution of the KGs from the structure of the QA pairs, we consider different training methods for augmentation of language models with KGs. Specifically, we compare marginal ranking (MR) training with masked language modeling (MLM) training. For MLM, we directly concatenate the question and the correct answer in our synthetic QA set and then train RoBERTa on these sentences using the MLM objective.

3.3.4 Tasks

We select commonsense tasks based on two criteria. Firstly, we strive to cover a diverse set of tasks, both in terms of their format (question answering, pronoun resolution, natural language inference), as well as their type of knowledge (e.g., social or physical knowledge). Secondly, we prefer larger task datasets that are manually constructed. For this reason, we do not include datasets like COPA (Gordon et al., 2012), or HellaSwag (Zellers et al., 2019). We opt for the following five task datasets:

Abductive NLI (aNLI) (Bhagavatula et al., 2019) is posed as a natural language inference task. Given the beginning and the ending of a story, the task is to choose the more plausible hypothesis out of two options. The dataset consists of nearly 170k entries.

CommonsenseQA (CSQA) (Talmor et al., 2019) evaluates a broad range of common sense aspects. Each entry contains a question and 5 answer candidates. The questions are crowdsourced based on a subgraph from `ConceptNet`. The answer candidates combine `ConceptNet` nodes with additional crowdsourced distractors.
PhysicalIQA (PIQA) (Bisk et al., 2020) is a two-choice question answering dataset which focuses on physical reasoning. Given a question, the system (or human) is asked to pick the more plausible out of two possible continuations.
SocialIQA (SIQA) (Sap et al., 2019b) is a question-answering dataset which requires reasoning about social interactions. Each entry contains a context, a question, and 3 answer candidates. The context is derived from the `ATOMIC` knowledge graph, the questions are generated based on nine templates (corresponding to the relations in `ATOMIC`), and the answers are crowdsourced.
WinoGrande (WG) (Sakaguchi et al., 2019) contains 44 thousand pronoun resolution problems. Each entry consists of a context description with an emphasized pronoun, and two options are offered as its possible references.

3.4 Experimental Setup

3.4.1 Baselines

We compare our results with the following baselines. *Majority* answers each question with the most frequent option in the entire dataset. *'Vanilla' versions of the language models* are used in order to understand the impact of further tuning. Here we directly uses the LMs to score the QA pairs without any finetuning. We also show the results of other unsupervised systems that leverage KGs: *Self-talk*, *COMET-DynaGen*, and *SMLM*. To indicate the upper bound of this work, we include results of a supervised fine-tuned RoBERTa system and of human evaluation.

3.4.2 Implementation

For the LM baselines, we directly load the weights from the Transformers library (Wolf et al., 2020) and evaluate on the downstream tasks. The finetuned LMs are trained for a single epoch on our synthetic QA set. For Adv-filter, we train the models for 5 epochs to compensate for less training data. We use our synthetic dev set to select the best model. We describe other hyper-parameters used and computing infrastructure in the appendix.

3.4.3 Hypotheses

Based on individual prior findings and understanding of different components of our framework, we put forward a set of hypotheses that will be validated in our experiments:

H1 *RoBERTa would have better performance than GPT-2.* This is in line with prior findings that RoBERTa has the advantage of bi-directional context (Zhou et al., 2020).

H2 *Pre-training a language model with artificially created question-answer sets enhances zero-shot performance.* This is also supported in previous study about unsupervised QA (Li et al., 2020b)

Table 3.3: Comparison of different QA generation strategies.

RoBERTa-L	Strategy	aNLI	CSQA	PIQA	SIQA	WG
+ATOMIC	Random	**70.8**(±**1.2**)	**64.2**(±**0.7**)	72.1(±0.5)	**63.1**(±**1.5**)	59.6(±0.3)
+ATOMIC	Adv-answer	70.4(±0.8)	62.3(±0.9)	**72.6**(±**1.8**)	61.6(±0.3)	60.5(±0.5)
+ATOMIC	Adv-question	70.8(±0.6)	55.6(±0.9)	70.6(±0.8)	51.6(±0.8)	58.5(±0.3)
+ATOMIC	Adv-filter	68.6(±1.8)	46.4(±1.5)	67.9(±1.1)	51.8(±1.2)	**60.8**(±**0.6**)
+CWWV	Random	**70.0**(±**0.3**)	67.9(±0.8)	72.0(±0.7)	**54.8**(±**1.2**)	59.4(±0.5)
+CWWV	Adv-answer	69.5(±1.1)	**68.5**(±**0.8**)	**72.7**(±**0.3**)	53.8(±0.6)	**60.7**(±**0.7**)
+CWWV	Adv-question	68.3(±2.3)	60.9(±2.3)	69.6(±0.6)	47.0(±2.0)	59.0(±1.4)
+CWWV	Adv-filter	69.7(±0.7)	64.7(±2.3)	72.0(±1.3)	50.1(±1.0)	59.4(±1.4)

H3 *The impact of more knowledge depends on the alignment between KGs and the task*, partial evidence for which is provided by (Ma et al., 2019; Mitra et al., 2019).

H4 *Adding diverse knowledge (from different KGs) improves performance*. This is the initial motivation behind the creation of CSKG (Ilievski et al., 2020), but has not been investigated in detail.

H5 *When selecting negative samples for a question, it helps to use an adversarial strategy that ensures the question is not trivial for a language model.* H5 is inspired by adversarial filtering, which has not been investigated in detail for automatically-generated questions and across KGs.

H6 *Preserving the task structure when generating synthetic data leads to better accuracy.* This is implicitly assumed in prior data augmentation work (Kocijan et al., 2019).

H7 *The automatically created questions are notably easier for humans than they are for machines* - a general assumption made by commonsense task creators and typically correct for any existing, human-generated benchmark.

3.5 Results

We evaluate various combinations of: knowledge sources, question generation strategies, LMs, training regimes, and tasks. We use accuracy as a metric. All our experiments are performed in a zero-shot setting, i.e., the models do not leverage the official training data of the task. We report results on the dev sets of these tasks, as the official test sets are not publicly available. We note that, since we did not use the tasks' dev sets for hyperparameter tuning or checkpoint selection, the dev sets can be used effectively as test sets.

3.5.1 Main Results

Table 2 shows that GPT-2 and RoBERTa outperform the majority baseline by a large margin on all tasks, indicating that the LMs have already learned relevant knowledge during pretraining. Despite being a smaller model, RoBERTa outperforms GPT-2 on 4 out of 5 tasks without pretraining, and on all tasks when pretraining over different synthetic QA sets. This shows the advantage of leveraging bi-directional context, and confirms our hypothesis H1. As expected (H2), training RoBERTa on our ATOMIC or CWWV synthetic sets brings notable performance gain on all 5 tasks. We observe that models trained on ATOMIC sets have a large advantage on SIQA compare to models trained on CWWV, while CWWV brings advantage on the CSQA task. This is not surprising as these two tasks are derived from ConceptNet and ATOMIC, respectively. The difference between ATOMIC and CWWV on the remaining three tasks is relatively small. This supports our hypothesis H3: knowledge alignment is crucial for obtaining better performance.

Training on the combined question set (CSKG) is mostly able to retain the best of its both partitions. Training on CSKG leads to best performance on three out of five tasks, showing that a global commonsense resource is able to bring consistent gain across different tasks. This supports our hypothesis H4: adding more diverse knowledge is beneficial for language models. Finally, even with this knowledge, we recognize that there is still a large gap between our model's accuracy and that of the supervised RoBERTa model.

3.5.2 Comparison of QA Generation Strategies

Table 3.3 shows the results with different sampling strategies, thus addressing H5. The best performing adversarial algorithm, *Adv-answer*, yields comparable accuracy to the *random* strategy, revealing that distractors sampled with a more sophisticated strategy are not necessarily more informative for the LMs. *Adv-question* and *Adv-filter* typically lead to declines in accuracy. Considering *Adv-question*, this could be due to the similarity of the distractors to the question, which might guide the model to learn to pick the most dissimilar candidate as the correct answer, which is an artifact of our question generation and cannot be expected to work well for downstream tasks. Our manual inspection of the remaining questions preferred by *Adv-filter* indicates that many questions are unfair, as some distractors are also correct answers, which is a consequence of the incompleteness of the KGs. Adv-filter prioritizes these questions as they are "difficult" for LMs, however, training on them might teach the LM incorrect knowledge and harm downstream accuracy.

3.5.3 Comparison of Training Regimes

Table 3.4 presents results with two different training regimes. In comparison to the baseline without additional training, MLM training on ATOMIC only improves on the SIQA task, and harms on the rest. With CWWV, it brings large gain on CSQA and small improvements on SIQA and WG. At the same time, marginal ranking training on either question set consistently outperforms MLM training by a large margin, suggesting that preserving the task structure is beneficial in addition to the question content and validating H6.

Table 3.4: Comparison between MLM and MR training.

RoBERTa-L	Train	aNLI	CSQA	PIQA	SIQA	WG
baseline	-	65.5	45.0	67.6	47.3	57.5
+ ATOMIC	MLM	62.9	43.8	65.8	53.9	55.5
+ ATOMIC	MR	70.8	64.2	72.1	63.1	59.6
+ CWWV	MLM	65.3	57.3	67.2	49.3	59.4
+ CWWV	MR	70.0	67.9	72.0	54.8	59.4

3.5.4 Difficulty of the Synthetic QA Sets

Ideally, the generated question-answer pairs should be challenging for the models but easy for humans to solve (H7). Here, we probe this hypothesis by assessing the difficulty of our synthetic QA sets both by humans and 'vanilla' LMs. We evaluated both models on the dev sets of our synthetic data. For human evaluation, we randomly sample 50 questions from ATOMIC and 50 questions from CWWV. A total of five researchers were asked to first *provide the correct answer*, then *rate the question difficulty*. For the latter, the annotator chose between easy, moderate, hard, or non-sensical - as a guideline, nonsensical questions have unfair distractors and cannot be easily understood. Following this procedure, we obtained three judgments for each question.

The inter-annotator agreement on selecting the correct answer is 0.62 using Fleiss Kappa score, which is substantial agreement. The Kripendorf alpha (Krippendorff, 2004) for rating question difficulty is 0.35, which is fair agreement. The results of the baseline LMs and human performance (Table 3.5) show that the ATOMIC subset presents a harder challenge for both models, as well as for humans. Overall, the results support our hypothesis H7: the synthetic questions are relatively easy for humans to solve and much harder for models. However, the annotation pointed to several directions for improving the synthetic QA sets. A number of questions generated from ATOMIC are ungrammatical, which makes them harder to understand, while some questions from CWWV were rated as unfair. For example, all answer options for the question *A person can* are valid: (a) *cost several thousand dollars* (b) *expressing boredom* (c) *check snow level*. As discussed earlier, this is due to the incompleteness of our KGs, and the current lack of understanding on how to generate fair, yet informative, distractors.

3.6 Discussion

3.6.1 Towards a Commonsense Service

The overarching pursuit of this paper is to understand whether generating artificial QA sets with KGs improves the zero-shot QA performance of LMs. We observe a consistent leap in accuracy across tasks, LMs, knowledge sources, and question generation strategies. While these accuracies are notably below supervised LM accuracies, they might further improve by architectural improvements of the LMs, knowledge sources with wider coverage and stronger semantics, and well-tuned

Table 3.5: LM and human accuracy on our synthetic QA sets.

Model	ATOMIC	CWWV
GPT2-L	43.2	69.5
RoBERTa-L	45.9	64.5
Human	78.0	80.7

scoring functions.[4] In addition, despite its complexity and diversity, commonsense knowledge (unlike knowledge on entities and events, which changes rapidly) is largely static and evolves slowly over time, thus making the dataset-specific finetuning unnecessary in theory. A natural question arises: can we build a sufficiently reliable, general commonsense service, by pretraining an LM on a rich set of questions covering a wide spectrum of knowledge types?

3.6.2 Impact of Knowledge

In general, we observed that using knowledge from a wider set of sources is beneficial. However, on `aNLI` and `CommonSenseQA`, the best accuracy was obtained with a subset of all questions. This could be due to the kinds of knowledge covered:

1. `aNLI` focuses on expectations of agents and events, for which `ATOMIC` is directly useful, whereas the other KGs might be delusive;
2. `CommonSenseQA` mostly requires knowledge about properties of objects (e.g., function or appearance) which is the focus of `ConceptNet`, `Wikidata`, and `VisualGenome`, but not of `ATOMIC`.

This indicates a tension between H3 and H4: while more knowledge often helps, it might not be the case when the task and the knowledge are not well-aligned. Our current understanding of the dimensions of commonsense knowledge in knowledge sources and benchmarks is limited, and would benefit from further study.

3.6.3 Generating Fair and Informative Questions

Alternatively, this result may be explained by our human evaluation: not all automatically generated questions are fair and a subset has more than one correct answer, as a direct consequence of the inherent incompleteness of KGs. Besides being a disadvantage of automatic question generation, this finding points to a more substantial challenge: generating fair and informative multiple-choice questions is not yet well understood. Adversarial strategies yield more plausible candidates than random sampling, making the task less fair; yet, fully relying on random sampling would generate distractors that are trivially discernible from the correct option. Balancing between fairness and informativeness, thus, is essential for multiple-choice question generation. Our empirical evidence

[4]For example, scoring sequences of tokens by a language model might improve the performance of LMs (Tamborrino et al., 2020).

suggests that it could be achieved by a mixed approach, where part of distractors is generated randomly, and part by adopting suitable adversarial strategies.[5]

3.7 Conclusions and limitations

In this chapter, we proposed a framework for zero-shot QA by pretraining LMs on artificial data created based on KGs. We studied the interplay between five explicit knowledge sources, four data generation strategies, two LMs, two training regimes, and five tasks. Overall, our results show that combining multiple explicit knowledge sources with implicit knowledge helps the model achieve strong generalization to unseen tasks of different domains.

Despite the steady progress toward improving the generalizability of the QA models, we also identified the following limitations of this part of the thesis: 1) we have only focused on the multiple-choice commonsense QA tasks so far, which is a less realistic setting in practice. Although these tasks require the model to perform reasoning over commonsense knowledge, providing the answer candidates to the model inevitably reduces the complexity of the task. It is rare to have answer candidates being provided in most real-world QA applications. 2) We have mostly focused on a close-book setting. In both Chapter 2 and Chapter 3, we mostly improved the reader component of the QA system we built, again due to the multiple-choice task formulation. Although we tried to use explicit knowledge from different commonsense knowledge graphs to enhance reader in Chapter 2, the coverage of these knowledge graphs is still quite limited. 3) Despite the improved generalization to many unseen tasks, the models' reasoning process remains a black box. Motivated by the limitations of these two chapters, we expand the focus of our exploration and switch to more realistic settings in the following chapters, e.g. open-domain question answering.

[5]Question formulation is another challenge. Template-based questions may be trivially easy for LMs to solve, as discussed in `https://cs.nyu.edu/faculty/davise/papers/CYCQns.html`.

Chapter 4

Open Domain Question Answering with A Unified Knowledge Interface

In this chapter, we aim to address the shortcomings discussed in the previous chapter. In particular, we shift the focus to more realistic settings of QA where the model is only provided an input question, e.g. open-domain question answering (ODQA). This naturally allows us to better study the open-book setting and focus more on the retriever component of the QA system. Moreover, we expand the types of explicit knowledge sources as well as their scope, including unstructured text, structured tables and web-scale knowledge graphs. To maintain and improve the interpretability of the system, we propose a modularized framework instead of end-to-end models.

Prior works on ODQA typically equipped their QA systems with large-scale text corpora as explicit knowledge sources. Although some efforts have sought to increase knowledge coverage by incorporating structured knowledge beyond text, accessing heterogeneous knowledge sources through a unified interface remains an open question. While data-to-text generation has the potential to serve as a universal interface for data and text, its feasibility for downstream tasks remains largely unknown. In this chapter, we bridge this gap and use the data-to-text method as a means for encoding structured knowledge for ODQA. Specifically, we propose a *verbalizer-retriever-reader* framework for ODQA over data and text where verbalized tables from Wikipedia and graphs from Wikidata are used as augmented knowledge sources. We show that our **U**nified **D**ata and **T**ext **QA**, `UDT-QA`, can effectively consolidate heterogeneous and benefit from the expanded knowledge scale, leading to large gains over text-only baselines. Overall, these results show improved generalization across different knowledge types of the retriever component.

4.1 Introduction

Pretrained language models (Devlin et al., 2019; Brown et al., 2020) have been shown to store certain knowledge (linguistic or factual) implicitly in parameters (Manning et al., 2020; Petroni et al., 2019; Roberts et al., 2020), partially explaining the superior generalization abilities over downstream tasks. However, besides the well-known hallucination issue, the *implicit knowledge* learned through language modeling objective over text struggles at reflecting up-to-date knowledge from text and structured data for answering open-domain questions. To overcome this, recent work on open domain question answering (ODQA) focuses on the semi-parametric method (Karpukhin

et al., 2020; Guu et al., 2020) where the pretrained language models can leverage external *explicit knowledge* sources for reasoning. For example, in the *retriever-reader* framework (Min et al., 2021, *inter alia*), the reader produces answers by grounding on the relevant evidence from the retriever, the interface to the explicit knowledge source (Wikipedia text passages). In this work, we focus on the semi-parametric approach for ODQA going beyond textual knowledge. Specifically, we are interested in the question: *Can we develop a viable unified interface over a realistic heterogeneous knowledge source containing both data and text?*

Recent retriever-reader models (Oguz et al., 2020; Agarwal et al., 2021) have demonstrated that expanding the textual knowledge source with more structured data is beneficial. However, only knowledge base (KB) is considered in (Agarwal et al., 2021), limiting the applicability of their method to other structured data. In (Oguz et al., 2020), both tables and KB triples are simply linearized as inputs to the reader, but different retrievers are required for individual cases. Here, we propose a *verbalizer-retriever-reader* semi-parametric framework, `UDT-QA`, which provides a unification of both representation and model for ODQA over data and text. The key idea is to augment the retriever with a data-to-text verbalizer for accessing heterogeneous knowledge sources, *i.e.,* KB graphs from WikiData, tables and passages from Wikipedia.

Given its potential in providing a universal interface for data and text, data-to-text generation is increasingly popular (Gardent et al., 2017; Parikh et al., 2020; Nan et al., 2021) with various methods developed recently for converting structured knowledge into natural language (Wang et al., 2020; Ribeiro et al., 2020; Chen et al., 2020c). Nevertheless, most existing work has focused on *intrinsic evaluations* exclusively, i.e. the quality of generated text measured by metrics like BLEU (Papineni et al., 2002), leaving its usefulness on downstream tasks largely unknown. Moreover, it remains unclear whether a single data-to-text model is able to verbalize heterogeneous structured data effectively.

To bridge the gap, we develop a novel data-to-text generation paradigm for our framework. We introduce data filtering and beam selection to maximize the faithful coverage of the input information. To remedy the lack of in-domain data, we further propose an iterative training approach to augment the existing data-to-text training set with high quality outputs selected from the target domain. With this verbalizer, we convert all tables from Wikipedia (10x more than (Oguz et al., 2020)) and sub-graphs from Wikidata together with Wikipedia text passages as the knowledge source for ODQA.

We first validate our data-to-text method using intrinsic metrics on DART (Nan et al., 2021) and additional faithfulness evaluation on the target ODQA data. We show that our data-to-text approach can effectively improve the target-domain faithful metric without compromising too much on the intrinsic metrics. To further evaluate the end-to-end effectiveness, we experiment with `UDT-QA` on the ODQA task using a recent state-of-the-art (SOTA) retriever-reader pipeline, including DPR (Karpukhin et al., 2020) and UnitedQA (Cheng et al., 2021b). Consistent with previous work, our results also suggest that extra knowledge source is beneficial for ODQA. Notably, we find that the verbalized knowledge is favored by the reader compared to the raw format (linearization), especially when the structured data size is comparable to text, leading to more pronounced improvements. Overall, `UDT-QA` shows large improvements over text-only baselines and performs competitively with more complicated methods on both Natural Questions (NQ) (Kwiatkowski et al.,

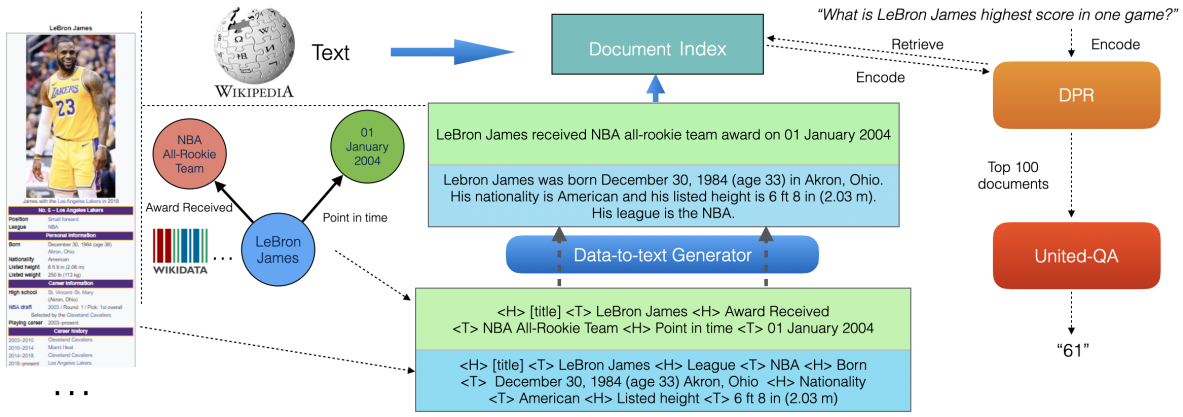

Figure 4.1: An overview of UDT-QA based on the *verbalizer-retriever-reader* pipeline.

2019) and WebQuestions (WebQ) (Berant et al., 2013). In particular, UDT-QA achieves new SOTA on NQ under the single-model open-book setting.[1]

4.2 Overview of UDT-QA

In this section, we present the overall pipeline of our UDT-QA framework for ODQA over data and text (Figure 4.1).

The major difference between our approach and the popular *retriever-reader* ODQA systems (Min et al., 2021, *inter alia*) is the use of a data-to-text verbalizer (§4.3) for converting structured data into natural language text, *i.e.,* virtual documents, as the universal knowledge source. Here, we consider two types of structured knowledge (§4.4.2) — tables and KB sub-graphs. After verbalizing the structured knowledge, a subsequent pipeline consisting of a DPR retriever and a UnitedQA-E reader is used for answer inference. Since the retriever and reader are not the main focus of this work, we only briefly describe them below.

The DPR retriever (Karpukhin et al., 2020) is a bi-encoder model consisting of a question encoder and a context encoder, which is used for data and text retrieval. Following previous work (Karpukhin et al., 2020; Oguz et al., 2020), we use the uncased BERT-base (Devlin et al., 2019) model as the encoder, where the `[CLS]` token representation is used as the document/question vector. During training, positive and negative pairs of (question, context) are used to update the model. For inference, the entire document index is encoded with context encoder and the encoded question vector is used to retrieve the top documents with highest dot-product scores.

The UnitedQA-E (Cheng et al., 2021b) is an extractive reader based on ELECTRA (Clark et al., 2020) trained with enhanced objectives (Cheng et al., 2021a, 2020) for answer inference. Here, a pair of a question and a support passage is jointly encoded into neural text representations. These representations are used to compute scores of possible answer begin and end positions, which are then used to compute probabilities over possible answer spans. Finally, the answer string

[1]Data and code available at https://github.com/Mayer123/UDT-QA

probabilities are computed based on the aggregation over all possible answer spans from the entire set of support passages.

4.3 Verbalizer: Data-to-text Generation

Here, we formally describe the data-to-text model developed in this paper, including the input format (§4.3.1) and the adaptation for ODQA (§4.3.2).

4.3.1 Input Format

Given a structured data input D, the data-to-text generator G aims to generate a natural language passage P that faithfully describes the information presented in D. In the literature, the structured data input can be in the form of a set of triples (Nan et al., 2021), a few highlighted cells from a table (Parikh et al., 2020) or a full table (Chen et al., 2020b). Correspondingly, P could a simple surface-form verbalization of D (*e.g.,* when D is a triple set) or a high-level summarization in case of a full table or a large KB graph. Since we consider (noisy) tables/KB sub-graphs of arbitrary size in this paper, directly feeding the entire input into the generator is not feasible, likely incurring significant computation challenges. Moreover, it is also desirable to maximize the information coverage of P so that most relevant information in D can be leveraged by the downstream QA retriever and reader. Based on this, we verbalize both tables and KB graphs at a fine-grained level.

In this work, we verbalize tables row by row, i.e. input each table row to G individually, where each row is a set of cells $r = \{c_i\}_{i=1}^{k}$, and k is the number of cells in the corresponding row. Most relevant to our setting, recent work (Nan et al., 2021) represents each cell in a triple. To form such triples, they manually annotate the tree ontology of column headers and then create triples using table title, headers, cell value and header relations, *e.g.,* (`[TABLECONTEXT], [title], LeBron James`), (`LeBron James, League, NBA`) where `LeBron James` is the parent cell. Although such triples with fine-grained ordering may help guide the generator, directly applying such a generator to a target domain with no ontology annotation (our case) likely results in degradation.

To overcome this, we propose to convert the triple set to pairs, *e.g.,* (`[title], LeBron James`), (`League, NBA`).

We find such conversion has little impact on the intrinsic evaluation (§4.5). After all rows are verbalized, we assemble the text outputs back to form the verbalized table.

For KB, we follow previous work (Agarwal et al., 2021) and break the KB into small sub-graphs based on subject entity. Here, each sub-graph contains one central entity and its neighbors. Although this conversion would inevitably create undesirable artifacts (*e.g.,* hurdles for multi-hop reasoning across sub-graphs), this preprocessing allows us to unify the input representations for both table and KB graphs, making it possible for a single verbalizer to convert structured knowledge into text format. Specifically, we convert all KB sub-graphs into the same format as table cell sets above, where the subject entity is treated as the title and all the edges are represented using pairs in the form of (`relation`, `object`). Then we verbalize each sub-graph with the generator G. Examples of input and output for table rows and KB sub-graphs are shown in Figure 4.1.

4.3.2 Improved Data-to-Text Model Training

A known problem in data-to-text generation is that the model tends to hallucinate or neglect information in the input (Wang et al., 2020; Agarwal et al., 2021).

Faithfulness and information coverage are especially important when we apply the verbalized output to knowledge-intensive downstream tasks like ODQA. To address this, we subsample training data `T` such that the instances are filtered out if they are likely to steer the model towards missing information. In particular, we compute ROUGE-1 (Lin, 2004) scores between the input and target of training instances and filter out those whose scores are below a certain threshold. We denote the filtered version as `T-F`. Examples of the filtered instances can be found in Table B.5, as we discuss more in section B.6, these instances may bias the model towards unwanted behaviors.

Another challenge we face is that most data-to-text training examples have succinct structured inputs. In other words, the cells in the structured input are usually single words or short phrases with corresponding short target sentences as well. In our case, a number of tables contain large cells with dozens of words. Models trained with existing data likely have a hard time verbalizing such inputs faithfully. To alleviate this domain-mismatch issue, we propose an iterative training set-up. In the first iteration, we train a generator on `T-F`. Then we apply the generator to our data. We then find high-quality verbalized outputs based on the ROUGE-1 score between the model inputs and model outputs, and sample instances with scores higher than a threshold for the next-round training. We sample instances up to the same size of `T-F`, and denote this set as `ID-T` (examples shown in Table B.5). Finally, we mix the `ID-T` with `T-F` and train a second generator for verbalization.

Following recent work (Nan et al., 2021), we use the pretrained T5-Large (Raffel et al., 2019) model as our generator. Given paired training examples consisting of a structured data input and a target sentence, we finetune the T5 model to maximize the log-likelihood of generating the corresponding target sentences. Here, we follow the same experimental setup as (Ribeiro et al., 2020).

		Intrinsic Eval						Extrinsic Eval
Training Set	**# Examples**	**BLEU**	**METEOR**	**TER**	**MoverScore**	**BERTScore**	**BLEURT**	**Ans Cov**
DART (Nan et al., 2021)	62,659	50.66	0.40	0.43	0.54	0.95	0.44	-
DART ours (`T`)	62,628	51.05	0.40	0.43	0.54	0.95	0.43	95.4
DART (`T-F`)	55,115	51.04	0.41	0.43	0.54	0.95	0.43	96.0
DART (`T-F + ID-T`)	110,230	50.59	0.41	0.44	0.54	0.95	0.43	**98.4**

Table 4.1: Intrinsic and extrinsic evaluations of verbalization approaches on DART test and `NQ-table-Q` (§4.4.1), respectively. "Ans Cov" refers to Answer coverage. All metrics are higher the better except for TER.

4.4 Experiment Setup

In this section, we describe the data used for experiments and sources of structured knowledge.

4.4.1 Datasets

In this paper, we use DART (Nan et al., 2021) to train our verbalizer (data-to-text) and two ODQA datasets, NQ and WebQ, to train and evaluate our pipeline, with the same split as in (Lee et al., 2019) provided by (Karpukhin et al., 2020). Below we provide a brief description of each dataset and refer readers to their papers for details.

DART is a data-to-text dataset containing pairs of (triple-set, sentences) collected from WebNLG (Gardent et al., 2017), E2E (Novikova et al., 2017) and crowdsourcing based on tables found in WikiSQL (Zhong et al., 2017) and WikiTableQuestions (Pasupat and Liang, 2015).

Natural Questions contains questions mined from Google search queries and the answers are annotated in Wikipedia articles by crowd workers.

WebQuestions consists of questions from Google Suggest API and the answers are annotated as entities in Freebase.

We collect **knowledge-answerable questions** from NQ and WebQ in order to evaluate our verbalizer and construct the retrieval training data. Specifically, we find questions in the original NQ training set that can be answered by a table. For each question, we search through tables in its associated HTML page to locate exact answer matches. In total, we collected 14,164 triples of (question, answer, gold table) from NQ train and dev sets as `NQ-table-Q`. On WebQ, we find questions that can be answered by KB via expanding from question entities and search for their 1-hop neighbors. If an answer entity is matched, we keep this sub-graph. In total, we collected 2,397 triples of (question, answer, sub-graph) from WebQ train and dev set as `WebQ-KB-Q`.

4.4.2 Structured Knowledge Sources

In addition to regular Wikipedia text passages, we consider two types of structured knowledge — tables from Wikipedia and KB graphs from Wikidata.

For tables from Wikipedia, we follow OTT-QA (Chen et al., 2020a) with slight modifications. Chen et al. (2020a) only consider tables in a good format, *i.e.,* tables with no empty cell, multi-column or multi-row, and restrict the tables to have at most 20 rows or columns. Instead, we remove such constraints and keep everything with the `<table>` tag, resulting in a larger and noisier table set. We denote this more realistic set of tables as `OTT-tables`.

Note Oguz et al. (2020) only consider tables from the original NQ HTMLs. In addition to the size difference, `OTT-tables` are crawled from a more recent Wikipedia dump than the NQ version. To study the impact of knowledge source size, we also process tables from the NQ HTML pages with the heuristic suggested by (Herzig et al., 2021) to de-duplicate tables and filter lengthy cells (>80 words). We denote this set of tables as `NQ-tables`. To avoid overlap, we remove tables from `OTT-tables` whose page title are in `NQ-tables` set.

In total, we have a `All-tables` set with 2.2M tables from `OTT-tables` and 210K tables from `NQ-tables`, respectively.

For KB graphs, we consider using the English Wikidata (Vrandečić and Krötzsch, 2014) as our KB due to its broad coverage and high quality, noting its predecessor Freebase is no longer maintained despite its popularity in research. In order to be comparable with recent work (Agarwal et al., 2021), we directly use their partitioned KB graphs from WikiData in our experiments, which

is denoted as WD-graphs.

4.5 Experiments: Data-to-Text

In this section, we evaluate our data-to-text model with both intrinsic and extrinsic metrics. Since intrinsic metrics are probably less correlated with the downstream performance, we use them only as a sanity check for generation quality and focus on using an extrinsic metric for selecting models.
Intrinsic Evaluation: Since our model is developed mainly on DART, we first conduct the intrinsic evaluation on the DART test set to measure the impact of our improved data-to-text methods, *i.e.,* data filtering and iterative training.

Following (Nan et al., 2021), we use the official evaluation metrics including BLEU, METEOR (Banerjee and Lavie, 2005), TER, MoverScore (Zhao et al., 2019), BERTScore (Zhang et al., 2020) and BLEURT (Sellam et al., 2020).

Table 4.1 summarizes different data-to-text models on DART test. As we can see, the resulting model trained with our data conversion (row 2) performs on par with the model using the original format (row 1). More interestingly, filtering short samples has almost no impact on the verbalizer performance (row 3). Lastly, iterative training with additional target domain data (row 4) slightly hurts on BLEU and TER and achieves similar performances on other metrics. Overall, our verbalizer with the proposed data conversion and improved training remains very effective on DART.
Extrinsic Evaluation: Since we are interested in applying verbalized knowledge for ODQA, the QA model is more likely to predict the correct answer only if the answer still exists after the verbalization. Therefore, we also evaluate each generator using a metric more related with the downstream task performance: **answer coverage**. Specifically, we compute the answer coverage as the percentage of examples that the answer present in the raw structured knowledge is still preserved in the corresponding verbalized output.

First, we compute the answer coverage of different generators discussed in the previous section on NQ-table-Q where tables are known to contain question-triggering content.

The scores are reported in the last column of Table 4.1. Due to more lengthy tables in NQ-table-Q, data filtering improves the answer coverage as expected. Moreover, model trained with our iterative training demonstrates substantial improvements in answer coverage, indicating that our approach is highly effective for converting tables into text. Examples for comparing different verbalizer outputs are shown in Table B.6 in section B.6. Later, we use this best generator to verbalize All-tables. We use beam search of size 10 and save all beams. To retain as much input information as possible, a re-ranking stage is carried out over these predictions based on the ROUGE-1 score between the model inputs and model outputs. The highest-ranked prediction is then used as the final output.

Lastly, we directly apply our best generator (DART T-F + ID-T) for verbalizing KB graphs. To evaluate the performance, we compare our model with the recent method KELM-verbalizer (Agarwal et al., 2021) using answer coverage on the set WebQ-KB-Q where KB sub-graphs are known to contain answer entities.

Although never tuned for KB graph inputs, our model achieves 99.6 on answer coverage, outperforming the KELM-verbalizer (97.8 on answer coverage) by a large margin. This suggests that

Model	NQ	WebQ
Without Structured Knowledge		
DPR (Karpukhin et al., 2020)	41.5	35.2
UnitedQA (Cheng et al., 2021b)	51.8	48.0
With Structured Knowledge		
KEALM (Agarwal et al., 2021)	41.5	43.9
UnitK-QA (Oguz et al., 2020)	54.1	**57.8**
UDT-QA *w/* Raw Single Data	54.7	51.4
UDT-QA *w/* Verbalized Single Data	**55.2**	52.0
UDT-QA *w/* Verbalized Hybrid Data	55.1	52.5

Table 4.2: End-to-end open-domain QA evaluation of UDT-QA in comparison to recent state-of-the-art models on the test sets of NQ and WebQ. Exact match scores are reported (highest scores shown in **bold**).

our data-to-text approach is highly effective for both tables and KB sub-graphs.

4.6 Experiments: QA over Data and Text

Here we present our main experiments on ODQA over data and text. For regular Wikipedia text, we use the same index containing 21M passages as in (Karpukhin et al., 2020). To augment text, two settings are considered, *i.e.,* the *single data* setting and the *hybrid data* setting.

In the single data setting for NQ, we augment the text index with tables from the All-tables set (§4.4.2). For comparison, we also experiment with the raw representations using a simple linearization of tables similar to (Oguz et al., 2020). In single data setting for WebQ, we consider combining text with KB graphs from WD-graphs in the single data setting. Different from (Oguz et al., 2020) where a separate entity-linking based retriever is used for KB, we use a single model over the text index with either linearization of raw KB graphs or our verbalized KB graphs. Hence, in our case, both text and data (tables and KB graphs) can be handled by a unified retriever-reader pipeline.

In the hybrid data setting for both NQ and WebQ, we use text, All-tables and WD-graphs for retrieval. The statistics of our document index are shown in Table B.1 in section B.1.

We create additional retriever training data from NQ-Table-Q and WebQ-KB-Q in a similar fashion as in the text-only setting, so that DPR can better handle additional knowledge. Following (Oguz et al., 2020), we also use the iterative training set-up for retriever training. More training details can be found in section B.2.

To evaluate the effectiveness of our UDT-QA for ODQA, we first include recent state-of-the-art ODQA models using text as the only knowledge source, DPR and UnitedQA. We also compare our UDT-QA with recent models using additional structured knowledge, KEALM and UnitK-QA. Following the literature, we report the exact match (EM) score for evaluation. The results are in Table 4.2.

Source	Format	R20	R100	EM
text	-	80.8	86.1	49.6
+NQ-tables	raw	85.2	90.1	51.1
+NQ-tables	V	85.5	90.2	51.2
+All-tables	raw	85.8	**90.7**	52.1
+All-tables	V	**86.0**	**90.7**	**52.5**
text	-	78.9	82.3	52.6
+WD-graphs-WebQ	raw	**83.4**	86.1	**57.1**
+WD-graphs-WebQ	V	**83.4**	85.0	55.7
+WD-graphs	raw	82.8	86.1	54.3
+WD-graphs	V	82.8	**86.7**	55.4

Table 4.3: Impact of document index size over separately trained retriever-reader models (Top for NQ and bottom for WebQ). All metrics are computed on the corresponding dev set. V stands for Verbalized here and on-wards.

As we can see, models with additional structured knowledge achieve better performance than text-only models. This indicates that both KB graphs and tables contain complementary knowledge which is either absent in text or harder to be reasoned over. For NQ, although we consider a significantly larger structured knowledge source which is likely to be more challenging, all our models substantially outperform UnitK-QA.

As for WebQ, our model achieves competitive performance, although worse than UnitK-QA. We attribute this gap to two possible reasons. First, UnitK-QA uses a separate entity-linking based retriever for KBs which might lead to higher retrieval recall. Second, since WebQ is fully based on FreeBase, using WikiData only in our models likely suffers from mismatch (Pellissier Tanon et al., 2016). Nevertheless, our verbalizer-based models achieve better performances than the corresponding raw format models on both datasets, indicating that the proposed verbalizer is highly effective for tables and KB graphs.

4.7 Analysis

In this section, we present analyses of the impact of document index size, the use of additional structured knowledge in a hot-swap setting, comparison to a recent KB-only data-to-text approach in an end-to-end fashion, and manual exam of the verbalized/raw tables for their impact on ODQA.

How does the size of the document index affect retriever and reader performance? More knowledge is likely to have better coverage of relevant information. On the other hand, larger and noisier index also increases the reasoning complexity.

To understand the impact of the increased document index size, we conduct experiments with a restricted setting where only relevant subset of knowledge to the corresponding dataset (a prior) is used for retrieval. Similar to (Oguz et al., 2020), we experiment with the combined document index of text and NQ-tables for NQ. As for WebQ, we keep documents from WD-graphs that

Source	Format	R20	R100	EM
Text-only		81.3	87.3	51.8
+NQ-tables	raw	83.9	90.3	51.7
+NQ-tables	V	84.3	90.4	52.5
+All-tables	raw	84.0	**90.6**	51.7
+All-tables	V	**84.5**	**90.6**	**52.7**

Table 4.4: Hot-swap evaluation of raw vs verbalized table using a text-only retriever-reader model on NQ test.

contain any of the question entity in WebQ to build WD-graphs-WebQ, and experiment with using text + WD-graphs-WebQ.

In addition to EM, we report R20 and R100, evaluating the retrieval accuracy of gold passages in the top-20 and top-100 documents, respectively. The results are reported in Table 4.3.

For NQ, in spite of being more challenging, we see that using All-tables yield substantial improvement in both recall and answer exact match compared to using NQ-tables.

This indicates that, with proper training, ODQA models are likely to benefit from enriched knowledge.

Although the larger raw form index brings in decent improvement (+1 EM) in terms of reader performance (+All-tables vs+NQ-tables), our verbalized knowledge is more friendly for answer reasoning leading to a more notable QA improvement (+1.3 EM). Different from NQ, we observe that on WebQ the restricted setting with WD-graphs-WebQ achieves better results. We hypothesize that this is likely due to the scale of WebQ dataset.

The small amount of WebQ training makes the retriever insufficient to handle large-scale document index. We leave the verification of this hypothesis for future work.

Does a text-only retriever-reader model benefit more from verbalized knowledge compare to raw format (hot-swap)? Since both retriever and reader are based on pretrained language models, we hypothesize that they would probably benefit more from the verbalized knowledge due to its similar style as text. This can be particularly useful for a hot-swap setting where both retriever and reader have only seen textual knowledge during training.

To verify that verbalized knowledge is more amenable, we carry out a hot-swap experiment here. Specifically, we directly use a DPR model trained on NQ text-only data for additionally indexing both NQ-tables and All-tables. Then, the inference retrieval is performed on the augmented document index for an input question, and a text-only United-QA-E reader trained on NQ is applied for answer inference afterwards. The results are summarized in Table 4.4. Similar to the previous fully fine-tuned settings, we see that additional knowledge still provides substantial improvements for text-only retriever using either raw or verbalized knowledge. However, the improvement in recall is not reflected in the later reader performance for the raw format, whereas the hot-swap answer inference performance is notably improved with verbalized knowledge. This observation further validates our hypothesis that verbalized knowledge is more beneficial, especially for reader.

Source	R20	R100	EM
KELM	78.2	85.3	51.5
WD-graphs (Ours)	**78.5**	**85.5**	**52.0**

Table 4.5: Comparison of verbalized knowledge from our verbalizer and KELM for retriever and reader on WebQ test. Dev results can be found in Table B.3 in section B.4.

Q&A	V table	Raw table
Q: star wars the clone wars season 3 episode 1 **A:** Clone Cadets	**TITLE:** List of Star Wars: The Clone Wars episodes .... the theatrical film: "the new padawan" "castle of deception" "castle of doom" "castle of salvation" is no. 3-6 in the series of star wars: the clone wars episodes. **"clone cadets" in season 3 of star wars: the clone wars is number 1 in season and number 7 in series.** "supply lines" is episode 8 in series and 3 in season of star wars: the clone wars game	| no. in series, season, no. in season, title | .... | 3-6, empty, empty, theatrical film: "the new padawan" "castle of deception" "castle of doom" "castle of salvation" | **7, 3, 1, "clone cadets"** | 8, 3, empty, "supply lines" | ....
Q: when was the last time mount ruapehu erupted **A:** 25 September 2007	**TITLE:** Mount Ruapehu .... mount ruapehu is a stratovolcano mountain with an age of 200,000 years. **the last eruption was 25 september 2007** and the volcanic arc/belt is taupo volcanic zone. mount ruapehu was first ascent in 1879 by g. beetham and j. p. maxwell. the easiest route to climb mount ruapehu is hike.	| empty, empty, empty, elevation, prominence, listing, coordinates, empty, translation, empty, empty, empty, age of rock, mountain type, volcanic arc/belt, **last eruption**, empty, first ascent, easiest route | .... 200,000 years, strato-volcano, taupo volcanic zone, **25 september 2007**, climbing, 1879 .... |
Q: who has the most yards per carry in nfl history **A:** Emmitt Smith	**TITLE:** List of National Football League career **emmitt smith** of the dallas cowboys (1990-2002) and arizona cardinals (2003-2004) **was the first player on the national football league career rushing yards leaders list**. walter payton of the chicago bears (1975-1987) ranked second	rushing yards leaders | rank, player, team(s) by season, carries, yards, average | **1, emmitt smith**, dallas cowboys (1990-2002) arizona cardinals (2003-2004), 4,409, 18,355, 4.2 | 2, walter payton, chicago bears ....
Q: which country has the smallest population in europe **A:** Vatican City	**TITLE:** List of European countries by population .... **vatican city ranks 50 on the list of european countries by population with 1,000 current population** and 0.0 % of population. the list of european countries by population has 0.0 average relative annual growth(%) and 0 average absolute annual growth. the source is official estimate and the date of last figure is 2012. The total population	| rank, country, current population, % of population, average relative annual growth(%), average absolute annual growth, estimated doubling time(years), official figure, date of last figure, regional grouping, source | 1 .... 49 .... | **50, vatican city, 1,000**, 0.0, 0.0, 0, -, 0, 2012, empty, official estimate | empty, total, ...

Table 4.6: Examples of tables/chunks retrieved by our model given the question, where the evidence is bolded. In raw table, | is the row separator and empty is the filler token used by our table parsing heuristic (to make the table in good shape)

How does the proposed verbalizer compare to recent data-to-text models? Lastly, we compare our verbalizer with the recently proposed data-to-text generator for converting KB graphs only, KELM (Agarwal et al., 2021). Since both KELM generator and our verbalizer are based on the same partitioned Wikidata, this evaluation can fully reflect their corresponding generation impacts on ODQA in an end-to-end fashion. Here, we evaluate using our verbalized WD-graphs and

the KELM corpus as additional knowledge on WebQ. In particular, we follow the same procedure to train and evaluate our retriever and reader except that we swap the `WD-graphs` with KELM corpus in data construction and retrieval. Both retriever and reader performances are reported in Table 4.5. Note that the KELM data-to-text model is customized solely for converting KB graphs and trained with a much larger dataset (about 8M training instances), whereas our verbalizer is applicable to both tables and KB graphs with smaller training data (only 110K instances).

Nevertheless, consistent with its better extrinsic performance (§4.5), our verbalizer again outperforms the KELM generator in both retrieval and reading, which provides further support for the effectiveness of our approach as a unified interface for ODQA over data and text.

4.8 Related Work

Data-to-Text Generating text from structured data has been a popular task in NLP. Many dataset have been proposed for this task such as Wikibio (Lebret et al., 2016), Rotowire (Wiseman et al., 2017), WebNLG (Gardent et al., 2017) and E2E (Novikova et al., 2017), where each dataset focuses on a particular domain. More recently, large-scale datasets that contain open-domain examples have been proposed including DART (Nan et al., 2021), TOTTO (Parikh et al., 2020), WikiTableT (Chen et al., 2021b) and GenWiki (Jin et al., 2020). On the modeling side, finetuning the pretrained models typically achieves promising performance (Ribeiro et al., 2020). Wang et al. (2020) propose customized loss functions to reduce model hallucination during generation. Muti-task learning is used to improve the model's robustness towards input variations (Hoyle et al., 2021). Chen et al. (2020c) introduce a generalized format and a pretrained model that can generate text from both table rows and knowledge graphs. Most previous work on data-to-text generation has only conducted internal evaluation, using typical generation metrics such as BLEU and ROUGE, hence the data-to-text is considered the target task. In this paper, we argue that different training strategies and evaluation metrics should be adapted when applying data-to-text models to downstream tasks, i.e. ODQA. Related to our work, Agarwal et al. (2021) convert the entire Wikidata to natural language using a finetuned T5 model (Raffel et al., 2019).

In this work, we generalize the data-to-text approach for verbalizing both tables and KB graphs in a unified fashion and study the verbalized knowledge on ODQA.

QA with Data and Text As the knowledge required to answer the questions may not be available in the textual corpus, previous studies have sought to incorporate knowledge from different sources such as tables and knowledge bases. Min et al. (2019) use Wikidata to expand seed passages found by the retriever and enhance encoded passage representations in the reader. Li et al. (2021a) propose a hybrid framework that takes both text and tables as inputs to produce answers and SQL queries. Recently, Chen et al. (2020a) developed the OTT-QA dataset containing questions that require joint reasoning over both tables and text, where the tables and text come from entire Wikipedia. There is also a line of work that studies model architectures for tables specifically or joint encoding of tables and text (Yin et al., 2020; Herzig et al., 2020; Zayats et al., 2021; Glass et al., 2021). However, their focus is not on open-domain QA tasks. Most similar to our work is (Oguz et al., 2020), where they use both tables and Wikidata/Freebase knowledge graph

along with Wikipedia text for ODQA. However, they simply linearized structured data without using any verbalizer, thus may suffer from sub-optimal input representation. Also, their tables are only mined from original NQ HTMLs, *i.e.,* a constrained setting. In contrast, we consider tables from full Wikipedia which is a much larger set. Additionally, separate retrieval models are used for tables and KB in (Oguz et al., 2020) whereas we develop a unified model over text and data.

4.9 Discussion and Conclusion

In this chapter, we demonstrated a unified *verbalizer-retriever-reader* framework, `UDT-QA`, for open-domain QA over structured data and text. We proposed a novel data-to-text paradigm that can largely improve the verbalization effectiveness for downstream knowledge-intensive applications, *i.e.,* open-domain QA, when attaining good intrinsic performances. With the verbalized knowledge, we achieved a new state-of-the-art result for NQ. Remarkably, we showed that simply augmenting the text index with the verbalized knowledge improves the performance without retraining the model.

The efforts in this chapter further improved the systems' generalization across different knowledge types. While encouraging, we noticed that the questions we studied so far are mostly simple questions, i.e. the questions can be solved with one single piece of evidence. In practice, a lot of user questions are complex and require the system to gather multiple pieces of evidence, potentially from different knowledge sources. Since our existing framework can not address such complex questions well, we proceed to propose new solutions to address this issue in the next chapter.

Chapter 5

Open Domain Question Answering via Chain-of-Reasoning over heterogeneous knowledge

As discussed in the previous chapter, there exist many complex questions that require the system to perform multiple steps of reasoning to derive the answer. Previous work on complex QA built customized models that do not work well for simple questions. Also, they rely solely on the retriever to gather all evidence in isolation, which lacks interpretability of the intermediate reasoning steps. In this chapter, we propose a novel open-domain question answering (ODQA) framework for answering single/multi-hop questions across heterogeneous knowledge sources to address the shortcomings of previous works. The key novelty of our method is the introduction of the intermediary modules into the current retriever-reader pipeline. Specifically, we decompose the overall reasoning process into four discrete operations, retriever, link, rerank and read. Each of the operations is implemented by a standalone module, and these modules are executed in a row to answer the question. With such formulation, we are able to train one single model that is able to solve both single-hop and multi-hop questions, leveraging the tables and text passages from Wikipedia. In addition to this improved generalization across question types, we also show that this modularization naturally offers better interpretability of the intermediate reasoning process.

5.1 Introduction

The task of open-domain question answering (ODQA) typically involves multi-hop reasoning, such as finding relevant evidence from knowledge sources, piecing related evidence with context together, and then producing answers based on the final supportive set. While many questions can be answered by a single piece of evidence (Joshi et al., 2017; Kwiatkowski et al., 2019), answering more complex questions are of great interest and require reasoning beyond *local* document context (Yang et al., 2018; Geva et al., 2021). The problem becomes more challenging when evidence is scattered across heterogeneous sources, *e.g.,* unstructured text and structured tables (Chen et al., 2020a), which necessitates hopping from one knowledge modality to another. Consider the question in Figure 5.2. To form the final answer, a model needs to find the entry list (a table) of the mentioned touring car race, look up the driver name with the correct rank, search for the corresponding driver information, and extract the birthplace from the free-form text.

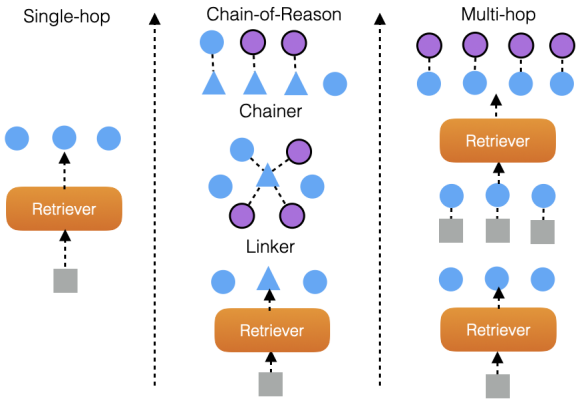

Figure 5.1: Our CORE vs. previous retriever-only methods for evidence discovery. The grey square is the question, the blue dots are 1st-hop passages, the blue triangles are 1st-hop tables, and the purple dots are 2nd-hop passages.

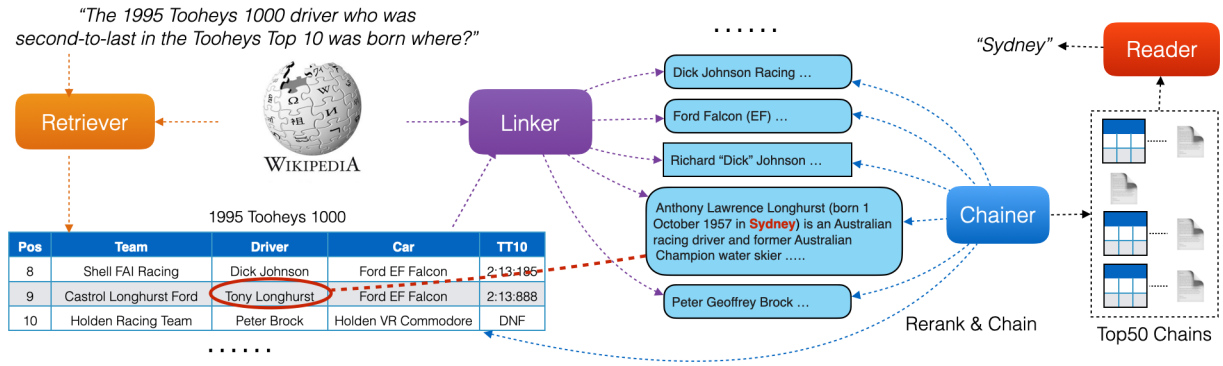

Figure 5.2: An illustration of CORE for ODQA. Given a question, the *retriever* first finds hop-1 evidence from the entire Wikipedia (orange arrows). Then the *linker* gathers relevant documents for the hop-1 evidence (purple arrows), which are treated as hop-2 evidence. Next, the *chainer* reranks all hop-1 and hop-2 evidence and splices them together into evidence chains (blue arrows). Finally, the *reader* takes in the top-50 chains and produces the answer (black arrows). The gold evidence chain is marked in red.

Existing *retriever-reader* methods (Min et al., 2021, *inter alia*) for ODQA mainly customize the retriever model for tackling individual question types, *i.e.,* exclusively relying on the retriever to gather all necessary context in a query-dependent fashion. As shown in Figure 5.1, the single-hop model (Karpukhin et al., 2020) only retrieves a list of isolated passages (blue dots). For multi-hop cases, an iterative retriever looks for a query-dependent path of passages (blue-purple dot chains) (Xiong et al., 2020b), *i.e.,* the later hop passages are retrieved using expanded queries including the original question and previous hop passages. Although those retrieval-only methods achieve promising results on their targeted cases, the customized retrievers are unable to generalize well. For example, an iterative passage retriever trained with both multi-hop and single-hop questions performs poorly over both types (Xiong et al., 2020b). For real-world applications with heterogeneous knowledge sources, it is desirable for an ODQA system to handle both cases well, and the retrieval-only methods are unlikely to succeed.

We propose a novel **C**hain **O**f **RE**asoning (CORE) framework that generalizes well on both single-hop and multi-hop question answering. The main contribution of CORE is the introduction of two intermediary modules, the linker and the chainer, that play the bridging role between the retriever and the reader, *i.e.,* piecing together related evidence with necessary context for single/multi-hop questions. These two modules work in a forward-backward fashion. In the forward pass, the linker, a novel table entity linking model (§5.3.1), links the raw evidence with its related context (*e.g.,* table-passage graphs in Figure 5.1). The chainer, a new unsupervised reranker (§5.3.2), then prunes all linked evidence using the corresponding question generation scores from a pretrained language model (Sanh et al., 2021) to form a shortlist of relevant evidence chains in a backward noisy channel fashion (*e.g.,* table-passage paths in Figure 5.1). By delegating the *hopping* operation to the intermediary, our formalization can potentially gather evidence more effectively over different question types.

To demonstrate the effectiveness of CORE, we evaluated the proposed model on two popular ODQA datasets, the multi-hop dataset OTT-QA (Chen et al., 2020a) and the single-hop dataset NQ (Kwiatkowski et al., 2019). Empirically, we show that our approach is general for both types of reasoning in ODQA. In particular, CORE substantially outperforms the previous state-of-the-art (SOTA) on OTT-QA by 14+ points on exact match scores ($45\%+$ relative gain), and it is competitive with SOTA models on NQ. Moreover, we show that one single unified model can learn to solve both tasks under our framework.

From our analysis, we also find that our evidence chains can potentially help answer single-hop questions by enriching the evidence with more supportive context. [1]

5.2 Overview of the CORE Framework

The CORE framework is designed to answer questions where the answer is a contiguous span from a table t or a passage p. Here neither t nor p is given, so they need to be retrieved from the table corpus $\mathbb{C}_t$ and the passage corpus $\mathbb{C}_p$, respectively. For single-hop questions, a single t or p may be sufficient, whereas for multi-hop questions, one or more t and p are required to find the answer.

[1] Data and code available at https://github.com/Mayer123/UDT-QA

As shown in Figure 5.2, CORE consists of a *retriever*, a *linker*, a *chainer* and a *reader*. We adopt the DPR model (Karpukhin et al., 2020) as our retriever. We only briefly describe the retriever here as it is not the main focus of our work. The DPR is a bi-encoder model that consists of a question encoder and a context encoder. For our setting, the questions and passages/tables are represented by the [CLS] embedding produced by their respective encoder, and the retrieval is done by maximum inner product search in the vector space. For a given question, we use DPR to retrieve the initial evidence set which includes tables and passages.

Given the initial evidence set (*e.g.,* the car race entry list table in Figure 5.2), our intermediary module produces a list of query-dependent evidence chains (*e.g.,* the red line linked evidence chain consisting of the car race entry list and the driver's Wikipedia page). We first propose a linker model (§5.3.1) to expand the candidate evidence set by including extra passages related to tables in the initial set (purple arrows in Figure 5.2). This step allows the model to enrich the evidence context, especially including reasoning chains needed for multi-hop questions.

Since there could be many links between a piece of evidence and others (*i.e.,* a densely connected graph), considering all links is computationally infeasible for the downstream reader. Thus, we develop a chainer model (§5.3.2) to prune the evidence graph with the corresponding question and then chain the evidence across hops together to form query-dependent paths. Here, we only keep top-K scored chains for reading so that the reader can work on a fixed computation budget.

Finally, the Fusion-in-Decoder (FiD) (Izacard and Grave, 2021), a T5-based generative model (Raffel et al., 2019), is used as the reader for generating the final answer. The model first encodes each top-K evidence chain independently along with the question. During decoding, the decoder can attend to all chains, thus fusing all the input information.

5.3 Intermediary Modules

In this part, we present the two key components of CORE for supporting multi-hop reasoning, *i.e.,* the linker for building evidence graphs and the chainer for forming query-dependent paths.

5.3.1 Linker

In this work, we mainly focus on linking an entity mention in the retrieved evidence to the corresponding Wikipedia page for building evidence graphs. This setup is related to the recent entity linking work (Wu et al., 2020). However, there are important modifications for ODQA. In particular, instead of assuming the entity mention as a prior, we consider a more realistic end-to-end scenario for ODQA: the linker model has to first *propose candidate entity mentions (spans)* for a given evidence (*e.g.,* "Tony Longhurst" in Figure 5.2), and then *links the proposed mention* to its Wikipedia page. Another major difference is that we study entity mentions in tables instead of text. As tables contain more high-level summary information than text, using tables as pivots for constructing evidence graphs can potentially help improve the recall of evidence chains for QA. In the meanwhile, this task is challenging due to the mismatch between the lexical form of the table cells and their linked passage titles. For example, the table of "NCAA Division I women's volleyball tournament" contains the cell *VCU*, which refers to *VCU Rams* instead of *Virginia Commonwealth University*. Thus simple lexical matching would not work.

In the following, we first describe the model for entity mention proposal and then present a novel entity linking model for mentions in tables. Both models are based on a pretrained language model, BERT (Devlin et al., 2019). Following previous work (Oguz et al., 2020), we flatten the table row-wise into a sequence of tokens for deriving table representations from BERT. In particular, we use $x_1, \ldots, x_N$ to denote an input sequence of length N. Typically, when using BERT, there is a prepended token `[CLS]` for all input sequences, *i.e.,* `[CLS]`$, x_1, \ldots, x_N$. Then the output is a sequence of hidden states $\mathbf{h}_{\texttt{[CLS]}}, \mathbf{h}_1, \ldots, \mathbf{h}_N \in \mathbb{R}^d$ from the last BERT layer for each input token, where d is the hidden dimension.

Entity Mention Proposal In realistic settings, the ground truth entity mention locations are not provided. Directly applying an off-the-shelf named entity recognition (NER) model can be sub-optimal as the tables are structured very differently from the text. Thus, we develop a span proposal model to label the entity mentions in the table. Specifically, we use BERT as the encoder (BERT$^{\text{m}}$) and add a linear projection to predict whether a token is part of an entity mention for all tokens in the table,

$$\mathbf{h}^{\text{m}}{}_1, \ldots, \mathbf{h}^{\text{m}}{}_N = BERT^{\text{m}}(t_1, \ldots, t_N), \tag{5.1}$$

$$\hat{\mathbf{y}} = W\mathbf{h}^{\text{m}}, \tag{5.2}$$

where $\mathbf{h}^{\text{m}} \in \mathbb{R}^{N \times d}$ and $W \in \mathbb{R}^{2 \times d}$. The model is trained with a token-level binary loss

$$-\frac{1}{N}\sum_{N}^{n=1}(y_n \log P(\hat{\mathbf{y}})_1 + (1 - y_n) \log P(\hat{\mathbf{y}})_0), \tag{5.3}$$

where y_n is the 0-1 label for the token at position n, and $P(\cdot)$ is the softmax function.

Table Entity Linking Once the candidate entity mentions are proposed, we follow Wu et al. (2020) to use a bi-encoder model for linking. Similarly, two BERT models are used to encode tables (BERT$^{\text{t}}$) and passages (BERT$^{\text{p}}$), respectively. In contrast, as there are multiple entity mentions for each table, we want to avoid repetitively inserting additional marker tokens and re-computing representations for each mention occurrence. Accordingly, we cannot simply take the `[CLS]` embeddings for linking as previous work (Wu et al., 2020). Inspired by Baldini Soares et al. (2019), to represent each entity mention, we propose a new entity representation based on the entity start and end tokens. For an entity mention with a start position i and end position j in the table, we compute our proposed entity embedding $\mathbf{e} \in \mathbb{R}^d$ for linking by

$$\mathbf{e} = (\mathbf{h}^{\text{t}}{}_i + \mathbf{h}^{\text{t}}{}_j)/2, \tag{5.4}$$

For passages, we directly take the `[CLS]` hidden state $\mathbf{p} = \mathbf{h}^{\text{p}}{}_{\texttt{[CLS]}} \in \mathbb{R}^d$ as the passage representation. Following the literature, the table entity linker is trained based on a contrastive learning objective

$$L_{\text{sim}} = -\frac{\exp(\text{sim}(\mathbf{e}, \mathbf{p}^+))}{\sum_{\mathbf{p}' \in \mathcal{P} \cup \{\mathbf{p}^+\}} \exp(\text{sim}(\mathbf{e}, \mathbf{p}'))}, \tag{5.5}$$

where $\mathbf{p}^+$ is the corresponding (positive) passage and $\mathcal{P}$ is the irrelevant set of negative passages.

Training To train our linker including entity mention proposal and table entity linking, we leverage a small set of Wikipedia tables with ground truth hyperlinks, where each linked mention and the first paragraph of the linked page constitute a positive pair (see section C.3). Similar to Wu et al. (2020), we use BM25 (Robertson and Walker, 1994b) to mine hard negative pairs for each entity

mention.

Inference During inference, we first use the entity span proposal model to label entity mentions in the tables and then run the table entity linking model to link predicted mentions to passages via maximum inner product search (Johnson et al., 2019). Here, we allow searching over all passages rather than the first paragraph of each Wikipedia page for recall purposes.

For efficient inference, the passage corpus can be pre-computed and stored as done in dense retrieval (Karpukhin et al., 2020). Then we can build links for all tables in the corpus (inducing graph-structured corpus), and directly use the graph to enrich the isolated evidence items from the retriever.

5.3.2 Chainer

Although the linker model can effectively enrich the table with all relevant passages and provide possible hopping paths for reasoning, the amount of resulting information is too overwhelming if sent directly for reading, *i.e.,* a table could contain multiple entity mentions, resulting in a densely connected evidence graph. To make it easy for the reader, we use the chainer model to prune graphs and extract the most relevant paths. Since the linker (§5.3.1) builds the evidence graphs in a query-agnostic fashion, an important function of chainer here is to incorporate the question when selecting the top-K evidence paths for answer inference.

Motivated by recent work (Sachan et al., 2022) in using pretrained generative language models for passage reranking, we also build the chainer on T0 (Sanh et al., 2021) in a zero-shot fashion, *i.e.,* no training is required. Different from their approach, we design a novel relevance scoring for query-dependent hybrid evidence path reranking rather than isolated passages.

Given a question q, we model the relevance of a question-table-passage path using the following conditional

$$P(t, p|q) = P(p|t, q)P(t|q), \tag{5.6}$$

where $t \in \mathbb{C}_{\mathbb{T}}$, $p \in \mathbb{C}_{\mathbb{P}}$. The second term $P(t|q)$ is modeled by our retriever. Given that our linker is query-agnostic (*i.e.,* only modeling $P(p|t)$), we do not have a good estimation for $P(p|t, q)$ on the right-hand side. To remedy this, we apply the Bayes rule to Equation 5.6

$$P(t, p|q) \approx P(q|t, p)P(p|t)P(t|q). \tag{5.7}$$

To estimate $P(q|t, p)$, we use the question generation likelihood as in Sachan et al. (2022).

Different from Sachan et al. (2022), here we have two conditional variables. Naively computing question generation scores on all pairs results in quadratic complexity which is very computation intensive for T0.[2] To reduce the inference cost, we further decompose $P(q|t, p)$ into two question generation scores $S_{T0}(q|p)$ and $S_{T0}(q|t)$, both based on the question generation likelihood from T0. In this way, we can reuse $S_{T0}(q|t)$ for corresponding linked passages with a linear cost. To compute $S_{T0}(q|p)$ and $S_{T0}(q|t)$, we append the instruction *"Please write a question based on this passage."*[3] to every passage/table and use the same mean log-likelihood of the question tokens conditioned on the passage/table (Sachan et al., 2022).

[2]Even the smallest T0 model has 3B parameters.

[3]Changing "passage" to "table" in the prompt does not make much difference.

As our pilot study suggests that the query-agnostic linker scores are not so informative for query-table-passage paths, we only combine the retriever score with two question generation scores from Equation 5.7 as the final chainer score for reranking evidence paths (*i.e.,* $P(p|t)$ is dropped)

$$S_R(t,q) + \alpha S_{T0}(q|t) + \beta S_{T0}(q|p), \tag{5.8}$$

where $S_R(t,q) \sim P(t|q)$, and is defined as

$$S_R(t,q) = -\log(\frac{\exp(\text{sim}(t,q))}{\sum_{t_i \in \mathcal{T}} \exp(\text{sim}(t_i,q))}), \tag{5.9}$$

$\text{sim}(\cdot,\cdot)$ is the unnormalized retriever score, $\mathcal{T}$ is the first-hop evidence set, α and β are hyper-parameters. For singleton cases (first-hop table/passage without link), we modify the last two terms of Equation 5.8 to $2\alpha S_{T0}(q|t)$ and $2\alpha S_{T0}(q|p)$ for tables and passages, respectively. This can help ensure that the chainer scores for singletons and table-passage paths are on the same scale. Then we sort all singletons and chains using the chainer score and keep the top-k results. We also use heuristics to reduce potential duplication and details are in section C.1.

5.4 Experiments

In this section, we first describe datasets and knowledge sources (§5.4.1). Then we discuss single-set and joint-set experiment setups (§5.4.2) and baselines for comparison (§5.4.3). Finally, we present experiment results on OTT-QA and NQ (§5.4.4).

5.4.1 Datasets

OTT-QA (Chen et al., 2020a) is an open-domain QA dataset that contains mostly multi-hop questions. These questions require joint reasoning over evidence from tables and text passages.
Natural Questions (NQ) (Kwiatkowski et al., 2019) contains real user queries submitted to the Google search engine and the questions are mostly single-hop and considered solvable using either a single text passage or a table. We adopt the open-domain setting proposed by Lee et al. (2019).
For OTT-QA, we adopt its official release of text passages and tables. The passages are first paragraphs from all Wikipedia pages, whereas tables are well-formed tables from full Wikipedia (*i.e.,* no infoboxes, no multi-column/multi-row tables, etc.). For NQ, we adopt the Wikipedia passage splits from Karpukhin et al. (2020) and we use the processed table sets of full Wikipedia released by Ma et al. (2022b). Statistics can be found in section C.2.

5.4.2 Experimental Settings

We train a single **linker** on the OTT-QA training set, and directly apply it for both tasks. For **chainer**, we apply the off-the-shelf T0-3B (Sanh et al., 2021) for reranking on both tasks, *i.e.,* no updates to the model. Hence both models are task-independent. For both **retriever** and **reader**, we consider both *single-set* setup where separate models are trained for individual tasks and the *joint-set* setting where a single model is trained to solve both tasks. During inference, the retriever searches over task-specific knowledge sources. For OTT-QA, since most of its questions have

	Dev		Test	
	EM	**F1**	**EM**	**F1**
HYBRIDER (Chen et al., 2020d)	10.3	13.0	9.7	12.8
FR+CBR(Chen et al., 2020a)	28.1	32.5	27.2	31.5
CARP (Zhong et al., 2022)	33.2	38.6	32.5	38.5
Oracle Link + FR+CBR	35.2	39.9	35.0	39.5
Oracle Link&table + HYBRIDER	44.1	50.8	43.0	49.8
`CORE` (single)	**49.0**	**55.7**	46.7	53.5
`CORE` (joint)	**49.0**	**55.7**	**47.3**	**54.1**

Table 5.1: End-to-end QA results on OTT-QA.

tables as the first-hop evidence, we run the retriever only on the table set in the first hop to find top-100 tables. For NQ, the retriever searches over the joint index of all text and tables and keeps the top-100 items in the first hop. After retrieval, we use the linker to expand tables in the top-100 set into graphs and use the chainer to select the top-50 chains. As the chained evidence is typically longer, unless otherwise specified, we only use the top-50 chains and set the maximum sequence length to 500 for the FiD reader. More training details are in section C.3.

5.4.3 Baselines

We briefly describe the baselines for both tasks. For OTT-QA, the HYBRIDER (Chen et al., 2020d) is a reading comprehension model for joint reasoning over tables and passages. This baseline uses BM25 to retrieve relevant tables and passages. Instead, Fusion Retriever + Cross-Block Reader (FR+CBR) (Chen et al., 2020a) first links table rows to passages using BM25 to build an index of linked documents (blocks). Then it trains a biencoder dense retriever (further enhanced by Inverse Cloze Task pretraining (Lee et al., 2019)) to find the most relevant blocks. Then they use the ETC (Ainslie et al., 2020) as the reader to process up to 4K tokens returned by the retriever. CARP (Zhong et al., 2022) employs similar retriever and reader models as FR+CBR, and it additionally extracts hybrid knowledge chains to facilitate the reader's reasoning process. All approaches are dataset-specific, hence are unlikely to handle other question types from NQ. We note that both FR+CBR and CARP's reader components adopt models specifically designed for handling long sequences. Since our overall goal is to build a unified system for both single-hop and multi-hop questions, we choose FiD based on its ability to handle different cases. We also compare against two oracle settings from Chen et al. (2020a). The first one uses gold hyperlinks in fusion retriever instead of BM25 to link table rows and passages. The second setting adopts gold hyperlinks and gold tables for the HYBRIDER reader, *i.e.,* no retrieval is involved. This setting is previously considered as an estimated upper bound for this task (Chen et al., 2020a).

On NQ, we compare with text-only baselines: DPR (Karpukhin et al., 2020) which applies a BERT-based reader to first select the best passage from top-k returned by the retriever and then extracts the answer span in it; FiD (Izacard and Grave, 2021); and UnitedQA (Cheng et al., 2021b))

	Tables	EM
DPR (Karpukhin et al., 2020)	N	41.5
FiD (Izacard and Grave, 2021)	N	51.4
UnitedQA (Cheng et al., 2021b)	N	51.8
Unik-QA (Oguz et al., 2020)	Y	54.1
UDT-QA (Ma et al., 2022b)	Y	**54.7**
CORE (single)	Y	54.6
CORE (joint)	Y	53.9

Table 5.2: End-to-end QA results on NQ test.

which is an extractive model based on ELECTRA (Clark et al., 2020) and enhanced with additional training objectives (Cheng et al., 2021a, 2020). We also compare with models that consider tables as knowledge sources: Unik-QA (Oguz et al., 2020) which augments the document index with NQ tables and uses FiD reader to generate the answer; UDT-QA (Ma et al., 2022b) which incorporates the tables from full Wikipedia and adopts UnitedQA as its reader.

5.4.4 Results

The end-to-end results on OTT-QA are shown in Table 5.1. Our CORE models substantially outperform all baselines by a large margin, illustrating the effectiveness of our proposed framework. It is worth noting that our model also outperforms two oracle settings proposed in Chen et al. (2020a). This is potentially because the capacity of their reader models is quite limited compared to ours. For FR+CBR, the model can only read up to 4K tokens, whereas in our case, the FiD reader can process up to 25K tokens. We also observe that evidence to questions in OTT-QA can be found in many different reasoning chains. In other words, tables that are not annotated as gold may still provide valuable information for reasoning. However, only one table is considered in their oracle experiment with HYBRIDER. Also, it is worth noting that the joint model outperforms the single model on OTT-QA, indicating that our framework can effectively leverage NQ data to benefit OTT-QA.

Table 5.2 summarizes the results on NQ. Similar to previous work using tables as the extra knowledge source, we also find our method to be consistently better than text-only baselines. Overall, CORE achieves competitive performance compared to SOTA models. It is also worth noting that both Unik-QA and UDT-QA use the iterative training strategy for the retriever, which leads to higher retriever performance. In particular, UDT-QA achieves 91.9 recall@100 on the NQ test set, whereas we only get 90.3. This difference in retriever recall likely explains the gap in the end-to-end QA performance. Since iterative training of the retriever is not used by baselines on OTT-QA, we leave that out in our experiments. For consistency, we only train our retriever for one round. Also, we note that the joint model performs slightly worse than the single-dataset model, which is different from the trend observed on OTT-QA. We hypothesize that this is due to the distribution difference of the evidence chains in the two datasets. Most of the top-ranked evidence are

	#Docs	Max Length	EM
CORE (single)	50	500	49.0
no QGS of hop1	50	500	45.1
no QGS of hop1	100	300	44.8
no QGS of hop1&2	100	300	38.7
no Chainer	100	300	29.1

Table 5.3: Chainer ablation on OTT-QA dev.

chained cases on OTT-QA, whereas that reduces to only 30% on NQ (the rest 70% are singleton cases). The current FiD reader may have a hard time reasoning over both singleton and chained cases simultaneously. We leave further exploration for future work.

5.5 Analysis

In this section, we conduct ablation studies and analyze different components of our framework.

5.5.1 Ablation Study

We start by ablating our chainer on the OTT-QA task using CORE (single) from Table 5.1. The results are summarized in Table 5.3. First, we experiment with **removing question generation score (QGS) for hop1 evidence** (row 2). In this case, Equation 5.8 will not have the second term and a large performance drop is observed, suggesting that it is important to apply chainer to reweight the retrieved items. Under the same setting, we try to increase the number of items and decrease the max sequence length for reading (row 3). Here we are interested in seeing the impact of reading more chains while constraining the length of individual chains. The slight performance drop indicates that potential information loss from longer chains is detrimental to the final performance.

Then we test **removing QGS entirely** (row 4). As there is no reranking at all, we simply take all linked passages for each table based on the initial retrieved order. Here, since we exhaustively include all linked passages for each table, the reader budget can be quickly filled by the top few retrieved tables, which probably makes the reader suffer from information loss with too many irrelevant items. Indeed we see another large drop in the QA performance, showing the importance of reranking. Finally, we experiment with **removing chainer** (row 5). It goes one step further by not concatenating the table rows with linked passages and only passing each piece of evidence independently. Here we are interested in whether FiD is able to fuse the information without explicit chaining. The largest performance drop is observed here, suggesting that chaining the items from different hops is vital for the reader.

We also study the effect of the number of retrieved items as input to the reader on its QA performance by varying **the number of chains sent to the reader**. The results on OTT-QA dev set are shown in Table 5.4. As we can see, when reading only the top 10 chains (much less than 4K tokens on average considered by previous work (Chen et al., 2020a; Zhong et al., 2022)), CORE

# Docs	Avg # Tokens	EM	F1
50	11,897	**49.0**	**55.7**
20	4,757	44.9	51.8
10	2,351	40.8	47.0

Table 5.4: Reader ablation with different number of documents on OTT-QA dev.

	R@20	R@100	AR@20	AR@50
Joint Retriever	**83.8**	**92.1**	31.8	37.6
+Linker&Chainer	83.5	90.8	**74.5**	**82.9**
Retrieve full index	82.4	90.6	34.6	42.6

Table 5.5: Evidence recall of the joint retriever on OTT-QA dev set, where R@K evaluates gold table chunk recall and AR@K evaluates answer recall.

still outperforms the previous SOTA method by a large margin, further validating the effectiveness of our framework. In summary, the reader performance increases with more items, and we hypothesize that the reader can be further improved when given a larger capacity. We leave this exploration for future work.

5.5.2 Impact of Linker & Chainer

We measure evidence recall scores of our joint model to better understand the impact of the linker and the chainer. On OTT-QA, since most questions require multi-step reasoning, we evaluate the retriever by both gold cell recall and answer recall. Here the gold cells refer to table cells whose gold linked passage contains the answer. We consider a table chunk to be gold if it contains at least one gold cell. For answer recall, we use top-K chains produced by the chainer. Since OTT-QA official test set is hidden, we only report dev set results in Table 5.5. As expected, the answer recall scores are extremely low without the linker and chainer, as most OTT-QA questions are multi-hop and the answers are likely to appear in the hop-2 passages. To verify if the retriever itself is able to discover the full reasoning chain, we further allow the retriever to search through the joint index of tables and passages (last row). Compared with the first row, there is some improvement. On the other hand, with the help of the linker and chainer (row 2), AR@20 and AR@50 are substantially improved, which also explains our superior QA performance. We also observe similar trends on NQ, and more results and discussion are in Appendix C.4.1.

5.5.3 Alternative Linking Strategy

Since most NQ questions only require single-hop reasoning, one alternative linking strategy for NQ is to skip the linker for single-hop ones. As the question type is typically unknown in real cases, we experiment with training a question classifier to predict whether a question requires

	# Link	# No Link	AR@20	AR@50
CORE	2,214	0	74.5	82.9
Classifier	2,133	81	73.3	81.9
CORE	8,757	0	85.7	88.1
Classifier	370	8,387	86.0	88.1

Table 5.6: Answer recall on OTT-QA (top) and NQ (bottom) dev with different linking strategies.

Q&A	Evidence Chain
Q: who plays ryders mum on home and away **A:** Lara Cox	**Table Title:** List of Home and Away characters ‖ character, actor(s), duration ‖ *ryder jackson*, lukas radovich, 2017- ‖ *Ryder Jackson* is a fictional character from the Australian television soap opera "Home and Away" ... **His mother is Quinn Jackson (Lara Cox)**, who is estranged from Alf. When ...
Q: what was blur 's first number one single in the uk **A:** Country House	**Table Title:** List of UK top-ten singles in 1994 ‖ artist, weeks, singles ‖ blur, 3, *"girls & boys"*, "parklife" ‖ *Girls & Boys* (Blur song) is a 1994 song by British rock band Blur. It was released ... "Girls & Boys" was Blur's first top 5 hit and their most successful single until **"Country House" reached number 1** the following year ...
Q: Who is the dad of the cyclist that placed directly behind Marieke van Wanroij at the 2011 Holland Hills Classic ? **A:** Hans Daams	**Table Title:** Holland Hills Classic ‖ Year, First, Second, Third ‖ 2011, Marianne Vos, Marieke van Wanroij, *Jessie Daams* ‖ *Jessie Daams* Jessie Daams (born 28 May 1990) is a Belgian racing cyclist . She competed in the 2013 UCI women 's road race in Florence . **Her father is the Dutch cyclist Hans Daams** .
Q: What other team did the Cuban player on the 2012 Charlotte Eagles team play for ? **A:** Wichita Wings	**Table Title:** 2012 Charlotte Eagles season ‖ No, Position, Player, Nation ‖ 19, Midfielder, *Miguel Ferrer*, Cuba ‖ *Miguel Ferrer* (footballer) Miguel Ferrer (born March 28 , 1987) is a Cuban footballer who **played for the Wichita Wings** in the Major Indoor Soccer League .

Table 5.7: Example evidence chains found by our CORE, where ‖ separates tables and passages. The answer evidence is **bold** and the linked entity mention in the table is *italic*. The first two are from NQ and the latter are from OTT-QA.

multi-step reasoning or not. We directly use the encoded question representation produced by our joint retriever, and train a linear classifier for this task. For simplicity, we consider all NQ questions to be single-hop and all OTT-QA questions to be multi-hop. Based on this, the classifier can achieve 95.9% accuracy on the combined dev set of NQ and OTT-QA. We then proceed to compute answer recall for both NQ and OTT-QA using the classifier to decide the linking strategy for each question. The results are shown in Table 5.6. We observe the difference in answer recall between CORE and that using the question classifier is quite small. Thus, future work can also use this strategy for handling different types of questions.

5.5.4 Linker Performance

The success of our framework depends on the linking quality, thus we also evaluated our linker model as a standalone module to better understand its performance. Here we use the 789 tables in the OTT-QA dev set with ground truth hyperlinks for evaluation. For metrics, we compute

precision, recall and F1 score for finding all the links for each table and only consider the top-1 retrieved results in all settings. We compare the cell linker model proposed by Chen et al. (2020a) as a baseline. In this model, they first train a GPT-2 (Radford et al., 2019) model on OTT-QA training set to generate queries for every cell in the table (empty if the model decides to not link a certain cell), and then use BM25 to retrieve passages. In addition, we also consider a baseline that uses ground truth table cells (cells that have links) as queries and retrieve with BM25. The results are shown in Table 5.8. Our linker achieves the best F1 score compared to the two baselines, and the advantage on recall is especially prominent. The oracle+BM25 model has the best precision because the information for whether a cell requires linking is given as a prior. However, it cannot retrieve well when the passage title does not overlap significantly with the cell text, as discussed in (§5.3.1). The GPT2 model can alleviate this issue to some extent by generating additional terms for matching, however, it still lags behind our proposed linker model.

	Precision	**Recall**	**F1**
Oracle cell + BM25	**61.7**	51.0	55.9
GPT2+BM25	59.9	58.3	59.1
Our Linker	60.3	**63.0**	**61.6**

Table 5.8: Linker variants on OTT-QA Dev set tables. In total, there are 20,064 unique passages attached to these 789 tables

5.5.5 Case Study

We manually inspect evidence chains found by our model to better understand the benefits of our intermediary modules. Examples where predicted chains contain supportive evidence are in Table 5.7.

Despite that NQ questions are usually short and single-hop, we posit that some questions can potentially benefit from the proposed chain of reasoning. As illustrated by the first two examples, though the entity mentions from the tables are not directly relevant, the table-passage chains actually contain the supportive information. Therefore, in contrast to considering the evidence in isolation, our way of constructing query-dependent table-passage chains is likely to improve the density of relevant information for single-hop questions.

For the OTT-QA chains, there is little overlap between the question and the target entity passage as expected. Thus, it is relatively difficult for the retriever to succeed in gathering all relevant information alone. On the contrary, our framework can effectively handle it with our proposed linking and chaining operations.

Notably, the evidence chains also offer greater interpretability of the model's reasoning process. By inspecting the intermediate hopping entities, we can understand how the second-hop evidence is gathered. Consequently, when the model makes mistakes, it would be much easier to pinpoint the source of errors and come up with more suitable solutions, e.g. deficiency of retriever vs linker.

5.6 Related Work

Multi-hop question answering (QA) has been studied extensively for both knowledge base (KB) setting (Yih et al., 2015; Zhang et al., 2017; Talmor and Berant, 2018) and open-domain setting (Yang et al., 2018; Feldman and El-Yaniv, 2019; Geva et al., 2021). For KB-based setting, the models are trained to parse questions into logical forms that can be executed against KB (Das et al., 2021; Yu et al., 2022) or directly select entities in KB as answers (Sun et al., 2019a). On the other hand, the open-domain setting requires the model to retrieve multiple pieces of evidence from a textual corpus and then produce the answer (Nie et al., 2019; Zhu et al., 2021). Our work falls into the latter category.

One stream of open-domain work uses multiple rounds of retrieval, where the later rounds depend on the previous ones. It is typically achieved by some form of query reformulation, *e.g.,* expanding the question with previous passages (Xiong et al., 2020b; Zhao et al., 2021) or re-writing the query using relevant evidence keywords (Qi et al., 2019, 2021). The other line of work directly leverages the gold graph structure to expand the initial set of passages for hopping (Ding et al., 2019; Asai et al., 2019; Zhang et al., 2021a), assuming the presence of oracle hyperlinks. Different from those customized text-only multi-hop methods, our approach constructs evidence graphs on-the-fly and we show that it can handle single/multi-hop questions with a unified model over heterogeneous knowledge sources. Moreover, we do not presume the existence of gold hyperlinks in the corpus, making our model more applicable in realistic settings.

There are also recent efforts in leveraging structured knowledge for ODQA. Li et al. (2021a) proposed to leverage information from both text and tables to generate answers and SQL queries. Oguz et al. (2020) studied the benefits of both tables and knowledge bases on a set of ODQA tasks. Ma et al. (2022b) introduced a unified knowledge interface that first verbalizes tables and KB subgraphs into text and then uses a single retriever-reader model to handle all knowledge sources. Similarly, we also consider heterogeneous knowledge sources for ODQA. Instead of developing task-specific models and considering the evidence in isolation, we focus on finding evidence paths across different knowledge types for single/multi-hop questions.

5.7 Discussion and Conclusion

In this chapter, we present a new framework, CORE, for ODQA over heterogeneous knowledge sources. With the novel task-agnostic intermediary module, CORE can effectively handle single/multi-hop tasks using a unified model and achieve new SOTA results on OTT-QA. Overall, CORE not only achieves better generalization across question types but enables the tracing of intermediate hoping entities, leading to two of the three target properties of a QA system we set out to achieve.

At the same time, we also identify the limitations of our work so far:

Conceptually, our proposed linker model is generic and it should be able to build edges between documents regardless of their type, e.g. passage-to-passage, passage-to-table. However, in this chapter, we focus on one instantiation of the linker model, linking tables to passages.

More importantly, the four modules in CORE are separately trained and there is no interaction among them at all, despite the similarity of the modeling architecture for some of the modules.

For example, both the retriever and linker have a bi-encoder architecture and are trained with a contrastive objective. Also, the chainer model we used incurs a very large memory footprint and computation cost even just for inference. These limitations make us wonder, can we merge some of the modules together and build a more lightweight model? In other words, can we maintain the same level of generalization and interpretability while improving the model size and computation efficiency? This question leads us to the final chapter of this thesis.

Chapter 6

Chain-of-Skills: A Configurable Model for Open-Domain Question Answering

As discussed in the previous chapter, despite the improved generalization across different question types and a more interpretable reasoning process, the CORE framework is not computationally efficient. Specifically, the modules in CORE are separately trained and stored, and there is no interaction among the modules. This naturally leads to large memory and computation overhead, especially when the number of modules in the system increases. Motivated by these limitations, in this chapter, we propose the Chain-of-Skills model to address the shortcomings of the previous system. In particular, we first identify the essential reasoning skills required in ODQA that can be reused across datasets, and train one module for each of these skills. To reduce the model size and computation, we re-formulated the training objectives of some of the skills to contrastive learning and merged all modules into a shared encoder. We further proposed a novel modularization parameterization inspired by the sparse Transformer to reduce task interference. In other words, we allow implicit knowledge sharing across modules while keeping the modeling capacity to learn specialized knowledge for each skill. Once the model is learned, it can be configured flexibly based on the target task to boost performance, i.e. one model for many tasks without retraining. We demonstrate that our proposed Chain-of-Skills (COS) achieves the state-of-the-art retrieval performance on NQ, HotpotQA and OTT-QA, while having a much smaller overall model size and reduced computation. Since the COS inherits the modularized design from previous chapters, it naturally possesses the interpretable reasoning process. Hence we are able to achieve all three target properties generalizability, interpretability, and efficiency with our final model COS.

6.1 Introduction

Gathering supportive evidence from external knowledge sources is critical for knowledge-intensive tasks, such as open-domain question answering (ODQA; Lee et al., 2019) and fact verification (Thorne et al., 2018). Since different ODQA datasets focus on different information-seeking goals, this task typically is handled by customized retrieval models (Karpukhin et al., 2020; Yang et al., 2018; Wu et al., 2020; Ma et al., 2022a). However, this dataset-specific paradigm has limited model scalability and transferability. For example, augmented training with single-hop data hurts multi-hop retrieval (Xiong et al., 2020b). Further, as new information needs constantly emerge,

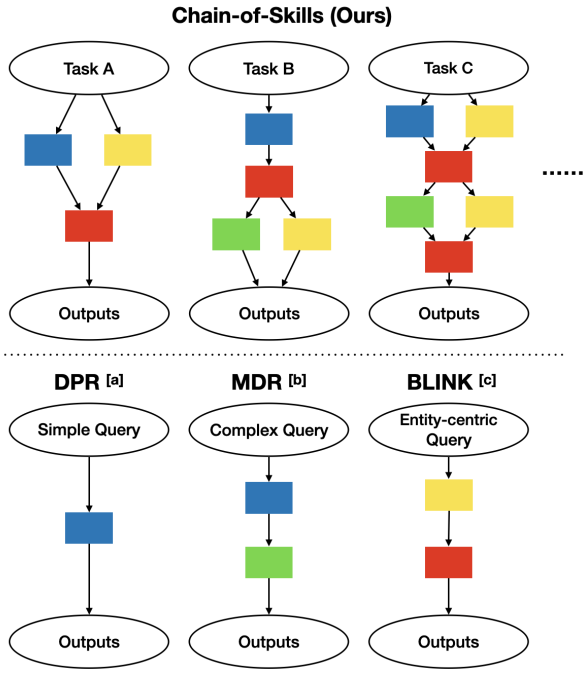

Figure 6.1: Comparison of dense retrievers in terms of considered query type and supported skill configuration [a](Karpukhin et al., 2020) [b](Xiong et al., 2020b) [c](Wu et al., 2020). Each box represents a skill (=*single retrieval*, =*expanded retrieval*, =*linking*, =*reranking*,) and the arrows represent the order of execution. In our case, we can flexibly combine and chain the skills at inference time for different tasks to achieve optimal performance.

dataset-specific models are hard to reuse.

In this work, we propose `Chain-of-Skills` (`COS`), a modular retriever based on Transformer (Vaswani et al., 2017), where each module implements a *reusable* skill that can be used for different ODQA datasets. Here, we identify a set of such retrieval reasoning skills: *single retrieval, expanded query retrieval, entity span proposal, entity linking* and *reranking* (§6.2). As shown in Figure 6.1, recent work has only explored certain skill configurations. We instead consider jointly learning all skills in a multi-task contrastive learning fashion. Besides the benefit of solving multiple ODQA datasets, our multi-skill formulation provides unexplored ways to chain skills for individual use cases. In other words, it allows flexible configuration search according to the target domain, which can potentially lead to better retrieval performance (§6.4).

For multi-task learning, one popular approach is to use a shared text encoder (Liu et al., 2019a), *i.e.,* sharing representations from Transformer and only learning extra task-specific headers atop. However, this method suffers from undesirable task interference, *i.e.,* negative transfer among retrieval skills. To address this, we propose a new modularization parameterization inspired by the recent mixture-of-expert in sparse Transformer (Fedus et al., 2021), *i.e.,* mixing specialized and shared representations. Based on recent analyses on Transformer (Meng et al., 2022), we design an attention-based alternative that is more effective in mitigating task interference (§6.5). Further, we develop a multi-task pretraining using *self-supervision* on Wikipedia so that the pretrained `COS`

can be directly used for retrieval without dataset-specific supervision.

To validate the effectiveness of COS, we consider zero-shot and fine-tuning evaluations with regard to the model in-domain and cross-dataset generalization. Six representative ODQA datasets are used: Natural Questions (NQ; Kwiatkowski et al., 2019), WebQuestions (WebQ; Berant et al., 2013), SQuAD (Rajpurkar et al., 2016), EntityQuestions (Sciavolino et al., 2021), HotpotQA (Yang et al., 2018) and OTT-QA (Chen et al., 2020a), where the last two are multi-hop datasets. Experiments show that our multi-task pretrained retriever achieves superior *zero-shot* performance compared to recent state-of-the-art (SOTA) *self-supervised* dense retrievers and BM25 (Robertson and Zaragoza, 2009). When fine-tuned using multiple datasets jointly, COS can further benefit from high-quality supervision effectively, leading to new SOTA retrieval results across the board. Further analyses show the benefits of our modularization parameterization for multi-task pretraining and fine-tuning, as well as flexible skill configuration via Chain-of-Skills inference.[1]

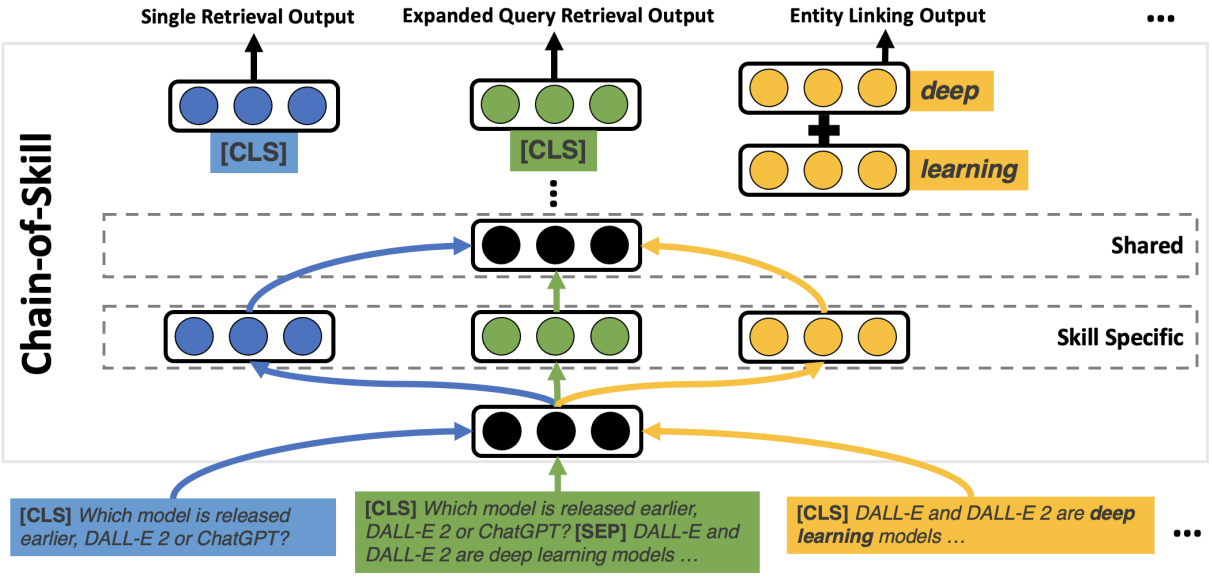

Figure 6.2: Chain-of-Skills (COS) model architecture with three different query types. The left blue box indicates the single retrieval query input. The middle green box is the expanded query retrieval input based on the single retrieval results. The right orange case is the entity-centric query with "deep learning" as the targeted entity.

6.2 Background

We consider five retrieval reasoning skills: *single retrieval*, *expanded query retrieval*, *entity linking*, *entity span proposal* and *reranking*. Conventionally, each dataset provides annotations on a different combination of skills (see Table D.1). Hence, we can potentially obtain training signals for individual skills from multiple datasets. Below we provide some background for these skills.

[1]Data and code available at https://github.com/Mayer123/UDT-QA

Single Retrieval Many ODQA datasets (*e.g.,* NQ; Kwiatkowski et al., 2019) concern simple/single-hop queries. Using the original question as input (Figure 6.2 bottom-left), single-retrieval gathers isolated supportive passages/tables from target sources in one shot (Karpukhin et al., 2020).
Expanded Query Retrieval To answer complex multi-hop questions , it typically requires evidence chains of two or more separate passages (*e.g.,* HotpotQA; Yang et al., 2018) or tables (*e.g.,* OTT-QA; Chen et al., 2020a). Thus, follow-up rounds of retrieval are necessary after the initial single retrieval. The expanded query retrieval (Xiong et al., 2020b) takes an expanded query as input, where the question is expanded with the previous-hop evidence (Figure 6.2 bottom-center). The iterative retrieval process generally shares the same target source.
Entity Span Proposal Since many questions concern entities, detecting those salient spans in the question or retrieved evidence is useful. The task is related to named entity recognition (NER), except requiring only binary predictions, *i.e.,* whether a span corresponds to an entity. It is a prerequisite for generating entity-centric queries (context with target entities highlighted; Figure 6.2 bottom-right) where targeted entity information can be gathered via downstream entity linking.
Entity Linking Mapping detected entities to the correct entries in a database is crucial for analyzing factoid questions. Following Wu et al. (2020), we consider an entity-retrieval approach, *i.e.,* using the entity-centric query for retrieving its corresponding Wikipedia entity description.
Rereanking Previous work often uses a reranker to improve the evidence recall in the top-ranked candidates. Typically, the question with a complete evidence chain is used together for reranking.

6.3 Approach

In this work, we consider a holistic approach to gathering supportive evidence for ODQA, *i.e.,* the evidence set contains both singular tables/passages (from single retrieval) and connected evidence chains (via expanded query retrieval/entity linking). As shown in Figure 6.2, `COS` supports flexible skill configurations, *e.g.,* expanded query retriever and the entity linker can build upon the single-retrieval results. As all retrieval skill tasks are based on contrastive learning, we start with the basics for our multi-task formulation. We then introduce our modularization parameterization for reducing task interference. Lastly, we discuss ways to use self-supervision for pretraining and inference strategies.

6.3.1 Reasoning Skill Modules

All reasoning skills use text encoders based on Transformer (Vaswani et al., 2017). Particularly, only BERT-base (Devlin et al., 2019) is considered without further specification. Text inputs are prepended with a special token `[CLS]` and different segments are separated by the special token `[SEP]`. The bi-encoder architecture (Karpukhin et al., 2020) is used for single retrieval, expanded query retrieval, and entity linking. We use dot product for $\text{sim}(\cdot, \cdot)$.
Retrieval As single retrieval and expanded query retrieval only differ in their query inputs, these two skills are discussed together here. Specifically, both skills involve examples of a question Q, a positive document P^+. Two text encoders are used, *i.e.,* a query encoder for questions and a context passage encoder for documents. For the expanded query case (Figure 6.2 bottom-center), we concatenate Q with the previous-hop evidence as done in Xiong et al. (2020b), *i.e.,* `[CLS]` Q

[SEP] P_1^+ [SEP]. Following the literature, [CLS] vectors from both encoders are used to represent the questions and documents respectively. The training objective is

$$L_{\text{ret}} = -\frac{\exp(\text{sim}(\mathbf{q}, \mathbf{p}^+))}{\sum_{\mathbf{p}' \in \mathcal{P} \cup \{\mathbf{p}^+\}} \exp(\text{sim}(\mathbf{q}, \mathbf{p}'))}, \tag{6.1}$$

where $\mathbf{q}, \mathbf{p}$ are the query and document vectors respectively and $\mathcal{P}$ is the set of negative documents.

Entity Span Proposal To achieve a multi-task formulation, we model entity span proposal based on recent contrastive NER work (Zhang et al., 2022b). Specifically, for an input sequence with N tokens, $x_1, \ldots, x_N$, we encode it with a text encoder to a sequence of vectors $\mathbf{h}_1^m, \ldots, \mathbf{h}_N^m \in \mathbb{R}^d$. We then build the span representations using the span start and end token vectors, $\mathbf{m}_{(i,j)} = \tanh((\mathbf{h}_i^m \oplus \mathbf{h}_j^m)W^a)$, where i and j are the start and end positions respectively, $\oplus$ denotes concatenation, tanh is the activation function, and $W^a \in \mathbb{R}^{2d \times d}$ are learnable weights. For negative instances, we randomly sample spans within the maximum length of 10 from the same input which do not correspond to any entity. Then we use a learned anchor vector $\mathbf{s} \in \mathbb{R}^d$ for contrastive learning, *i.e.,* pushing it close to the entity spans and away from negative spans.

$$L_{\text{pos}} = -\frac{\exp(\text{sim}(\mathbf{s}, \mathbf{m}^+))}{\sum_{\mathbf{m}' \in \mathcal{M} \cup \{\mathbf{m}^+\}} \exp(\text{sim}(\mathbf{s}, \mathbf{m}'))}, \tag{6.2}$$

where $\mathcal{M}$ is the negative span set which always contains a special span corresponding to [CLS], $\mathbf{m}^{\texttt{[CLS]}} = \mathbf{h}_0^m$. However, the above objective alone is not able to determine the prediction of entity spans from null cases at test time. To address this, we further train the model with an extra objective to learn a dynamic threshold using $\mathbf{m}^{\texttt{[CLS]}}$

$$L_{\text{cls}} = -\frac{\exp(\text{sim}(\mathbf{s}, \mathbf{m}^{\texttt{[CLS]}}))}{\sum_{\mathbf{m}' \in \mathcal{M}} \exp(\text{sim}(\mathbf{s}, \mathbf{m}'))}. \tag{6.3}$$

The overall entity span proposal loss is computed as $L_{\text{span}} = (L_{\text{pos}} + L_{\text{cls}})/2$. Thus, spans with scores higher than the threshold are predicted as positive.

Entity Linking Unlike Wu et al. (2020) where entity markers are inserted to the entity mention context (the entity mention with surrounding context), we use the raw input sequence as in the entity span proposal task. For the entity mention context, we pass the input tokens $x_1, \ldots, x_N$ through the entity query encoder to get $\mathbf{h}_1^e, \ldots, \mathbf{h}_N^e \in \mathbb{R}^d$. Then we compute the entity vector based on its start position i and end position j, *i.e.,* $\mathbf{e} = (\mathbf{h}_i^e + \mathbf{h}_j^e)/2$. For entity descriptions, we encode them with the entity description encoder and use the [CLS] vector $\mathbf{p}_e$ as representations. The model is trained to match the entity vector with its entity description vector

$$L_{\text{link}} = -\frac{\exp(\text{sim}(\mathbf{e}, \mathbf{p}_e^+))}{\sum_{\mathbf{p}' \in \mathcal{P}_e \cup \{\mathbf{p}_e^+\}} \exp(\text{sim}(\mathbf{e}, \mathbf{p}'))}, \tag{6.4}$$

where $\mathbf{p}_e^+$ is the linked description vector and $\mathcal{P}_e$ is the negative entity description set.

Reranking Given a question Q and a passage P, we concatenate them as done in expanded query retrieval format [CLS] Q [SEP] P [SEP], and encode it using another text encoder. We use the pair consisting of the [CLS] vector $\mathbf{h}^r_{\texttt{[CLS]}}$ and the first [SEP] vector $\mathbf{h}^r_{\texttt{[SEP]}}$ from the output for

reranking. The model is trained using the loss

$$L_{\text{rank}} = -\frac{\exp(\text{sim}(\mathbf{h}^{r+}_{\texttt{[CLS]}}, \mathbf{h}^{r+}_{\texttt{[SEP]}}))}{\sum_{\mathbf{p}^{r\prime} \in \mathcal{P}_r \cup \{\mathbf{p}^{r+}\}} \exp(\text{sim}(\mathbf{h}^{r\prime}_{\texttt{[CLS]}}, \mathbf{h}^{r\prime}_{\texttt{[SEP]}}))}, \tag{6.5}$$

where $\mathcal{P}_r$ is the set of negative passages concatenated with the same question. Intuitively, our formulation encourages $\mathbf{h}^{r}_{\texttt{[CLS]}}$ to capture more information about the question and $\mathbf{h}^{r}_{\texttt{[SEP]}}$ to focus more on the evidence. The positive pair where the evidence is supportive likely has higher similarity than the negative ones. Our formulation thus spares the need for an extra task-specific header. As the model only learns to rerank single passages, we compute the score for each passage separately for multi-hop cases.

6.3.2 Modular Skill Specialization

Implementing all aforementioned modules using separate models is apparently inefficient. As recent work finds that parameter sharing improves the bi-encoder retriever (Xiong et al., 2020b), we thus focus on a multi-task learning approach.

One popular choice is to share the text encoder's parameter of all modules (Liu et al., 2019a). However, this approach suffers from task interference, resulting in degraded performance compared with the skill-specific model (§6.5.1). We attribute the cause to the competition for the model capacity, *i.e.,* conflicting signals from different skills require attention to individual syntactic/semantic patterns. For example, the text encoder for entity-centric queries likely focuses on the local context around the entity while the expanded query one tends to represent the latent information based on the relation between the query and previous hop evidence.

Motivated by recent modular approaches for sparse Transformer LM (Fedus et al., 2021), we propose to mitigate the task interference by mixing *skill-specific Transformer blocks* with shared ones. A typical Transformer encoder is built with a stack of regular Transformer blocks, each consisting of a multi-head self-attention (MHA) sub-layer and a feed-forward network (FFN) sub-layer, with residual connections (He et al., 2015) and layer-normalization (Ba et al., 2016) applied to both sub-layers. The shared Transformer block is identical to a regular Transformer block, *i.e.,* all skill inputs are passed through the same MHA and FFN functions.

As shown in Figure 6.2, for skill-specific Transformer blocks, we select a specialized sub-layer from a pool of I parallel sub-layers based on the input, *i.e.,* different skill inputs are processed independently. One option is to specialize the FFN expert sub-layer for individual skills, which is widely used by recent mixture-of-expert models (Fedus et al., 2021; Cheng et al., 2022a). As the FFN sub-layer is found to be important for factual associations (Meng et al., 2022), we hypothesize that using the popular FFN expert is sub-optimal. Since most reasoning skills require similar world knowledge, specializing FFN sub-layers likely hinders knowledge sharing. Instead, different skills typically require the model to attend to distinct input parts. Thus, we investigate a more parameter-efficient alternative, *i.e.,* MHA specialization. In our experiments, we find it to be more effective in reducing task interference (§6.5.1).

Expert Configuration Regarding the modularization, a naive setup is to route various task inputs to their dedicated sub-layers (experts), *i.e.,* two experts for each bi-encoder task (single retrieval, expanded query retrieval and entity linking) and one expert for each cross-encoder task (entity

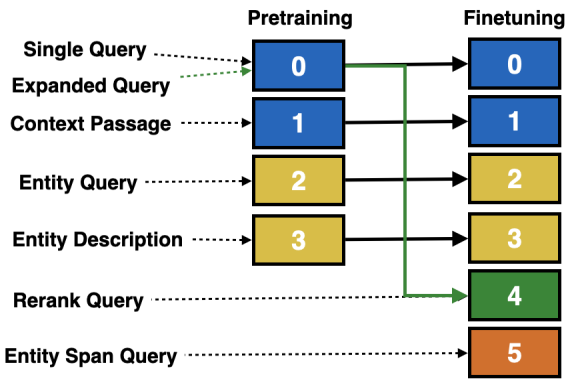

Figure 6.3: Expert configuration for COS at pretraining and fine-tuning. Each numbered box is a skill-specific expert. The lines denote input routing where solid ones also indicate weight initialization mappings. Green lines highlight the expanded query routing which is different for pretraining and fine-tuning.

span proposal and reranking), leading to eight experts in total. To save computation, we make the following adjustments. Given that single and expanded query retrievers share the same set of target passages, we merge the context expert for both cases. Due to data sparsity, we find that routing the expanded queries and reranker inputs which are very similar to separate experts is problematic (§6.5.1). Thus, we merge the expert for expanded queries and reranker inputs. During self-supervised pretraining with three bi-encoder tasks, we further share the expert for single and expanded queries for efficiency. The overall expert configuration is shown in Figure 6.3.

Multi-task Self-supervision Inspired by the recent success of Izacard et al. (2021), we also use *self-supervision* on Wikipedia for pretraining. Here, we only consider pretraining for bi-encoder skills (*i.e.,* single retrieval, expanded query retrieval, and entity linking) where abundant self-supervision is available. Unlike prior work focusing only on single-type pretraining, we consider a multi-task setting using individual pages and the hyperlink relations among them. Specifically, we follow Izacard et al. (2021) and Wu et al. (2020) to construct examples for single retrieval and entity linking, respectively. For single retrieval, a pair of randomly cropped views of a passage is used as a positive example. For entity linking, a short text snippet with a hyperlinked entity (entity mention context) is used as the query, and the first paragraph of its linked Wikipedia page is treated as the target (entity description). For a given page, we construct an expanded query using a randomly-sampled short text snippet with its first paragraph, and use one first paragraph from linked pages as the target.

6.3.3 Inference

During inference, different skills can be flexibly combined to boost retrieval accuracy. Those studied configurations are illustrated in Figure 6.1. To consolidate the evidence set obtained by different skills, we first align the linking scores based on the same step retrieval scores (single or expanded query retrieval) for sorting. Documents returned by multiple skills are considered more relevant and thus promoted in ranking. More details with running examples are provided in

	NQ		WebQ		EntityQuestions		HotpotQA		Avg	
	Top-20	**Top-100**	**Top-20**	**Top-100**	**Top-20**	**Top-100**	**Top-20**	**Top-100**	**Top-20**	**Top-100**
BM25	62.9	78.3	62.4	75.5	**70.8**	**79.2**	37.5	50.5	58.4	70.9
Contriever (Izacard et al., 2021)	67.8	**82.1**	65.4	79.8	61.8	74.2	48.7	64.5	60.9	75.2
Spider (Ram et al., 2022)	**68.3**	81.2	65.9	79.7	65.1	76.4	35.3	48.6	58.7	71.5
COS (pretrain-only)	68.0	81.8	**66.7**	**80.3**	70.7	79.1	**77.9**	**87.9**	**70.8**	**82.3**

Table 6.1: Zero-shot top-k accuracy on test sets for NQ, WebQ and EntityQuestions, and dev set for HotpotQA.

section D.1.

6.4 Experiments

6.4.1 Datasets

We consider six popular datasets for evaluation, all focused on Wikipedia, with four single-hop data, NQ (Kwiatkowski et al., 2019), WebQ (Berant et al., 2013), SQuAD (Rajpurkar et al., 2016) and EntityQuestions (Sciavolino et al., 2021); two multi-hop data, HotpotQA (Yang et al., 2018) and OTT-QA (Chen et al., 2020a). Dataset-specific corpora are used for multi-hop datasets, because HotpotQA requires retrieval hopping between text passages while table-passage hopping is demanded by OTT-QA. For single-hop data, we use the Wikipedia corpus from Karpukhin et al. (2020). More detailed (pretraining/fine-tuning) data statistics and experimental settings are in section D.2.

6.4.2 Evaluation Settings

We evaluate our model in three scenarios.

Zero-shot Evaluation Similar to recent self-supervised dense retrievers on Wikipedia, we conduct zero-shot evaluations using the retrieval skill from our pretrained model on NQ, WebQ, EntityQuestions and HotpotQA. To assess the model's ability to handle expanded query retrieval, we design an oracle second-hop retrieval setting (gold first-hop evidence is used) based on HotpotQA. Following Izacard et al. (2021) and Ram et al. (2022), we report top-k retrieval accuracy (answer recall), *i.e.,* the percentage of questions for which the answer string is found in the top-k passages.

Supervised In-domain Evaluation We further fine-tune our pretrained model with two extra skills (entity span proposal and reranking) on NQ, HotpotQA and OTT-QA, again in a multi-task fashion. Unlike multi-hop data with supervision for all skills, only single retrieval and reranking data is available for NQ. During training, all datasets are treated equally without any loss balancing. Different from previous retrieval-only work, we explore `Chain-of-Skills` retrieval by using different skill configurations. Specifically, we use skill configuration for task A, B and C shown in Figure 6.1 for NQ, OTT-QA and HotpotQA, respectively. We again report top-k retrieval accuracy for NQ and OTT-QA following previous work. For HotpotQA, we follow the literature using the top-1 pair of evidence accuracy (passage EM).

	Top-20	Top-100
DPR-multi (Karpukhin et al., 2020)	79.5	86.1
ANCE-multi (Xiong et al., 2020a)	82.1	87.9
DPR-PAQ (Oguz et al., 2022)	84.7	89.2
co-Condenser (Gao and Callan, 2022)	84.3	89.0
SPAR-wiki (Chen et al., 2021c)	83.0	88.8
COS	**85.6**	**90.2**

Table 6.2: Supervised top-k accuracy on NQ test.

	Top-20	Top-50	Top-100
CORE (Ma et al., 2022a)	74.5	82.9	87.1
COS	79.9	**88.9**	**92.2**
COS w/ CORE configuration	**80.5**	88.6	91.8

Table 6.3: Supervised top-k accuracy on OTT-QA dev.

Cross-data Evaluation To test the model's robustness towards domain shift, we conduct cross-data evaluations on SQuAD and EntityQuestions. Although considerable success has been achieved for supervised dense retrievers using in-domain evaluations, those models have a hard time generalizing to query distribution shift (*e.g.,* questions about rare entities; Sciavolino et al., 2021) compared with BM25. In particular, we are interested to see whether `Chain-of-Skills` retrieval is more robust. Again, top-k retrieval accuracy is used.

6.4.3 Results

	Passage EM
MDR (Xiong et al., 2020b)	81.20
Baleen (Khattab et al., 2021)	86.10
IRRR (Qi et al., 2021)	84.10
TPRR (Zhang et al., 2021a)	86.19
HopRetriever-plus (Li et al., 2021b)	86.94
AISO (Zhu et al., 2021)	88.17
COS	**88.89**

Table 6.4: Supervised passage EM on HotpotQA dev.

Zero-shot Results For zero-shot evaluations, we use two recent self-supervised dense retrievers,

Contriever Izacard et al. (2021) and Spider Ram et al. (2022), and BM25 as baselines. The results are presented in Table 6.1. As we can see, BM25 is a strong baseline matching the average retrieval performance of Spider and Contriever over considered datasets. COS achieves similar results on NQ and WebQ compared with self-supervised dense methods. On the other hand, we observe significant gains on HotpotQA and EntityQuestions, where both dense retrievers are lacking. In summary, our model shows superior zero-shot performance in terms of average answer recall across the board, surpassing BM25 with the largest gains, which indicates the benefit of our multi-task pretraining.

Supervised In-domain Results As various customized retrievers are developed for NQ, OTT-QA and HotpotQA, we compare COS with different dataset-specific baselines separately. For NQ, we report two types of baselines, 1) bi-encoders with multi-dataset training and 2) models with *augmented pretraining*. For the first type, we have DPR-multi (Karpukhin et al., 2020) and ANCE-multi (Xiong et al., 2020a), where the DPR model is initialized from BERT-based and ANCE is initialized from DPR. For the second type, DPR-PAQ (Oguz et al., 2022) is initialized from the RoBERTa-large model (Liu et al., 2019b) with pretraining using synthetic queries (the PAQ corpus (Lewis et al., 2021)), co-Condenser (Gao and Callan, 2022) incorporated retrieval-oriented modeling during language model pretraining on Wikipedia; SPAR-wiki (Chen et al., 2021c) combine a pretrained lexical model on Wikipedia with a dataset-specific dense retriever. Both co-Condenser and SPAR-wiki are initialized from BERT-base. As shown by results for NQ (Table 6.2), COS outperforms all baselines with or without pretraining. It is particularly encouraging that despite being a smaller model, COS achieves superior performance than DPR-PAQ. The reasons are two-fold: Oguz et al. (2022) has shown that scaling up the retriever from base to large size only provides limited gains after pretraining. Moreover, DPR-PAQ only learns a single retrieval skill, whereas COS can combine multiple skills for inference. We defer the analysis of the advantage of chain-of-skills inference later (§6.5.2).

For OTT-QA, we only compare with the SOTA model CORE (Ma et al., 2022a), because other OTT-QA specific retrievers are not directly comparable where extra customized knowledge source is used. As CORE also uses multiple skills to find evidence chains, we include a baseline where the inference follows the CORE skill configuration but uses modules from COS. For HotpotQA, we compare against three types of baselines, dense retrievers focused on expanded query retrieval MDR (Xiong et al., 2020b) and Baleen Khattab et al. (2021), sparse retrieval combined with query reformulation IRRR (Qi et al., 2021) and TPRR (Zhang et al., 2021a) and ensemble of dense, sparse and hyperlink retrieval HopRetriever (Li et al., 2021b) and AISO (Zhu et al., 2021). The results on OTT-QA and HotpotQA are summarized in Table 6.3 and Table 6.4. It is easy to see that COS outperforms all the baselines here, again showing the advantage of our configurable multi-skill model over multiple types of ODQA tasks. Later, our analyses show that both Chain-of-Skills inference and pretraining contribute to the observed gains.

Cross-data Results Given that both EntityQuestions and SQuAD are single-hop, we use baselines on NQ with improved robustness for comparison. Particularly, SPAR-wiki is an ensemble of two dense models with one pretrained using BM25 supervision on Wikipedia and the other fine-tuned on NQ. BM25 is included here, as it is found to achieve better performance than its dense counterpart on those two datasets. The evaluation results are shown in Table 6.5. Overall,

	EntityQuestions		SQuAD	
	Top-20	**Top-100**	**Top-20**	**Top-100**
BM25	70.8	79.2	71.1	81.8
DPR-multi (Karpukhin et al., 2020)	56.6	70.1	52.0	67.7
SPAR-wiki (Chen et al., 2021c)	73.6	81.5	**73.0**	**83.6**
COS	**76.3**	**82.4**	72.6	81.2

Table 6.5: Cross-dataset top-k accuracy on test sets.

	#Params	**Top-20**	**Top-100**
`Chain-of-Skills` inference			
No Expert	111M	90.2	92.4
FFN Expert(naive)	252M	91.3	93.4
MHA Expert(naive)	182M	92.0	94.0
MHA Expert(COS)	182M	92.0	94.2
Retrieval-only inference			
Multi-hop Retriever(fully separated modules)	110M(440M)	85.1	88.9
MHA Expert(naive)	182M	82.8	87.0
MHA Expert(COS)	182M	85.9	89.6

Table 6.6: Ablation results on HotpotQA dev using top-k retrieval accuracy. All models are initialized from BERT-base and trained on HotpotQA only.

our model achieves the largest gains over BM25 on both datasets, indicating that our multi-task fine-tuned model with `Chain-of-Skills` inference is more robust than previous retrieval-only approaches.

6.5 Analysis

6.5.1 Task Interference

We conduct ablation studies on HotpotQA to compare different ways of implementing skill-specific specialization (discussed in §6.3.2) and their effects on task interference. As MHA experts are used for our model, we consider two variants for comparison: 1) the no-expert model where all tasks share one encoder, and 2) the FFN expert model where specialized FFN sub-layers are used. Then we also compare the proposed expert configuration with a variant where the expanded query retrieval inputs share the same expert as single retrieval, denoted as the naive setting. The results are shown in the upper half of Table 6.6. Compared with the no-expert model, both FFN and MHA experts can effectively reduce task interference, wherein MHA expert is more effective overall. Our proposed expert configuration can further help.

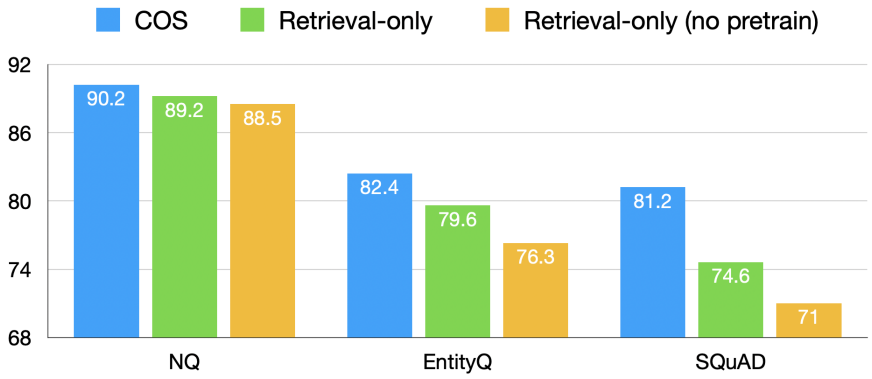

Figure 6.4: Top-100 retrieval accuracy on inference strategy: `Chain-of-Skills` vs retrieval-only.

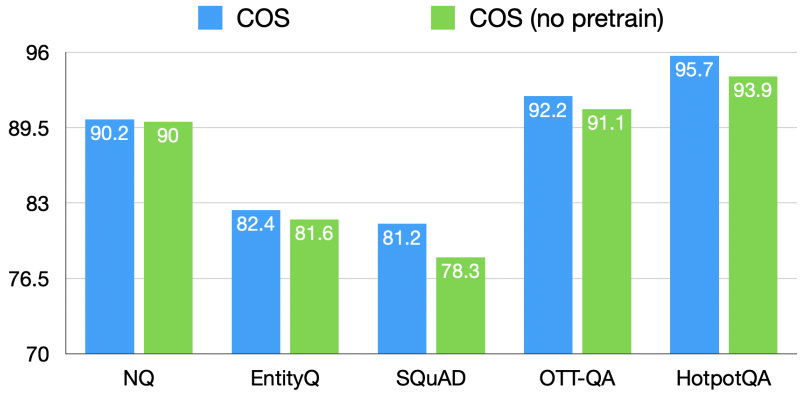

Figure 6.5: Comparison on the effect of pretraining using top-100 retrieval accuracy with COS inference.

6.5.2 Benefit of `Chain-of-Skills` Inference

Here we explore the benefits of the chained skill inference over the retrieval-only version. We additionally train a multi-hop retriever following Xiong et al. (2020b), and compare it with the two MHA expert models using the same two rounds of retrieval-only inference. The comparison is shown in the lower part of Table 6.6. As we can see, retrieval-only inference suffers large drops in performance. Although our proposed and naive MHA expert configurations have similar performance using `Chain-of-Skills` inference, the naive configuration model shows severe degradation caused by task interference compared with the multi-hop retriever, validating the effectiveness of our proposed model. We further compare our `Chain-of-Skills` inference with the retrieval-only inference on NQ, EntityQuestions and SQuAD in Figure 6.4. It is easy to see that our pretraining can benefit the retrieval-only version. However, using better skill configurations via `Chain-of-Skills` inference yields further improvements, particularly on those unseen datasets.

6.5.3 Reduced Model Size and Computation

The multi-hop retriever experiment considered in subsection 6.5.2 can also be viewed as a setting where all modules in COS are fully separated. In that case, the entire system will have approximately 440M parameters, which is more than two times larger than `COS`. Compared to the `CORE` system proposed in the previous chapter, which has about 3.5B parameters, `COS` only has a small fraction of its parameters. Despite its much smaller size, `COS` is able to achieve stronger performance than these separately trained models, as shown in Table 6.6 and Table 6.3. Moreover, another benefit of our proposed expert configuration is the reduced computation when answering complex questions, compared to naive configuration or separate modules. Since the rerank expert and expanded-query expert are merged into the same module, we only need one forward pass to compute the representation used by both skills. In particular, after reranking the first hop results (found by either single retrieval or question linking), the output representations can be directly used to retrieve the second hop evidence. This naturally leads to improved computation efficiency and our results in Table 6.6 show that it also achieves better results.

6.5.4 Effect of Pretraining

To further demonstrate the benefit of our proposed multi-task pretraining, we fine-tune another multi-task model following the same training protocol as `COS` but BERT model weights are used for initialization. Both `COS` and the model without pretraining are then using the same skill configuration for inference. The results are illustrated in Figure 6.5. Similar to the retrieval-only version (Figure 6.4), we find that `COS` consistently outperforms the multi-task model without pretraining across all considered datasets using `Chain-of-Skills` inference. Again, the pretrained model is found to achieve improvements across the board, especially on out-of-domain datasets, which validates the benefits of our multi-task pretraining.

	Query	Doc	**Top-20**	**Top-100**
Single query*	0	1	**96.1**	**98.2**
Single query	4	1	90.1	95.2
Single query	2	1	91.8	95.9
Single query	2	3	87.4	92.7
Expanded query	0	1	94.2	97.0
Expanded query*	4	1	**95.3**	**97.4**
Expanded query	2	1	74.5	85.8
Expanded query	2	3	67.3	79.6

Table 6.7: Results of feeding the inputs to different experts, where the first two columns represent the query expert id and document expert id. * denotes the proposed setup

6.5.5 Swapping Experts

To understand if different experts in our model learned different specialized knowledge, we experiment with swapping experts for different inputs on HotpotQA. In particular, we feed the single query input and expanded query input to different query experts and then retrieve from either the context passage index or the entity description index. For single query input, we measure if the model can retrieve one of the positive passages. For expanded query input, we compute the recall for the other positive passage as done in (§6.4.3). The results are shown in Table 6.7. Although both the single query expert and the expanded query expert learn to retrieve evidence using the `[CLS]` token, swapping the expert for either of these input types leads to a significant decrease in performance. Also, switching to the entity query expert and retrieving from the entity description index results in a large drop for both types of inputs. This implies that each specialized expert acquires distinct knowledge and cannot be substituted for one another.

6.6 Question Answering Experiments

Here, we conduct end-to-end question-answering experiments on NQ, OTT-QA and HotpotQA, using retrieval results from `COS`. Following the literature, we report exact match (EM) accuracy and F1 score.

For NQ and OTT-QA, we re-implement the Fusion-in-Encoder (FiE) model (Kedia et al., 2022) because of its superior performance on NQ. For NQ, the model reads top-100 passages returned by `COS`, and for OTT-QA, the model reads top-50 evidence chains, in order to be comparable with previous work. Here, separate models are trained for each dataset independently. Due to space constraints, we only present the results on OTT-QA and leave the NQ results to Table D.2. The OTT-QA results are summarized in Table 6.8. Our model, when coupled with the FiE, is able to outperform the previous baselines by large margins on OTT-QA, and we can see that the superior performance of our model is mainly due to `COS`.

Finally, for HotpotQA, since the task requires the model to predict supporting sentences in

	Dev		Test	
	EM	**F1**	**EM**	**F1**
HYBRIDER (Chen et al., 2020d)	10.3	13.0	9.7	12.8
FR+CBR(Chen et al., 2020a)	28.1	32.5	27.2	31.5
CARP (Zhong et al., 2022)	33.2	38.6	32.5	38.5
OTTer (Huang et al., 2022)	37.1	42.8	37.3	43.1
CORE (Ma et al., 2022a)	49.0	55.7	47.3	54.1
CORE + FiE	51.4	57.8	-	-
COS + FiE	**56.9**	**63.2**	**54.9**	**61.5**

Table 6.8: End-to-end QA results on OTT-QA.

addition to the answer span, we follow Zhu et al. (2021) to train a separate reader model to learn answer prediction and supporting sentence prediction jointly. Due to space constraints, we leave the full results to Table D.3. Overall, our method achieves competitive QA performance against the previous SOTA with improved exact match accuracy.

6.7 Related Work

Dense retrievers are widely used in recent literature for ODQA (Lee et al., 2019; Karpukhin et al., 2020). While most previous work focuses on single retrieval (Xiong et al., 2020a; Qu et al., 2021), some efforts have also been made towards better handling of other query types. Xiong et al. (2020b) propose a joint model to handle both single retrieval and expanded query retrieval. Chen et al. (2021c) train a dense model to learn salient phrase retrieval. Ma et al. (2022a) build an entity linker to handle multi-hop retrieval. Nevertheless, all those models are still customized for specific datasets, *e.g.,* only a subset of query types are considered or separate models are used, making them un-reusable and computationally intensive. We address these problems by pinning down a set of functional skills that enable joint learning over multiple datasets.

Mixure-of-expert models have also become popular recently (Fedus et al., 2021). Methods like gated routing (Lepikhin et al., 2020) or stochastic routing of experts (Zuo et al., 2021) do not differentiate the knowledge learned by different experts. Instead, our work builds expert modules that learn reusable skills that can be flexibly combined for different use cases.

Another line of work focus on unsupervised dense retrievers using self-supervised data constructed from the inverse-cloze-task (Lee et al., 2019), random croppings (Izacard et al., 2021), truncation of passages with the same span (Ram et al., 2022), hyperlink-induced passages (Zhou et al., 2022) or synthetic QA pairs (Oguz et al., 2022). Other model architecture adjustments on Transformer for retrieval are proposed (Gao and Callan, 2021, 2022). Our work can be viewed as a synergy of both. Our multi-task pretrained model can perform better zero-shot retrieval. Our modular retriever can be further fine-tuned in a multi-task fashion to achieve better performance.

6.8 Discussion and Conclusions

In this chapter, we propose a modular model Chain-of-Skills (COS) that learns five reusable skills for ODQA via multi-task learning. We reformulated all skills' learning objectives into contrastive learning and we train all modules with implicit knowledge sharing. To reduce task interference, we design a new parameterization for skill modules on top of the shared encoder. We conduct self-supervised pretraining and supervised finetuning, both in a multi-task learning fashion. COS can directly perform superior zero-shot retrieval after multi-task self-supervised pretraining on Wikipedia. We also show that skills learned by COS can be flexibly chained together to better fit the target task. When fine-tuned on multiple datasets, COS achieves SOTA results across the board. Hence the model is able to generalize across domains, knowledge types and question types. Similar to CORE from the previous chapter, each skill's execution during Chain-of-Skills inference naturally offers a more interpretable intermediate reasoning process. Finally, with the implicit knowledge sharing across modules, the overall model size and inference computation are also greatly reduced, leading to better efficiency. With all aforementioned advantages, COS is able to achieve all three target properties of an ideal QA system that we set out to build in this thesis.

At the same time, we realize that there still exists ample room for improvement: Our current COS's reranking expert only learns to rerank single-step results. Thus it can not model the interaction between documents in case of multi-passage evidence chains, which might lead to sub-optimal performance, *e.g.,* when we need to rerank the full evidence path for HotpotQA. At the same time, we hypothesize that the capacity of the small model used in our experiments is insufficient for modeling evidence chain reranking. We leave the exploration of learning a full path reranker for future work.

Also, our current pretraining setup only includes the three bi-encoder tasks, and thus we can not use the pretrained model out-of-box to solve tasks like end-to-end entity linking. Consequently, the learned skills from self-supervision can not be chained together to perform configurable zero-shot retrieval. It would be interesting to also include the entity span proposal skill in the pretraining stage, which could unleash the full potential of the Chain-of-Skills inference for zero-shot scenarios.

Chapter 7

Conclusions and Future Work

Previous works on QA often build customized models for specific datasets, without paying much attention to how systems generalize to different domains in an interpretable and efficient way, which hinders their application to real-world use cases. With the aim to build generalizable, interpretably and efficient QA systems, this thesis proposed various methods that effectively incorporate both explicit knowledge and implicit knowledge. We show that fusing the two types of knowledge improves models' generalization to different domains of questions, both in supervised and unsupervised settings. Building modularized frameworks not only further enhances models' generalization to different question types and knowledge types, but also offers better interpretability of the intermediate reasoning processes. Building on top of the modularized framework, sharing implicit knowledge across different modules further improved models' efficiency, while maintaining the advantages on generalization and interpretability, thus achieving all three target properties of the ideal system. Next, we summarize the contribution made in each chapter, present additional works that are also related to this thesis and discuss open challenges that are worth exploring for future work.

7.1 Summary of Contributions

- In Chapter 2 (Hybrid Knowledge Fusion), we studied various methods for incorporating knowledge from commonsense knowledge graphs into pretrained language models. We proposed different ways of eliciting suitable knowledge for input questions and developed a novel model architecture to encode and inject the extracted knowledge. The injected explicit knowledge helps bridge the knowledge gap in the implicit knowledge, therefore improving the systems' generalization in supervised settings.

- In Chapter 3 (Zero-shot QA Framework), we proposed a novel zero-shot QA framework, which automatically constructed large-scale QA datasets from commonsense knowledge graphs. With the carefully designed distractor sampling strategy, we show that models trained on this synthetic data can generalize well to a diverse set of commonsense question answering tasks in an unsupervised setting.

- In Chapter 4 (Unified Knowledge Interface), we proposed a verbalizer-retriever-reader framework that can effectively leverage explicit knowledge from unstructured text, structured ta-

bles and knowledge bases. We proposed a novel data-to-text verbalizer model that can handle both types of structured knowledge. This framework greatly improved the models' performance on open-domain QA tasks by leveraging the structured knowledge, even without updating the model, therefore achieving better generalization across knowledge types.

- In Chapter 5 (Chain-of-Reasoning), we proposed a chain-of-reasoning framework that decomposes the overall reasoning process in QA into four discrete steps, and each of the reasoning steps is implemented by a separate module. This framework naturally offers better interpretability by providing transparent intermediate reasoning steps, and it greatly enhances the systems' generalization across different question types by solving both simple and complex questions using a single model.

- In Chapter 6 (Chain-of-Skills), we introduced a further refined chain-of-skills model that inherent the benefits from the chain-of-reasoning framework. On top of this, we further allowed implicit knowledge sharing across different modules by merging all modules into a shared encoder and adding skill-specific specializations. Such an architecture greatly reduced the model size and computation cost. We also show that the model can be configured flexibly at inference time to achieve optimal performance of different tasks, i.e. chaining the skills in different ways. Therefore it also achieves much stronger generalization across different dimensions and offers an interpretable reasoning process.

7.2 Beyond This Thesis

Besides the work presented above, we have also explored other closely related areas which we briefly discuss here:

- **Hybrid Knowledge Fusion** For this part, we have also explored different dimensions of commonsense knowledge, and how the dimension relates to the impact on the downstream QA tasks. We start by categorizing commonsense knowledge from different knowledge graphs (Ilievski et al., 2020) into 13 commonsense dimensions using the relation. Then we transform each dimension of knowledge into synthetic QA sets to train models following the same pipeline we introduced in Chapter 3. We find that the dimensions that the models are not familiar with (i.e. low performance on synthetic data from that dimension) bring larger gains on downstream QA tasks performance (Ilievski et al., 2021a). On top of the zero-shot QA framework, we further explored the impact of the amount of synthetic QA pairs, different training regimes (single answer scoring vs. joint answer scoring) and different model sizes. We experimented with different data sub-sampling strategies and trained models ranging from 220M to 3B parameters with different loss functions. Our results show that the optimal amount of data depends on the model size, and the performance on different subsets of questions also reflects the characters of the synthetic data (Zhang et al., 2022a). Finally, we have also extended the study on knowledge injection methods to generative commonsense reasoning tasks (Li et al., 2020a). Again we show that augmented commonsense knowledge leads to improved task performance.

- **Model Analysis** For this part, we investigated the models' error patterns across different kinds of QA tasks and different model architectures. We selected four representative QA models (QANet (Yu et al., 2018a), BERT, DS-QA (Lin et al., 2018) and MultihopPointer Generator (Bauer et al., 2018)) and experimented with four popular QA benchmarks. Then we manually annotated the models' errors across all four benchmarks (SQuAD (Rajpurkar et al., 2016), HotpotQA (Yang et al., 2018), SearchQA (Dunn et al., 2017) and MSMARCO (Bajaj et al., 2016)). We found that design decisions when building models can often lead to specific types of errors (Pugaliya et al., 2019) e.g. a model with a copy mechanism is more likely to copy high-frequency words as output answer. We also suggest that a single dataset or datasets with similar characteristics are often insufficient to fully evaluate the generalizability of the model. Moreover, we also explored optimal strategies for adapting pretrained language models to commonsense reasoning tasks. In particular, we compared full finetuning with light-weight adaptation methods including prefix-tuning (Li and Liang, 2021) and Autoprompt (Shin et al., 2020). We found that different adaptation methods actually train the model to learn different inductive biases, e.g. content vs task structure. While finetuning leads the model to mostly memorize the content of the training set, lightweight adaptation methods can additionally teach the model to learn task structure, which achieves much stronger performance on unseen questions (Ma et al., 2021).

- **Procedural Reasoning** For this part, we studied the task of procedural text understanding where the model needs to track the states of entities across a narrative and make inference based on the entity states. We first proposed a novel model that effectively coalesced global context information with local time-stamped and entity-centric information (Ma et al., 2022c). When coupled with commonsense constraints and structured prediction, this model achieved state-of-the-art results on a popular procedural reasoning ProPara (Dalvi et al., 2018) and a story understanding benchmark TRIP (Storks et al., 2021). Since the procedural reasoning data requires dense annotation of entity states across the narrative, the dataset sizes are usually very small (Dalvi et al., 2018). To address this issue, We further explored different strategies to automatically generate densely annotated procedural reasoning data for training models in our follow-up study. We proposed a novel data generation framework leveraging semantic parsing tools (Allen and Teng, 2017) and a large language model (Chen et al., 2021a) to produce high-quality annotations for both natural stories (Mostafazadeh et al., 2016) and synthetic stories (Ilievski et al., 2021b). Our results show that the models trained on our automatically labeled data achieve much stronger out-of-domain generalization (Jiang et al., 2023a).

- **Other Applications** We have also explored various applications of the QA models we developed in the past and proposed new challenges to encourage future work in this direction. In particular, we investigated the models' reasoning abilities in traffic scenarios, i.e. can models answer questions based on the textual descriptions of traffic situations? We first repurposed existing traffic understanding datasets (Kim et al., 2018; Xu et al., 2021) into question answering format, and further collected additional data from human driving tests. Then we conducted a thorough evaluation of existing QA models trained with different kinds of su-

pervision (e.g. mixture of QA datasets Khashabi et al. (2020), natural language inference dataset (Williams et al., 2018) or commonsense knowledge graphs (Ma et al., 2020)). We found that while none of the models can achieve good performance consistently across all tasks, the models actually exhibit different strengths and weaknesses for different question types, suggesting ample room for improvement (Zhang et al., 2023). Besides traffic understanding, we also studied text-based games (Wang et al., 2022), where the model needs to interact with a virtual environment to complete a complex goal. Here, we incorporate the affordance knowledge about the objects in the environment as well as the memory of the past actions into reinforcement learning agents (He et al., 2016; Ammanabrolu and Hausknecht, 2020). We also leveraged a commonsense reasoning model (Zhang et al., 2022a) for action selection by considering each step in the episode separately. We found that across different models, incorporating additional knowledge is beneficial for most tasks (Chhikara et al., 2023). Finally, inspired by the success of the large language models, we proposed a novel task that requires lateral thinking abilities to solve, to test the true reasoning ability of the state-of-the-art models. We start by collecting brainteasers from online sources and then manually converting them into multiple-choice question answer pairs. To collect high-quality distractors, we prompted commonsense generator models (Bosselut et al., 2019) and LLM (e.g. ChatGPT) to get initial distractors and then manually corrected errors. We experimented with a set of instruction finetuned models (Chung et al., 2022) as well as commonsense enhanced models (Ma et al., 2020; Wang et al., 2023). We found that even the largest model, ChatGPT struggles to perform well on our new benchmark. The commonsense model performs even worse than the vanilla model in some cases, suggesting that commonsense knowledge can sometimes be a confounding factor for the lateral thinking process. Overall, our work presented an interesting benchmark that could be useful for future works on machine reasoning.

7.3 Open Challenges

The recent success of large language models (LLM) has revolutionized the field of NLP and machine learning. Instruction-finetuned models such as ChatGPT or GPT-4 are able to solve many challenging tasks in zero-shot or few-shot fashion (Kojima et al., 2022; Wei et al., 2022c,a). Consequently, the community is shifting towards adapting LLM for various tasks and applications (Shen et al., 2023). While it's tempting to leverage these models to solve any task at hand, these models still struggle with knowledge-intensive tasks such as question answering and fact-checking, even after scaling to billions of parameters (Trivedi et al., 2022). As previously discussed, these pretrained language models are good at encoding frequently appearing explicit knowledge as its implicit knowledge, but they can not handle less-frequent knowledge well. This has led to the problem of *hallucination* (Ji et al., 2023), i.e. generating false information in natural and convincing ways. In general, LLMs are versatile reasoners but they are unreliable, hindering their application to many use cases with high stakes. For example, people are more tolerant of the models' mistakes and false information during chit-chat, but it poses a much higher risk when using these models for educational purposes. Thus how to address the *hallucination* problems in the LLMs is a promising

direction for future work.

Moreover, these powerful LLMs have opened up countless possibilities for building AI-powered applications, e.g. general-purpose virtual assistants and embodied agents (Nakano et al., 2021; Driess et al., 2023). There has also been the discussion of artificial general intelligence (AGI) (Bubeck et al., 2023), suggesting the potential of these models to be like humans. Much like how humans interact with the world, these models are also expected to leverage different tools to complete various tasks. Then how to create the synergy between LLMs and existing tools/frameworks/infrastructure in the world is another interesting field to explore.

In addition, one major drawback of existing LLMs is that they are trained on stale knowledge. Consequently, they can not answer questions about the events that happened after their knowledge cutoff date. Although leveraging a search engine to find the latest information can address this issue to some extent, e.g. Bing Chat, there are also cases this method cannot resolve. For example, when a certain fact memorized by the LLMs is wrong, we would like to edit its knowledge to correct it. How to do so remains another challenging problem to solve.

Finally, another major bottleneck of current LLMs is their high computation cost. Training and serving such models would require thousands of high-end GPUs, and this cost is prohibitively expensive for many small businesses and research labs. Even for large corporations, serving these models to millions of users has been challenging. For example, OpenAI has been limiting the number of requests for GPT-4 model to 25 per 3 hours for every user. Thus, how to make these models more efficient while keeping or improving the level of performance is now a million-dollar question in the community.

Considering all the aforementioned challenges and what we have accomplished in this thesis, we identify the following specific directions that are worth exploring:

- **Retrieval-augmented generation**: As discussed in previous chapters, providing explicit knowledge to the reader model when answering questions effectively improves its performance. Similarly, we can also provide relevant evidence documents to the LLMs using a retriever, to enhance its generation quality. By retrieving knowledge that is not learned/memorized by LLMs, we can effectively reduce the *hallucination* problem. Several recent studies have made progress along this direction (Press et al., 2022; Khattab et al., 2022). However, this problem is far from being solved and there are still many research questions yet to answer.

  When should we do retrieval? Although state-of-the-art retrievers have achieved very strong performance (Ma et al., 2023), they are far from being perfect. Consequently, the errors made by the retriever can also mislead the generation of LLMs. Also, feeding the retrieved documents to LLMs significantly increased the prompt length, which also implies a much higher cost. Ideally, we would want to retrieve documents when the LLMs are not confident about their generation and avoid adding long and potentially distracting contexts when the LLMs are making good predictions. Some recent works have shed light on how to address this issue (Jiang et al., 2023b), yet there is still ample room for improvement.

  Besides when to retrieve, *how to retrieve* is also an interesting question to explore. For example, for complex QA tasks, many recent works propose to prompt LLM to decompose

the complex question and use the resulting simple questions for retrieval, but there is also the risk of error propagation. On the other hand, directly using the complex query might be challenging for retrievers that are not trained with similar data. Moreover, for long document generation tasks, there is no explicit query. What to use as the query for retrieval is also an interesting question (Jiang et al., 2023b).

Finally, there is the question of *What to include from the retrieval results?* In our own works from chapter 4, and chapter 6, we usually use the top 100 documents returned by the retrieve for answer reasoning. However, such a large amount of documents would be infeasible for LLMs to handle. Naturally, we have to trade-off between evidence recall and context window size. Existing work typically feeds the top 1 - top 5 documents to LLMs for answer prediction (Press et al., 2022; Khattab et al., 2022). However, using LLMs to rerank the results before final answer generation is also an interesting direction to explore (Sun et al., 2023). LLMs probably have the ability to further refine the retrieval results and filter out irrelevant information, which could lead to better overall performance. Then the remaining question is how can we elicit such reasoning ability from them? Naturally, a follow-up question would be: is there a way to efficiently rerank a large set of documents using LLMs? In this thesis, we have proposed various methods for retrieving and reranking relevant knowledge for QA tasks, the ideas and insights we had in previous chapters can also be extended to the territory of LLMs.

- **Tool-augmented LLMs**: Recently, the concept of empowering LLMs with tools has been widely adopted in the community (Schick et al., 2023; Wu et al., 2023). Here, the idea refers to delegating a specific functionality to a dedicated tool/model, instead of letting the LLMs solve the task end-to-end. For example, when solving math-related questions, instead of letting the model generate all reasoning steps and complete all the calculations by itself, we can just ask the model to generate programs for computing the final answer, and execute the programs in a Python interpreter, i.e. using a calculator (Gao et al., 2022). Such methods offload the parts of the task that are challenging for LLMs and enjoy the advantage of external tools, leading to much better results on a variety of tasks (Chen et al., 2022). E.g. a deterministic and correct program will always produce the correct answer, regardless of the syntax or language, but LLMs could easily fail with slightly different prompts. The retrieval-augmented generation framework can also be viewed as one kind of tool-augmented LLM, where the retriever is the tool LLMs use to find more relevant knowledge. Thus the models we built in this thesis can naturally be incorporated in this direction. In other words, by viewing the models such as `COS` and `UDT-QA` as tools, we can teach LLMs to use them to better answer user questions. The concept of tools also raises many interesting research questions:

  How to make LLMs actively use and compose tools? As discussed earlier, we would like the LLM to delegate the tasks that are challenging for it to solve to specific tools. However, how to make this decision intelligently is a non-trivial problem. Some recent works have tried to add the tools/functions' description in the LLM prompts, and use in-context examples to illustrate how to use those functions (Paranjape et al., 2023; Lu et al., 2023). While such methods work well on certain benchmarks, whether they can generalize to a general-

purpose agent is an open question. For example, in the academic benchmarks, it's almost guaranteed that function calls are required at every step to solve the input question, whereas in a chat session with a user, the need to call external functions might come up at any point in the conversation or might not come at all. Moreover, when the user instructions become complex, the LLM might need to compose multiple tools hierarchically to complete the task. Thus designing a smart tool-use mechanism is an interesting direction for future work.

How to deal with a large tool library? After hooking LLMs with external tools, the number of tools available can quickly explode. Augmenting the LLM prompts with tool descriptions is an effective approach when we only have a handful of tools. However, it can not scale up to thousands of tools due to the limited context window size. One possibility is to finetune the LLM with tool-calling data (Schick et al., 2023), and others have tried to augment the LLM's token embeddings only to reduce compute (Hao et al., 2023). We can also potentially treat the problem of selecting the appropriate tool for retrieval and use retrievers to find the most suitable tool. Still, a lot more research efforts are required to understand what's the best approach here.

Can LLMs build tools by themselves? Lastly, an interesting question is what should we do when we do not have a suitable tool at hand? Humans can learn to build new tools from raw materials when the existing ones do not meet their needs. It makes us wonder if LLMs can learn to do the same. All of the aforementioned tools for LLMs are computer programs after all, which are composed by code. At the same time, LLMs have exhibited very strong performance on code generation tasks (Cheng et al., 2022b; Ni et al., 2023). Thus LLMs should be able to write code to build new tools for solving specific problems. Additionally, they should be able to learn to improve their outputs or tools from the feedback, e.g. execution errors, rewards, or human preference, much like how humans build prototypes and keep improving. Some studies have shown that LLMs can also learn to self-improve from their own feedback (Madaan et al., 2023). All of these are interesting directions for future research.

- **LLM knowledge editing**. One thing users often find unsatisfying about LLMs is their stale knowledge. Since these models are trained on fixed snapshots of data, they do not have the ability to answer questions about the latest events and their implicit knowledge can not be easily updated. Although retrieval-augmented-generation can alleviate the formal problem to some extent, they are not suitable when we want to update the model's knowledge, e.g. correcting factual errors. Ideally, an intelligent assistant should be able to learn from its interaction with the world and constantly update its knowledge. There are many challenging research questions implied by such ability:

  Where is implicit knowledge located? In order to update a model's knowledge, we first need to know where the knowledge is located. Although we generally say that the implicit knowledge is stored in the model's parameters, it's extremely challenging to locate the specific parameters in the model that are associated with one particular fact. Although some studies have made progress towards understanding the implicit knowledge, e.g. facts are stored in FFN layers (Meng et al., 2022), this problem is far from being solved. Also, understanding

where is the knowledge stored may also help us understand how the model learned such knowledge, i.e. understanding where the emergent abilities of LLMs come from Wei et al. (2022a).

How do we edit a specific fact in the model? Due to the black-box nature of neural models, editing a specific piece of knowledge is a very challenging task. Some previous works have made some progress in this direction, through gradient-based updates at test time (De Cao et al., 2021) or training a separate editor network (Mitchell et al., 2021). However, as models get larger, these methods might not be as effective. Thus, there are ample room to explore for future works in the era of LLMs.

How to build a memory for LLMs? Recently, numerous efforts have tried to equip the LLMs with a memory component that stores past errors and user feedback (Madaan et al., 2022). The memory can then be used to improve future predictions and potentially correct previous mistakes. Hence the memory component can be thought of as another way for knowledge editing. The benefit of this approach is that no parameter update is required and we can flexibly grow the memory component. However, this could potentially create conflicts between the implicit knowledge and the memory, how to address such conflicts remains an open question. Moreover, as the memory size grows large, more advanced mechanisms are probably required to extract the most relevant memory fragments to solve the task at hand, similar to retrieval-augmented generation (Jiang et al., 2023b). Finally, the memory component also opens up the possibility of LLM personalization. If every user has access to their dedicated LLM assistant, who can store all of their interaction history into memory (Park et al., 2023) and use memory to guide future interactions, they will be able to create a truly personalized virtual assistant.

- **LLM specialization** Lastly, one major bottleneck of current LLMs is their huge model size and computation requirement. Training and serving such models require thousands of cutting-edge GPUs, which is unaffordable for many institutions and labs. This has motivated people to explore various cheaper alternatives for training/adapting LLMs (Hu et al., 2021). However, these methods do not reduce the inference computation cost. Taking a step back, one interesting question is, do we really need an entire LLM to solve every task? Intuitive, humans activate different parts of the brain for different scenarios, e.g. left brain for logic and right brain for creativity. Thus, treating an LLM as a human brain, it would be ideal to activate different parts of the model for different tasks. This also resembles the idea of Mixture-of-Experts (Fedus et al., 2021), except that here we can enforce a clear specialization strategy by learning one skill in each expert (Ma et al., 2023) or one domain in each expert (Gururangan et al., 2023). With such specialization, we can significantly reduce the serving cost of LLMs. Similarly, we can use a smaller model for easier tasks and switch to a larger model for more difficult parts (Madaan and Yang, 2022). Studies have shown that by having a mixture of smaller models, the overall performance can be improved compared to having a single large model with an equal number of parameters (Gururangan et al., 2023). This evidence suggests that focusing on specialization might be an important step for building next-generation of LLMs.

Bibliography

Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. 2021. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3554–3565, Online. Association for Computational Linguistics.

Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. 2020. ETC: Encoding long and structured inputs in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 268–284, Online. Association for Computational Linguistics.

James F. Allen and Choh Man Teng. 2017. Broad coverage, domain-generic deep semantic parsing. In *AAAI Spring Symposia*.

Prithviraj Ammanabrolu and Matthew Hausknecht. 2020. Graph constrained reinforcement learning for natural language action spaces.

Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. 2019. Learning to retrieve reasoning paths over wikipedia graph for question answering.

Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. 2020. Learning to retrieve reasoning paths over wikipedia graph for question answering. In *International Conference on Learning Representations*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2016. Ms marco: A human generated machine reading comprehension dataset.

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2895–2905, Florence, Italy. Association for Computational Linguistics.

Pratyay Banerjee and Chitta Baral. 2020. Self-supervised knowledge triplet learning for zero-shot question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 151–162, Online. Association for Computational Linguistics.

Satanjeev Banerjee and Alon Lavie. 2005. 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*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Lisa Bauer, Yicheng Wang, and Mohit Bansal. 2018. Commonsense for generative multi-hop question answering tasks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4220–4230, Brussels, Belgium. Association for Computational Linguistics.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Scott Wen-tau Yih, and Yejin Choi. 2019. Abductive commonsense reasoning. *arXiv preprint arXiv:1908.05739*.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: Reasoning about Physical Commonsense in Natural Language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 7432–7439.

Antoine Bosselut and Yejin Choi. 2019. Dynamic knowledge graph construction for zero-shot commonsense question answering. *ArXiv*, abs/1911.03876.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779, Florence, Italy. Association for Computational Linguistics.

Ronan Le Bras, Swabha Swayamdipta, Chandra Bhagavatula, Rowan Zellers, Matthew E Peters, Ashish Sabharwal, and Yejin Choi. 2020. Adversarial filters of dataset biases. *arXiv preprint arXiv:2002.04108*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances*

in Neural Information Processing Systems, volume 33, pages 1877–1901. Curran Associates, Inc.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

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

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021a. Evaluating large language models trained on code.

Mingda Chen, Sam Wiseman, and Kevin Gimpel. 2021b. WikiTableT: A large-scale data-to-text dataset for generating Wikipedia article sections. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 193–209, Online. Association for Computational Linguistics.

Wenhu Chen, Ming-Wei Chang, Eva Schlinger, William Wang, and William W. Cohen. 2020a. Open question answering over tables and text.

Wenhu Chen, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. 2020b. Logical natural language generation from open-domain tables. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7929–7942, Online. Association for Computational Linguistics.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks.

Wenhu Chen, Yu Su, Xifeng Yan, and William Yang Wang. 2020c. KGPT: Knowledge-grounded pre-training for data-to-text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8635–8648, Online. Association for Computational Linguistics.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. 2020d. HybridQA: A dataset of multi-hop question answering over tabular and textual data. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1026–1036, Online. Association for Computational Linguistics.

Xilun Chen, Kushal Lakhotia, Barlas Oğuz, Anchit Gupta, Patrick Lewis, Stan Peshterliev, Yashar Mehdad, Sonal Gupta, and Wen tau Yih. 2021c. Salient phrase aware dense retrieval: Can a dense retriever imitate a sparse one?

Hao Cheng, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2020. Probabilistic assumptions matter: Improved models for distantly-supervised document-level question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5657–5667, Online. Association for Computational Linguistics.

Hao Cheng, Hao Fang, Xiaodong Liu, and Jianfeng Gao. 2022a. Task-aware specialization for efficient and robust dense retrieval for open-domain question answering.

Hao Cheng, Xiaodong Liu, Lis Pereira, Yaoliang Yu, and Jianfeng Gao. 2021a. Posterior differential regularization with f-divergence for improving model robustness. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1078–1089, Online. Association for Computational Linguistics.

Hao Cheng, Yelong Shen, Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2021b. UnitedQA: A hybrid approach for open domain question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3080–3090, Online. Association for Computational Linguistics.

Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2022b. Binding language models in symbolic languages.

Prateek Chhikara, Jiarui Zhang, Filip Ilievski, Jonathan Francis, and Kaixin Ma. 2023. Knowledge-enhanced agents for interactive text games.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz,

Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators.

Bhavana Dalvi, Lifu Huang, Niket Tandon, Wen-tau Yih, and Peter Clark. 2018. Tracking state changes in procedural text: a challenge dataset and models for process paragraph comprehension. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1595–1604, New Orleans, Louisiana. Association for Computational Linguistics.

Rajarshi Das, Manzil Zaheer, Dung Thai, Ameya Godbole, Ethan Perez, Jay Yoon Lee, Lizhen Tan, Lazaros Polymenakos, and Andrew McCallum. 2021. Case-based reasoning for natural language queries over knowledge bases. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9594–9611, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ernest Davis. 2014. *Representations of commonsense knowledge*. Morgan Kaufmann.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6491–6506, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. 2019. Cognitive graph for multi-hop reading comprehension at scale. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2694–2703, Florence, Italy. Association for Computational Linguistics.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. 2023. Palm-e: An embodied multimodal language model.

Matthew Dunn, Levent Sagun, Mike Higgins, V. Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new qa dataset augmented with context from a search engine.

Martin Fajcik, Martin Docekal, Karel Ondrej, and Pavel Smrz. 2021. R2-D2: A modular baseline for open-domain question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 854–870, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yuwei Fang, Siqi Sun, Zhe Gan, Rohit Pillai, Shuohang Wang, and Jingjing Liu. 2020. Hierarchical graph network for multi-hop question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8823–8838, Online. Association for Computational Linguistics.

William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity.

Yair Feldman and Ran El-Yaniv. 2019. Multi-hop paragraph retrieval for open-domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2296–2309, Florence, Italy. Association for Computational Linguistics.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A. Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. 2010. Building watson: An overview of the deepqa project. *AI Magazine*, 31(3):59–79.

Luyu Gao and Jamie Callan. 2021. Condenser: a pre-training architecture for dense retrieval. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 981–993, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Luyu Gao and Jamie Callan. 2022. Unsupervised corpus aware language model pre-training for dense passage retrieval. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2843–2853, Dublin, Ireland. Association for Computational Linguistics.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2022. Pal: Program-aided language models.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133, Santiago de Compostela, Spain. Association for Computational Linguistics.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Michael Glass, Mustafa Canim, Alfio Gliozzo, Saneem Chemmengath, Vishwajeet Kumar, Rishav Chakravarti, Avi Sil, Feifei Pan, Samarth Bharadwaj, and Nicolas Rodolfo Fauceglia. 2021. Capturing row and column semantics in transformer based question answering over tables. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1212–1224, Online. Association for Computational Linguistics.

Andrew Gordon, Zornitsa Kozareva, and Melissa Roemmele. 2012. SemEval-2012 task 7: Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 394–398, Montréal, Canada. Association for Computational Linguistics.

Suchin Gururangan, Margaret Li, Mike Lewis, Weijia Shi, Tim Althoff, Noah A. Smith, and Luke Zettlemoyer. 2023. Scaling expert language models with unsupervised domain discovery.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2023. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings.

Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. 2016. Deep reinforcement learning with a natural language action space. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1621–1630, Berlin, Germany. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Curran Associates, Inc.

Jonathan Herzig, Thomas Müller, Syrine Krichene, and Julian Eisenschlos. 2021. Open domain question answering over tables via dense retrieval. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 512–519, Online. Association for Computational Linguistics.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.

Jerry R Hobbs, William Croft, Todd Davies, Douglas Edwards, and Kenneth Laws. 1987. Commonsense metaphysics and lexical semantics. *Computational linguistics*, 13(3-4):241–250.

Alexander Miserlis Hoyle, Ana Marasović, and Noah A. Smith. 2021. Promoting graph awareness in linearized graph-to-text generation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 944–956, Online. Association for Computational Linguistics.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

Junjie Huang, Wanjun Zhong, Qian Liu, Ming Gong, Daxin Jiang, and Nan Duan. 2022. Mixed-modality representation learning and pre-training for joint table-and-text retrieval in openqa.

Filip Ilievski, Alessandro Oltramari, Kaixin Ma, Bin Zhang, Deborah L. McGuinness, and Pedro Szekely. 2021a. Dimensions of commonsense knowledge. *Knowledge-Based Systems*, 229:107347.

Filip Ilievski, Jay Pujara, and Hanzhi Zhang. 2021b. Story generation with commonsense knowledge graphs and axioms. In *Workshop on Commonsense Reasoning and Knowledge Bases*.

Filip Ilievski, Pedro Szekely, Jingwei Cheng, Fu Zhang, and Ehsan Qasemi. 2020. Consolidating commonsense knowledge. *arXiv preprint arXiv:2006.06114*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning.

Gautier Izacard and Edouard Grave. 2020. Distilling knowledge from reader to retriever for question answering.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, Online. Association for Computational Linguistics.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

Yifan Jiang, Filip Ilievski, and Kaixin Ma. 2023a. Transferring procedural knowledge across commonsense tasks.

Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023b. Active retrieval augmented generation.

Zhijing Jin, Qipeng Guo, Xipeng Qiu, and Zheng Zhang. 2020. GenWiki: A dataset of 1.3 million content-sharing text and graphs for unsupervised graph-to-text generation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2398–2409, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. 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)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

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

Akhil Kedia, Mohd Abbas Zaidi, and Haejun Lee. 2022. Fie: Building a global probability space by leveraging early fusion in encoder for open-domain question answering.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UNIFIEDQA: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1896–1907, Online. Association for Computational Linguistics.

Omar Khattab, Christopher Potts, and Matei Zaharia. 2021. Baleen: Robust multi-hop reasoning at scale via condensed retrieval.

Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. 2022. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp.

Jinkyu Kim, Anna Rohrbach, Trevor Darrell, John Canny, and Zeynep Akata. 2018. Textual explanations for self-driving vehicles. *Lecture Notes in Computer Science*, page 577–593.

Thomas N. Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907.

Vid Kocijan, Ana-Maria Cretu, Oana-Maria Camburu, Yordan Yordanov, and Thomas Lukasiewicz. 2019. A surprisingly robust trick for the Winograd schema challenge. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4837–4842, Florence, Italy. Association for Computational Linguistics.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners.

Klaus Krippendorff. 2004. *Content Analysis: An Introduction to Its Methodology (second edition)*. Sage Publications.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

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

Rémi Lebret, David Grangier, and Michael Auli. 2016. 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*, pages 1203–1213, Austin, Texas. Association for Computational Linguistics.

Haejun Lee, Akhil Kedia, Jongwon Lee, Ashwin Paranjape, Christopher D. Manning, and Kyoung-Gu Woo. 2021. You only need one model for open-domain question answering.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy. Association for Computational Linguistics.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding.

Yoav Levine, Barak Lenz, Or Dagan, Ori Ram, Dan Padnos, Or Sharir, Shai Shalev-Shwartz, Amnon Shashua, and Yoav Shoham. 2020. SenseBERT: Driving some sense into BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4656–4667, Online. Association for Computational Linguistics.

Patrick Lewis, Yuxiang Wu, Linqing Liu, Pasquale Minervini, Heinrich Küttler, Aleksandra Piktus, Pontus Stenetorp, and Sebastian Riedel. 2021. PAQ: 65 million probably-asked questions and

what you can do with them. *Transactions of the Association for Computational Linguistics*, 9:1098–1115.

Alexander Hanbo Li, Patrick Ng, Peng Xu, Henghui Zhu, Zhiguo Wang, and Bing Xiang. 2021a. Dual reader-parser on hybrid textual and tabular evidence for open domain question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4078–4088, Online. Association for Computational Linguistics.

Shaobo Li, Xiaoguang Li, Lifeng Shang, Xin Jiang, Qun Liu, Chengjie Sun, Zhenzhou Ji, and Bingquan Liu. 2021b. Hopretriever: Retrieve hops over wikipedia to answer complex questions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(15):13279–13287.

Tao Li and Vivek Srikumar. 2019. Augmenting neural networks with first-order logic. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 292–302, Florence, Italy. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Yikang Li, Pulkit Goel, Varsha Kuppur Rajendra, Har Simrat Singh, Jonathan Francis, Kaixin Ma, Eric Nyberg, and Alessandro Oltramari. 2020a. Lexically-constrained text generation through commonsense knowledge extraction and injection.

Zhongli Li, Wenhui Wang, Li Dong, Furu Wei, and Ke Xu. 2020b. Harvesting and refining question-answer pairs for unsupervised QA. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6719–6728, Online. Association for Computational Linguistics.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. KagNet: Knowledge-aware graph networks for commonsense reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2829–2839, Hong Kong, China. Association for Computational Linguistics.

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

Yankai Lin, Haozhe Ji, Zhiyuan Liu, and Maosong Sun. 2018. Denoising distantly supervised open-domain question answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1736–1745, Melbourne, Australia. Association for Computational Linguistics.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. K-bert: Enabling language representation with knowledge graph. In *AAAI*.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, Florence, Italy. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023. Chameleon: Plug-and-play compositional reasoning with large language models.

Shangwen Lv, Daya Guo, Jingjing Xu, Duyu Tang, Nan Duan, Ming Gong, Linjun Shou, Daxin Jiang, Guihong Cao, and Songlin Hu. 2019. Graph-based reasoning over heterogeneous external knowledge for commonsense question answering. *ArXiv*, abs/1909.05311.

Kaixin Ma, Hao Cheng, Xiaodong Liu, Eric Nyberg, and Jianfeng Gao. 2022a. Open-domain question answering via chain of reasoning over heterogeneous knowledge.

Kaixin Ma, Hao Cheng, Xiaodong Liu, Eric Nyberg, and Jianfeng Gao. 2022b. Open domain question answering with a unified knowledge interface. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1605–1620, Dublin, Ireland. Association for Computational Linguistics.

Kaixin Ma, Hao Cheng, Yu Zhang, Xiaodong Liu, Eric Nyberg, and Jianfeng Gao. 2023. Chain-of-skills: A configurable model for open-domain question answering.

Kaixin Ma, Jonathan Francis, Quanyang Lu, Eric Nyberg, and Alessandro Oltramari. 2019. Towards generalizable neuro-symbolic systems for commonsense question answering. In *Proceedings of the First Workshop on Commonsense Inference in Natural Language Processing*, pages 22–32, Hong Kong, China. Association for Computational Linguistics.

Kaixin Ma, Filip Ilievski, Jonathan Francis, Yonatan Bisk, Eric Nyberg, and Alessandro Oltramari. 2020. Knowledge-driven data construction for zero-shot evaluation in commonsense question answering.

Kaixin Ma, Filip Ilievski, Jonathan Francis, Eric Nyberg, and Alessandro Oltramari. 2022c. Coalescing global and local information for procedural text understanding. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1534–1545, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Kaixin Ma, Filip Ilievski, Jonathan Francis, Satoru Ozaki, Eric Nyberg, and Alessandro Oltramari. 2021. Exploring strategies for generalizable commonsense reasoning with pre-trained models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5474–5483, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yukun Ma, Haiyun Peng, and Erik Cambria. 2018. Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive lstm. In *AAAI*.

Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. 2022. Memory-assisted prompt editing to improve GPT-3 after deployment. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2833–2861, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback.

Aman Madaan and Yiming Yang. 2022. Flowgen: Fast and slow graph generation.

Christopher D. Manning, Kevin Clark, et al. 2020. Emergent Linguistic Structure in Artificial Neural Networks Trained by Self-Supervision. *PNAS*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems*, 35.

Todor Mihaylov and Anette Frank. 2018. Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 821–832, Melbourne, Australia. Association for Computational Linguistics.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409, Austin, Texas. Association for Computational Linguistics.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Sewon Min, Jordan Boyd-Graber, Chris Alberti, Danqi Chen, Eunsol Choi, Michael Collins, Kelvin Guu, Hannaneh Hajishirzi, Kenton Lee, Jennimaria Palomaki, Colin Raffel, Adam Roberts, Tom Kwiatkowski, Patrick Lewis, Yuxiang Wu, Heinrich Küttler, Linqing Liu, Pasquale Minervini, Pontus Stenetorp, Sebastian Riedel, Sohee Yang, Minjoon Seo, Gautier Izacard, Fabio Petroni, Lucas Hosseini, Nicola De Cao, Edouard Grave, Ikuya Yamada, Sonse Shimaoka, Masatoshi Suzuki, Shumpei Miyawaki, Shun Sato, Ryo Takahashi, Jun Suzuki, Martin Fajcik, Martin Docekal, Karel Ondrej, Pavel Smrz, Hao Cheng, Yelong Shen, Xiaodong Liu,

Pengcheng He, Weizhu Chen, Jianfeng Gao, Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Wen tau Yih. 2021. NeurIPS 2020 EfficientQA competition: Systems, analyses and lessons learned.

Sewon Min, Danqi Chen, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019. Knowledge guided text retrieval and reading for open domain question answering.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. 2021. Fast model editing at scale.

Arindam Mitra, Pratyay Banerjee, Kuntal Kumar Pal, Swaroop Mishra, and Chitta Baral. 2019. Exploring ways to incorporate additional knowledge to improve natural language commonsense question answering. *arXiv preprint arXiv:1909.08855*.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, California. Association for Computational Linguistics.

M Lynne Murphy. 2003. *Semantic relations and the lexicon: Antonymy, synonymy and other paradigms*. Cambridge University Press.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2021. Webgpt: Browser-assisted question-answering with human feedback.

Linyong Nan, Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, Yangxiaokang Liu, Nadia Irwanto, Jessica Pan, Faiaz Rahman, Ahmad Zaidi, Mutethia Mutuma, Yasin Tarabar, Ankit Gupta, Tao Yu, Yi Chern Tan, Xi Victoria Lin, Caiming Xiong, Richard Socher, and Nazneen Fatema Rajani. 2021. DART: Open-domain structured data record to text generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 432–447, Online. Association for Computational Linguistics.

Ansong Ni, Srini Iyer, Dragomir Radev, Ves Stoyanov, Wen tau Yih, Sida I. Wang, and Xi Victoria Lin. 2023. Lever: Learning to verify language-to-code generation with execution.

Yixin Nie, Songhe Wang, and Mohit Bansal. 2019. 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)*, pages 2553–2566, Hong Kong, China. Association for Computational Linguistics.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206, Saarbrücken, Germany. Association for Computational Linguistics.

Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Scott Yih. 2020. Unik-qa: Unified representations of structured and unstructured knowledge for open-domain question answering.

Barlas Oguz, Kushal Lakhotia, Anchit Gupta, Patrick Lewis, Vladimir Karpukhin, Aleksandra Piktus, Xilun Chen, Sebastian Riedel, Scott Yih, Sonal Gupta, and Yashar Mehdad. 2022. Domain-matched pre-training tasks for dense retrieval. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1524–1534, Seattle, United States. Association for Computational Linguistics.

Simon Ostermann, Michael Roth, Ashutosh Modi, Stefan Thater, and Manfred Pinkal. 2018. SemEval-2018 task 11: Machine comprehension using commonsense knowledge. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 747–757, New Orleans, Louisiana. Association for Computational Linguistics.

Xiaoman Pan, Kai Sun, Dian Yu, Heng Ji, and Dong Yu. 2019. Improving question answering with external knowledge. *CoRR*, cs.CL/1902.00993v1.

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

Bhargavi Paranjape, Scott Lundberg, Sameer Singh, Hannaneh Hajishirzi, Luke Zettlemoyer, and Marco Tulio Ribeiro. 2023. Art: Automatic multi-step reasoning and tool-use for large language models.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. ToTTo: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186, Online. Association for Computational Linguistics.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior.

Panupong Pasupat and Percy Liang. 2015. 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)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

Thomas Pellissier Tanon, Denny Vrandečić, Sebastian Schaffert, Thomas Steiner, and Lydia Pintscher. 2016. From freebase to wikidata: The great migration. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, page 1419–1428, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge enhanced contextual word representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 43–54, Hong Kong, China. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2022. Measuring and narrowing the compositionality gap in language models.

Hemant Pugaliya, James Route, Kaixin Ma, Yixuan Geng, and Eric Nyberg. 2019. Bend but don't break? multi-challenge stress test for QA models. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 125–136, Hong Kong, China. Association for Computational Linguistics.

Peng Qi, Haejun Lee, Tg Sido, and Christopher Manning. 2021. Answering open-domain questions of varying reasoning steps from text. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3599–3614, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Peng Qi, Xiaowen Lin, Leo Mehr, Zijian Wang, and Christopher D. Manning. 2019. 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)*, pages 2590–2602, Hong Kong, China. Association for Computational Linguistics.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5835–5847, Online. Association for Computational Linguistics.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer.

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

Ori Ram, Gal Shachaf, Omer Levy, Jonathan Berant, and Amir Globerson. 2022. Learning to retrieve passages without supervision. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2687–2700, Seattle, United States. Association for Computational Linguistics.

Qiu Ran, Peng Li, Weiwei Hu, and Jie Zhou. 2019. Option comparison network for multiple-choice reading comprehension. *CoRR*, abs/1903.03033.

Nils Reimers and Iryna Gurevych. 2020. Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4512–4525, Online. Association for Computational Linguistics.

Leonardo F. R. Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2020. Investigating pretrained language models for graph-to-text generation.

Kyle Richardson and Ashish Sabharwal. 2020. What does my QA model know? devising controlled probes using expert knowledge. *Transactions of the Association for Computational Linguistics*, 8:572–588.

Ellen Riloff and Michael Thelen. 2000. A rule-based question answering system for reading comprehension tests. In *ANLP-NAACL 2000 Workshop: Reading Comprehension Tests as Evaluation for Computer-Based Language Understanding Systems*.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics.

S. E. Robertson and S. Walker. 1994a. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '94, page 232–241, Berlin, Heidelberg. Springer-Verlag.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Stephen E. Robertson and Stephen Walker. 1994b. Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In *Proc. Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 232–241, Dublin, Ireland.

Devendra Singh Sachan, Mike Lewis, Mandar Joshi, Armen Aghajanyan, Wen tau Yih, Joelle Pineau, and Luke Zettlemoyer. 2022. Improving passage retrieval with zero-shot question generation.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Tali Bers, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. 2021. Multitask prompted training enables zero-shot task generalization.

Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A. Smith, and Yejin Choi. 2019a. ATOMIC: an atlas of machine commonsense for if-then reasoning. In *Proc. of AAAI*, pages 3027–3035.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019b. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4463–4473, Hong Kong, China. Association for Computational Linguistics.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools.

Christopher Sciavolino, Zexuan Zhong, Jinhyuk Lee, and Danqi Chen. 2021. Simple entity-centric questions challenge dense retrievers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6138–6148, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *ArXiv*, abs/1611.01603.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Vered Shwartz, Peter West, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Unsupervised commonsense question answering with self-talk. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4615–4629, Online. Association for Computational Linguistics.

Devendra Singh, Siva Reddy, Will Hamilton, Chris Dyer, and Dani Yogatama. 2021. End-to-end training of multi-document reader and retriever for open-domain question answering. In *Advances in Neural Information Processing Systems*, volume 34, pages 25968–25981. Curran Associates, Inc.

Push Singh, Thomas Lin, Erik T. Mueller, Grace Lim, Travell Perkins, and Wan Li Zhu. 2002. Open mind common sense: Knowledge acquisition from the general public. In *On the Move to Meaningful Internet Systems, 2002 - DOA/CoopIS/ODBASE 2002 Confederated International Conferences DOA, CoopIS and ODBASE 2002*, pages 1223–1237, Berlin, Heidelberg. Springer-Verlag.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proc. of AAAI*, AAAI'17, page 4444–4451.

Shane Storks, Qiaozi Gao, Yichi Zhang, and Joyce Chai. 2021. Tiered reasoning for intuitive physics: Toward verifiable commonsense language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4902–4918, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Haitian Sun, Tania Bedrax-Weiss, and William Cohen. 2019a. 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)*, pages 2380–2390, Hong Kong, China. Association for Computational Linguistics.

Kai Sun, Dian Yu, Jianshu Chen, Dong Yu, Yejin Choi, and Claire Cardie. 2019b. Dream: A challenge dataset and models for dialogue-based reading comprehension. *Transactions of the Association for Computational Linguistics*, 7:217–231.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Pengjie Ren, Dawei Yin, and Zhaochun Ren. 2023. Is chatgpt good at search? investigating large language models as re-ranking agent.

Alon Talmor and Jonathan Berant. 2018. 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)*, pages 641–651, New Orleans, Louisiana. Association for Computational Linguistics.

Alon Talmor and Jonathan Berant. 2019. MultiQA: An empirical investigation of generalization and transfer in reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4911–4921, Florence, Italy. Association for Computational Linguistics.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. 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)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Alon Talmor, Oyvind Tafjord, Peter Clark, Yoav Goldberg, and Jonathan Berant. 2020. Teaching pre-trained models to systematically reason over implicit knowledge. *arXiv preprint arXiv:2006.06609*.

Alexandre Tamborrino, Nicola Pellicanò, Baptiste Pannier, Pascal Voitot, and Louise Naudin. 2020. Pre-training is (almost) all you need: An application to commonsense reasoning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3878–3887, Online. Association for Computational Linguistics.

Niket Tandon, Bhavana Dalvi, Joel Grus, Wen-tau Yih, Antoine Bosselut, and Peter Clark. 2018. Reasoning about actions and state changes by injecting commonsense knowledge. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 57–66, Brussels, Belgium. Association for Computational Linguistics.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022. ScienceWorld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11279–11298, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Weiqi Wang, Tianqing Fang, Wenxuan Ding, Baixuan Xu, Xin Liu, Yangqiu Song, and Antoine Bosselut. 2023. Car: Conceptualization-augmented reasoner for zero-shot commonsense question answering.

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

Zhenyi Wang, Xiaoyang Wang, Bang An, Dong Yu, and Changyou Chen. 2020. Towards faithful neural table-to-text generation with content-matching constraints. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1072–1086, Online. Association for Computational Linguistics.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022a. Emergent abilities of large language models.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022b. Chain of thought prompting elicits reasoning in large language models.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022c. Chain-of-thought prompting elicits reasoning in large language models.

Dirk Weissenborn, Tom'avs Kovcisk'y, and Chris Dyer. 2018. Dynamic integration of background knowledge in neural nlu systems.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language*

Processing, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. 2023. Visual chatgpt: Talking, drawing and editing with visual foundation models.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Scalable zero-shot entity linking with dense entity retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6397–6407, Online. Association for Computational Linguistics.

Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008. Listwise approach to learning to rank: Theory and algorithm. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 1192–1199, New York, NY, USA. Association for Computing Machinery.

Jiangnan Xia, Chen Wu, and Ming Yan. 2019. Incorporating relation knowledge into commonsense reading comprehension with multi-task learning. In *Proc. of CIKM*, CIKM '19, page 2393–2396.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *ArXiv*, abs/1611.01604.

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

Wenhan Xiong, Xiang Lorraine Li, Srini Iyer, Jingfei Du, Patrick Lewis, William Yang Wang, Yashar Mehdad, Wen tau Yih, Sebastian Riedel, Douwe Kiela, and Barlas Oğuz. 2020b. Answering complex open-domain questions with multi-hop dense retrieval.

Li Xu, He Huang, and Jun Liu. 2021. Sutd-trafficqa: A question answering benchmark and an efficient network for video reasoning over traffic events. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yiben Yang, Chaitanya Malaviya, Jared Fernandez, Swabha Swayamdipta, Ronan Le Bras, Ji-Ping Wang, Chandra Bhagavatula, Yejin Choi, and Doug Downey. 2020. Generative data augmentation for commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1008–1025, Online. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. Cite arxiv:1906.08237Comment: Pretrained models and code are available at https://github.com/zihangdai/xlnet.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Zhi-Xiu Ye, Qian Chen, Wen Wang, and Zhen-Hua Ling. 2019. Align, mask and select: A simple method for incorporating commonsense knowledge into language representation models. *arXiv preprint arXiv:1908.06725*.

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

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8413–8426, Online. Association for Computational Linguistics.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018a. Qanet: Combining local convolution with global self-attention for reading comprehension.

Adams Wei Yu, David Dohan, Thang Luong, Rui Zhao, Kai Chen, and Quoc Le. 2018b. Qanet: Combining local convolution with global self-attention for reading comprehension.

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Wang, Zhiguo Wang, and Bing Xiang. 2022. Decaf: Joint decoding of answers and logical forms for question answering over knowledge bases.

Vicky Zayats, Kristina Toutanova, and Mari Ostendorf. 2021. Representations for question answering from documents with tables and text. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2895–2906, Online. Association for Computational Linguistics.

Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2018a. From recognition to cognition: Visual commonsense reasoning. *CoRR*, abs/1811.10830.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018b. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 93–104, Brussels, Belgium. Association for Computational Linguistics.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Jiarui Zhang, Filip Ilievski, Kaixin Ma, Jonathan Francis, and Alessandro Oltramari. 2022a. A study of zero-shot adaptation with commonsense knowledge. In *4th Conference on Automated Knowledge Base Construction*.

Jiarui Zhang, Filip Ilievski, Kaixin Ma, Aravinda Kollaa, Jonathan Francis, and Alessandro Oltramari. 2023. A study of situational reasoning for traffic understanding.

Sheng Zhang, Hao Cheng, Jianfeng Gao, and Hoifung Poon. 2022b. Optimizing bi-encoder for named entity recognition via contrastive learning.

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2018. Record: Bridging the gap between human and machine commonsense reading comprehension. *CoRR*, abs/1810.12885.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Xinyu Zhang, Ke Zhan, Enrui Hu, Chengzhen Fu, Lan Luo, Hao Jiang, Yantao Jia, Fan Yu, Zhicheng Dou, Zhao Cao, and Lei Chen. 2021a. Answer complex questions: Path ranker is all you need. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 449–458, New York, NY, USA. Association for Computing Machinery.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J. Smola, and Le Song. 2017. Variational reasoning for question answering with knowledge graph.

Yuyu Zhang, Ping Nie, Arun Ramamurthy, and Le Song. 2021b. Answering any-hop open-domain questions with iterative document reranking. *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, Florence, Italy. Association for Computational Linguistics.

Chen Zhao, Chenyan Xiong, Jordan Boyd-Graber, and Hal Daumé III. 2021. Multi-step reasoning over unstructured text with beam dense retrieval. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4635–4641, Online. Association for Computational Linguistics.

Chen Zhao, Chenyan Xiong, Corby Rosset, Xia Song, Paul Bennett, and Saurabh Tiwary. 2020. Transformer-xh: Multi-evidence reasoning with extra hop attention. In *International Conference on Learning Representations*.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer, and Steffen Eger. 2019. MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 563–578, Hong Kong, China. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning.

Wanjun Zhong, Junjie Huang, Qian Liu, Ming Zhou, Jiahai Wang, Jian Yin, and Nan Duan. 2022. Reasoning over hybrid chain for table-and-text open domain qa.

Wanjun Zhong, Duyu Tang, Nan Duan, Ming Zhou, Jiahai Wang, and Jian Yin. 2018. Improving question answering by commonsense-based pre-training. *CoRR*, abs/1809.03568.

Wanjun Zhong, Duyu Tang, Nan Duan, Ming Zhou, Jiahai Wang, and Jian Yin. 2019. Improving question answering by commonsense-based pre-training.

Jiawei Zhou, Xiaoguang Li, Lifeng Shang, Lan Luo, Ke Zhan, Enrui Hu, Xinyu Zhang, Hao Jiang, Zhao Cao, Fan Yu, Xin Jiang, Qun Liu, and Lei Chen. 2022. Hyperlink-induced pre-training for passage retrieval in open-domain question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7135–7146, Dublin, Ireland. Association for Computational Linguistics.

Xuhui Zhou, Y. Zhang, Leyang Cui, and Dandan Huang. 2020. Evaluating commonsense in pre-trained language models. In *AAAI*.

Yunchang Zhu, Liang Pang, Yanyan Lan, Huawei Shen, and Xueqi Cheng. 2021. Adaptive information seeking for open-domain question answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3615–3626, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Simiao Zuo, Xiaodong Liu, Jian Jiao, Young Jin Kim, Hany Hassan, Ruofei Zhang, Tuo Zhao, and Jianfeng Gao. 2021. Taming sparsely activated transformer with stochastic experts.

Chapter A

Appendix for Chapter 3

A.1 Hyperparameters

A.1.1 Model training

In all experiments that involve training, we used learning rate $1e^{-5}$, batch size 32, max sequence length 128, weight decay 0.01, adam epsilon $1e^{-6}$, $\beta_1 = 0.9$, $\beta_2 = 0.98$ and warm-up proportion 0.05, margin 1.0. For MLM training with RoBERTa, We only mask non-stop words in head or tail and we set the masking probability as 0.5 for `ATOMIC` and 0.3 for `CWWV`.

A.1.2 Tuned parameters

We also tested smaller margins for training (0.2, 0.5) and we observe that margin 1.0 works slightly better.

AFLite

In our experiments, we set N=64, τ=0.75, k_1=1/50 of $|Trn|$, k_2=1/50 of $|Dev|$ and O = 1/5 of $|Trn|$.

A.2 AFLite algorithm

Input: Trn, Dev, ensemble size N, cutoff sizes k_1,k_2, threshold τ, target size O

while $|Trn| > O$ **do**
  for *s in Trn+Dev* **do**
    Initialize Prediction P(s) = $\emptyset$
  end
  for *i=1, ... N* **do**
    Random Partition Trn into U, V s.t $|U|$ = O
    Train a linear classifier F on U
    for *s in V+Dev* **do**
      Add F(s) in P(s)
    end
  end
  for *s in Trn+Dev* **do**
    Acc(s) = $\frac{|p \in P(s) s.t. p = y|}{|P(s)|}$
  end
  Select top-k_1 samples S_1 in Trn s.t. Acc(s) $> \tau$
  Select top-k_2 samples S_2 in Dev s.t. Acc(s) $> \tau$
  Trn = Trn - S_1
  Dev = Dev - S_2
  if $|S_1| < k_1$ **then** break;
end

Output: filtered sets Trn' and Dev'

Algorithm 1: AFLite

Chapter B

Appendix for Chapter 4

B.1 Document Index Statistics

To be consistent with text passages, we also cut tables and KB sub-graphs (raw or verbalized) into chunks that has about 100 words. Hence the verbalized knowledge will have larger index size than raw format (see Table B.1).

B.2 Training Details

To train the retriever to better handle knowledge from tables and KB, we create additional training data from `NQ-Table-Q` and `WebQ-KB-Q`. Given a (question, answer, gold table) from `NQ-Table-Q`, we create a positive passage by concatenating rows containing the answer. Then we randomly sample and concatenate other rows in the table if the passage has less than 100 words. To find negative passages for training, we build a index consists of all the tables and use BM25 to retrieve relevant tables. Ones that do not contain the answer are considered as negative tables. Then we sample rows from the table to build negative passages. For the raw tables, the process is the same except that we also concatenate headers in the beginning to build positive and negative passages. We combine NQ training data with this set to train DPR.

For `WebQ-KB-Q`, we use the verbalized gold sub-graphs as positive passages. For the raw format, this is replaced by flattening the gold sub-graph. Then we build an index with all documents in `WD-graphs` and the top ranked documents by BM25 that do not contain the answer are treated as negatives. Here the documents refer to concatenated triples set for raw setting and sentences produced by the generator in verbalized setting. Additionally, we search through answer entities and their neighbors in the graph to find documents that has word overlap with the question. Then we build training instances in a similar fashion.

As pointed by previous work Oguz et al. (2020), mining harder negative passages using DPR and iterative training leads to better performance. We also adopted this approach in our experiments. After the first DPR is trained, we used it to retrieve passages from a joint index of text+`structured knowledge`. Then the negative passages are paired with the positive passages from the first round to build new sets of training data. Then we train a second DPR using the iteration1 data combined with the new training sets.

Source	Raw	Verbalized
Text	21M	-
OTT-tables	4.0M	6.3M
NQ-tables	446K	572K
WD-graphs	5.7M	5.8M

Table B.1: Statistics of Document Index

Source	Format	R20	R100	EM
text	-	81.3	87.3	51.8
+NQ-tables	raw	86.0	91.2	54.8
+NQ-tables	V	86.2	91.0	54.2
+All-tables	raw	86.9	**91.9**	54.7
+All-tables	V	**87.0**	91.7	**55.2**
text	-	73.2	81.4	48.0
+WD-graphs-WebQ	raw	**80.2**	**85.8**	51.5
+WD-graphs-WebQ	V	79.7	85.3	**52.6**
+WD-graphs	raw	78.8	85.1	51.4
+WD-graphs	V	78.5	85.5	52.0

Table B.2: Impact of document index size over separately trained retriever-reader models (Top for NQ and bottom for WebQ). All metrics are computed on the corresponding test set.

For retriver training, we follow the experiment set-up as specified by Karpukhin et al. (2020). Specifically, we use the Adam optimizer and a per-gpu batch size of 32 for NQ and 24 for WebQ, respectively. All trainings are done with a fixed learning rate of $2e-5$ and 40 epochs using 8 V100 GPUs. We select the best model based on the retrieval accuracy on the corresponding dev set.

For reader training, we follow the experiment set-up as described in Cheng et al. (2021b). Specifically, we use the Adam optimizer and a batch size of 16 for NQ and 8 for WebQ, respectively. We use 16 and 8 V100 GPUs for NQ and WebQ respectively. We select the learning rate in $\{3e-5, 5e-5\}$ and number of training epochs in $\{6, 8\}$. The best model is selected based on EM on the corresponding dev set. All of our reported results are obtained from a single run.

Regarding the number of parameters in the model, our verbalizer is based on T5-large, which has 770M parameters. Our retriever is a bi-encoder model based on bert-base, which has 220M parameters. Our reader model is based on ELECTRA-large, which has 330M parameters.

B.3 Impact of Document Index Size

We report the test set results of models trained with different document index in Table B.2 (corresponding to Table 4.3). Overall, we observe similar trends. For NQ, the model benefits more from a larger document index while for WebQ the restricted setting yield better performance.

Source	R20	R100	EM
KELM	**83.1**	**86.7**	55.1
WD-graphs (Ours)	82.8	**86.7**	**55.4**

Table B.3: Dev set results of models trained on WebQ with verbalized WD-graph and KELM

	V-correct	V-error
Raw-correct	1750	223
Raw-error	242	1395

Table B.4: Error matrix of UDT-QA trained with text+All-tables in raw and verbalized format

B.4 Comparison betweeh Our Verbalizer and KELM-verbalizer

We report the dev set results of WebQ models trained with our verbalized WD-graphs in comparison with KELM in Table B.3 (corresponding to Table 4.5).

B.5 Case Study on Raw vs Verbalized Tables

For manual analysis of verbalized and raw tables, we start by computing the error matrix of the NQ models trained with text+All-tables in both format, as shown in Table B.4. We then manually annotated 100 examples where only 1 format of knowledge successfully answered the question (50 for each format), and we select examples where at least 1 table chunk is marked as positive by the retriever. Out of 50 examples where verbalized tables contain the answer span, 40 of them are true positives that provide direct evidence to the questions. In 35 out of 40 questions, the retriever for the raw model actually find the same table/chunks that provide the answer. However, the model failed to extract answer for those cases and we think it's mainly because the raw format of the noisy tables can be hard for the model to reason over, as discussed in section 4.7.

We then looked at the other group of 50 questions (raw format). 37 of them are true positives that contain direct evidence. Then in 30 out of 37 questions, the verbalized retriever is able to find the corresponding verbalized table/chunks that also contain the answer. The remaining cases are all due to retriever failed to find the true positive table chunks. In these 30 cases, the most noticeable pattern is that the model is able to leverage structural shortcut to arrive at the answer, suggesting the limitation of verbalized tables.

B.6 Data-to-text Examples

In the top half of Table B.5 we show examples from DART that are filtered out by our method, i.e. low ROUGE scores between input and target. In the first example, information from 2 cells are completely omitted from the target. The model may learn to omit information from this kind of examples, which is problematic when we consider QA as our downstream task. Our filtering method is also able to prune noisy examples, as shown in row 2&3, where there is little correspondence

between input and target. In row 4, we show an example where the target contains the information not exist in the input. This kind of examples may teach the model to hallucinate which is also an unwanted behavior, hence they are also filtered out.

In the bottom half of Table B.5 we show examples from ID−T set, i.e. good quality input-output pairs produced by the verbalizer trained on T−F set, when applied to our table sets. These examples often have longer inputs and/or larger table cells and the information coverage rate in target is very high. By combining ID−T set with T−F set to train a new verbalizer, the model can more effectively learn to generate faithful description of the input.

Finally, in Table B.6 we show examples of question-answer pairs along with their associated gold tables(rows). For each raw input, we show its corresponding output generated by the verbalizer trained on T and the verbalizer trained on T−F+ID−T. We can see that the direction evidence to the answer is present in one of the raw table cells, however in all 3 cases the verbalizer(T) failed to generate that piece of information. On the other hand, the verbalizer(T−F+ID−T) faithfully described all information presented in the input, showing the effectiveness of our proposed method.

Input	<H> [title] <T> Conde McCullough <H> bridge name <T> oregon city bridge <H> carries <T> oregon route 43 <H> **year completed** <T> **1922** <H> **total length** <T> **745 feet 227 m** <H> location <T> oregon city oregon
Target	conde mccullough helped build the oregon city bridge, which is part of oregon rt. 43.
Input	<H> [title] <T> **Yury Aleksandrovich Golovkin** <H> russian state <T> russian empire <H> **date from** <T> **9 november 1818** <H> **date until** <T> **16 september 1822**
Target	the *first ambassador of russia to austria* was *prince dmitry mikhailovich golitsyn*.
Input	<H> [title] <T> 43 <H> **competition** <T> **afc champions league**
Target	the club was *officially founded as lucky-goldstar football club in 1983, by the lucky-goldstar group. fc seoul have won six league titles, two fa cups, two league cups and one super cup.*
Input	<H> [title] <T> Ana Maria Beldana Ferriera <H> hometown <T> santiago de los caballeros
Target	ana maría beldaña ferriera, from santiago de los caballeros, was a *contestant in the miss dominican republic 2002 contest.*
Input	<H> [title] <T> Meet the Fockers <H> edited by <T> jon poll lee haxall alan baumgarten <H> production company <T> tribeca productions everyman pictures <H> distributed by <T> universal pictures (north america) dreamworks pictures (international)
Target	meet the fockers was edited by jon poll, lee haxall, alan baumgarten and distributed by universal pictures (north america) dreamworks pictures (international). the production company was tribeca productions.
Input	<H> [title] <T> Lamar Hunt U.S. Open Cup <H> season <T> 2010 <H> player <T> paulo jr. nate jaqua <H> team <T> miami fc seattle sounders fc <H> goals <T> 5
Target	paulo jr. nate jaqua scored 5 goals for miami fc seattle sounders fc in the 2010 lamar hunt u.s. open cup.

Table B.5: Top: examples from DART that are filtered out by our method, the **bold** cells are omitted information from target, and *italic text* from target are likely to bias the model towards hallucination. Bottom: examples from (ID−T), which is generated by our 1st iteration verbalizer

Question	how many episodes in season 7 walking dead
Answer	16
Input	<H> [title] <T> The Walking Dead (season 7) <H> country of origin <T> united states <H> **no. of episodes** <T> **16** <H> the walking dead (season 7) <T> release <H> original network <T> amc
Verbalizer (`T`)	the original network for the walking dead (season 7) is amc. the country of origin for the walking dead (season 7) is united states.
Verbalizer (`T-F+ID-T`)	the original network of the walking dead (season 7) is amc and the country of origin is united states. **the walking dead (season 7) has 16 episodes.**
Question	when did nigeria adopt the presidential system of government
Answer	1963
Input	<H> [title] <T> Federal government of Nigeria <H> federal government of nigeria <T> coat of arms of nigeria <H> **formation** <T> **1963; 55 years ago** <H> founding document <T> constitution of nigeria
Verbalizer (`T`)	the constitution of nigeria is the founding document of the federal government of nigeria which was formed 55 years ago. the federal government of nigeria has the coat of arms of nigeria.
Verbalizer (`T-F+ID-T`)	the constitution of nigeria is the founding document of the federal government of nigeria which was formed **in 1963; 55 years ago**. the federal government of nigeria has the coat of arms of nigeria.
Question	what year did they stop making the saturn vue
Answer	2009
Input	<H> [title] <T> Saturn Vue <H> saturn vue <T> overview <H> manufacturer <T> saturn corporation (2002-2007) opel (general motors) (2008-2010) <H> **production** <T> **2001–2009** <H> model years <T> 2002–2010 <H> saturn vue <T> body and chassis
Verbalizer (`T`)	saturn vue's body and chassis were manufactured by saturn corporation (2002-2007) and opel (general motors) (2008-2010) during the model years 2002–2010.
Verbalizer (`T-F+ID-T`)	saturn corporation (2002-2007) opel (general motors) (2008-2010) **manufactured the saturn vue from 2001–2009** and model years 2002–2010. the saturn vue has a body and chassis.

Table B.6: Examples of verbalized table(rows) generated by different verbalizer, where the direct evidences to the answer are marked in **bold**

Chapter C

Appendix for Chapter 5

C.1 Chaining Strategy

At the chaining step, a table t can appear multiple times in this evidence chain list, if there are multiple passages linked to it. Also, a single passage can be linked to multiple tables, resulting in potential redundancy. Moreover, concatenating the table with the passage may make the sequence too long for the reader to handle. To reduce the duplication and sequence length, we adopt the following strategy: for a document in the first hop, we only add it to the Top-K list if it has not been included before; similarly, for a linked passage in the second hop, we only added it to the Top-K if it has not been added before, and we only concatenate the table header and its entity mention's row to it, and then include it as a separate document. We iterate over the reranked list of chains until reaching K chains.

C.2 Dataset Statistics

The dataset statistics are shown in Table C.1. For both datasets, we follow Ma et al. (2022b) to split tables into chunks of approximately 100 words, which results in about two chunks for each table. Thus the size of the table index also increased. (840K table chunks and 4.4M table chunks for OTT-QA and NQ, respectively)

C.3 Training Details

We run all of our experiments once due to computation constraints, and we largely follow previous works' hyper-parameters settings.

C.3.1 Linker Training

Only tables in OTT-QA train and dev set contain hyperlinks (8K for train, 0.8K for dev), and we only used these tables [1] to train our linker model (§5.3.1).

[1] `https://github.com/wenhuchen/OTT-QA`

Dataset	OTT-QA	NQ
Train	41,469	79,168
Dev	2,214	8,757
Test	2,158	3,610
# Tables	400K	2.4M
# Passages	6.1M	21M

Table C.1: Statistics of Datasets

To mine hard negatives for linker training, we adopted two strategies using BM25. For the first strategy, we used the entity mentions in the table are queries, and retrieve from an index of passage titles only. In the second one, we used the entity mentions along with the table title as queries, and retrieve from an index of passage titles concatenated with the first sentence from their corresponding pages. We observe very different negatives from these two strategies and we used negatives from both to train our linker model.

During training, we use randomly sample one hard negative for every (mention, positive passage) pair, and also use in-batch negatives to compute the contrastive loss. We train entity mention proposal model for 40 epochs and table entity linking model for 100 epochs, both with batch size 64, learning rate 2e-5, and linear warm-up scheduler.

C.3.2 Retriever Training

For training data, we directly used the data released by Karpukhin et al. (2020) for NQ [2] and Ma et al. (2022b) for NQ-table-answerable set [3]. Note that the NQ-table-answerable data is considered part of NQ in all settings. For OTT-QA, each question is annotated with one or more positive tables and we use BM25 to mine hard negatives from an index of all OTT-QA tables. Following previous work Karpukhin et al. (2020), we train our retrievers (for both single-dataset and joint setting) for 40 epochs, with batch size 128, learning rate 2e-5 and select the best checkpoint using validation average rank on dev set.

C.3.3 Chainer Inference

We set $\alpha = 16$ and $\beta = 9$ in Equation 5.8 for OTT-QA, and $\alpha = 10$ and $\beta = 12$ for NQ. These hyper-parameters are selected based on the respective dev set's answer recall performance. We did a grid search over integer values in the range [1, 20] for both parameters.

C.3.4 Reader Training

Following Izacard and Grave (2021), we train our reader models for 15000 steps, with batch size 64, learning rate 5e-5.

[2] https://github.com/facebookresearch/DPR

[3] https://github.com/Mayer123/UDT-QA

	R@20	R@100	AR@20	AR@50
OTT-QA Retriever	82.7	92.5	31.2	37.5
+Linker&Chainer	84.1	91.2	74.4	83.5
Retrieve full index	82.3	91.8	32.7	40.2

Table C.2: Evidence Recall of OTT-QA single retriever on OTT-QA Dev set, where R@K evaluates gold table chunk recall and AR@K evaluates answer recall

	Dev		Test	
	AR@20	AR@50	AR@20	AR@50
Joint retriever	82.3	87.0	83.6	88.2
+Linker&Chainer	85.7	88.1	86.8	89.4
NQ Retriever	83.2	87.4	84.1	88.4
+Linker&Chainer	85.9	88.4	86.5	89.4

Table C.3: Answer Recall on NQ task

C.4 Results and Discussion

C.4.1 Impact of Linker & Chainer

Here we evaluate the evidence recall performance of the OTT-QA single retriever in Table C.2. We can see that compared to results in Table 5.5, the performance of the single dataset retriever is quite similar for all metrics, suggesting that a joint model is sufficient to learn multiple tasks under our framework. We conduct the same evaluation on NQ in Table C.3. With the addition of Linker and Chainer, we see that the answer recall improved over the retriever-only setting. We also observe that the results for NQ-single retriever are quite similar to joint retriever, as seen on OTT-QA.

Chapter D

Appendix for Chapter 6

D.1 Inference Pipeline

At inference time, our model utilizes the retrieving skill or the linking skill or both in parallel to gather evidence at every reasoning step. When both skills are used, one problem is that the scores associated with the evidence found by different skills are not aligned, *i.e.,* naively sorting the retrieved documents and linked documents together may cause one pool of documents to dominate over the other. Thus we propose to align the linking scores based on the same step retrieval score:

$$\text{ls}_i = \text{ls}_i / \max(\{\text{ls}\} \cup \{\text{rs}\}) \times \max(\{\text{rs}\}), \tag{D.1}$$

where ls_i represents the linking score of the document i and $\{\text{ls}\}$, $\{\text{rs}\}$ represent the set of linking scores and retrieving scores for top-K documents from each skill. Effectively, if the raw linking score is larger than the retrieving score, we would align the top-1 document from each set. On the other hand, if the raw linking score is smaller, it would not get scaled. The reason is that certain common entities may also be detected and linked by our model *e.g.,* United States, but they usually do not contribute to the answer reasoning, thus we do not want to encourage their presence.

In the case of a document being discovered by both skills, we promote its ranking in the final list. To do so, we take the max of the individual score (after alignment) and then multiply by a coefficient α, which is a hyper-parameter.

$$s_i = \alpha \max(\text{ls}_i, \text{rs}_i). \tag{D.2}$$

Finally, we use the reranking skill to compute a new set of scores for the merged evidence set, and then sort the documents using the combination of retrieving/linking score and reranking score:

$$s_i + \beta \, \text{rankscore}_i. \tag{D.3}$$

β is another hyper-parameter. For multi-hop questions, the same scoring process is conducted for the second-hop evidence documents and then the two-hop scores are aggregated to sort the reasoning chains. The inference pipeline is also illustrated in Figure D.1.

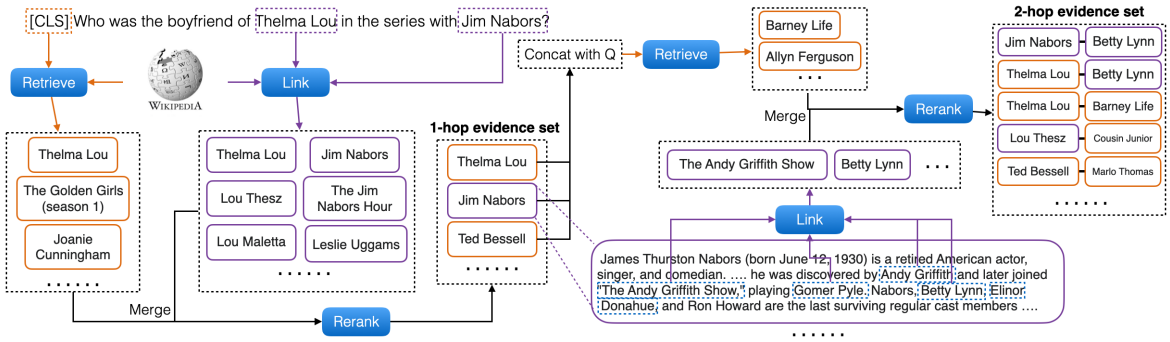

Figure D.1: The reasoning pipeline of `Chain-of-Skills (COS)`. Given a question, `COS` first identifies salient spans in the question, then the retrieving and linking skills are both used to find first-hop evidence, using the `[CLS]` token and entity mention representation respectively. Then we merge all the evidence through score alignment and the reranking skill. For top-ranked evidence documents, we concatenate each of them with the question and perform another round of retrieving and linking. Then the second hop evidence are merged and reranked in the same fashion. Finally, the reasoning paths are sorted based on both hops' scores

D.2 Experimental Details

D.2.1 Data Statistics

The detailed data statistics are shown in Table D.1.

Pretraining We follow Izacard et al. (2021) and Wu et al. (2020) to construct examples for single retrieval and entity linking, respectively. For single retrieval, a pair of randomly cropped views of a passage is treated as a positive example. Similar to Spider (Ram et al., 2022), we also use the processed DPR passage corpus based on the English Wikipedia dump from 2018/12/20. For entity linking, we directly use the preprocessed data released by BLINK (Wu et al., 2020) based on the English Wikipedia dump from 2019/08/01. For expanded query retrieval, we construct the pseudo query using a short text snippet with the first passage from the same page, and we treat the first passage from linked pages as the target. As no hyperlink information is preserved for the DPR passage corpus, we use the English Wikipedia dump from 2022/06/01 for data construction. In each Wikipedia page, we randomly sample 30 passages with hyperlinks. (If there are less than 30 passages with hyperlinks, we take all of them.) Each sampled passage, together with the first passage of the page, form a pseudo query. Then, in each sampled passage, we randomly pick an anchor entity and take the first passage of its associated Wikipedia page as the target. To avoid redundancy, if an anchor entity has been used 10 times in a source page, we no longer pick it for the given source. If the query and the target together exceed 512 tokens, we will truncate the longer of the two by randomly dropping its first token or its last token.

Finetuning For NQ, we adopted the retriever training data released by Ma et al. (2022b) and further used them for the reranking skill. Note that data from Ma et al. (2022b) also contains table-answerable questions in NQ, and we simply merged the corresponding training splits with the text-based training split. That's why the number of examples in the last column is greater than

the number of questions in the training set.

For HotpotQA, we adopted single retrieval and expanded query retrieval data released by Xiong et al. (2020b). For question entity linking data, we heuristically matched the entity spans in the question with the gold passages' title to construct positive pairs, and we use the same set of negative passages as in single retrieval. For passage entity linking, we collected all unique gold passages in the training set and their corresponding hyperlinks for building positives and mined negatives using BM25. Finally, the reranking data is the same as single retrieval.

For OTT-QA, we adopt the single retrieval and table entity linking data released by Ma et al. (2022a). For expanded query retrieval, we concatenate the question with the table title, header, and row that links to the answer-containing passage as the query, and the corresponding passage is treated as a positive target. The negatives are mined with BM25. Finally, reranking data is the same copy as in single retrieval except that we further break down tables into rows and train the model to rank rows. This is because we want to make the reranking and expanded query retrieval more compatible.

Since iterative training is shown to be an effective strategy by previous works (Xiong et al., 2020a; Ma et al., 2022b), we further mined harder negatives for HotpotQA and OTT-QA skill training data. Specifically, we train models using the same configuration as in pretraining (four task-specific experts, with no reranking data or span proposal data) for HotpotQA and OTT-QA respectively (models are initialized from BERT-based-uncased). Then we minded harder negatives for each of the data types using the converged model. The reranking and the entity span proposal skills are excluded in this round because the reranking can already benefit from harder negative for single retrieval (as two skills share the same data) and the entity span proposal does not need to search through a large index. Finally, the data splits coupled with harder negatives are used to train our main `Chain-of-Skills (COS)` and conduct ablation studies.

D.2.2 Training Details

Pretraining Similar to Contriever (Izacard et al., 2021), we adopt a continual pretraining setup based on the uncased BERT-base architecture, but our model is initialized from the Contriever weights. We train the model for 20 epochs with the batch size of 1024 and the max sequence length of 256. Here, we only use in-batch negatives for contrastive learning. The model is optimized using Adam with the initial learning rate of 1e-4. The final checkpoint is used for fine-tuning later.

Finetuning When initializing from pretrained `COS`, the weights mapping for the first 5 experts are illustrated in Figure 6.3 and the last expert is initialized from BERT-base-uncased. For all experiments, we train models for 40 epochs with the batch size of 192, the learning rate of 2e-5, and the max sequence length of 256. During training, each batch only contains training data for one of the skills from one dataset, thus the model can effectively benefit from the in-batch negatives. To train the entity span proposal skill, we use the same data as entity linking. In particular, we route the data to span proposal experts 20% of the time otherwise the data go through entity linking experts.

Dataset	Train	Dev	Test	Skill Training Data	# Examples
Pretraining	-	-	-	single retrieval expanded query retrieval passage entity linking	6M 6M 9M
NQ	79,168	8,757	3,610	single retrieval reranking	86,252 86,252
HotpotQA	90,447	7,405	7,405	single retrieval expanded query retrieval question entity linking passage entity linking reranking	90,447 90,447 80,872 104,335 90,447
OTT-QA	41,469	2,214	2,158	single retrieval expanded query retrieval table entity linking reranking	41,469 31,638 19,764 41,479
EntityQuestions	-	22,068	22,075	-	-
WebQ	-	-	2,032	-	-
SQuAD	-	-	10,570	-	-

Table D.1: Statistics of datasets used in our experiments, columns 2-4 represent the number of questions in each split. The last two columns contain the type of training data and the corresponding number of instances

	#Params	**EM**
FiD (Izacard and Grave, 2021)	770M	51.4
UnitedQA-E (Cheng et al., 2021b)	330M	51.8
FiD-KD (Izacard and Grave, 2020)	770M	54.4
EMDR^2 (Singh et al., 2021)	440M	52.5
YONO (Lee et al., 2021)	440M	53.2
UnitedQA (Cheng et al., 2021b)	1.87B	54.7
R2-D2 (Fajcik et al., 2021)	1.29B	55.9
FiE (Kedia et al., 2022)	330M	**58.4**
FiE (ours implementation)	330M	56.3
COS + FiE	330M	56.4

Table D.2: End-to-end QA Exact Match score on NQ

D.2.3 Inference Details

Zero-shot-evaluation We directly use the single retrieval skill to find the top100 documents and compute the results in Table 6.1.

Supervised and Cross-dataset For NQ, EntityQuestions and SQuAD, the reasoning path has a length of 1, *i.e.,* only single passages. We use both single retrieval and linking skills to find a total of top 1000 passages first, and then reduce the set to top 100 using the reranking skill.

Both HotpotQA and OTT-QA have reasoning paths with max length 2. For OTT-QA, we first find top 100 tables using the single retrieval skill following Ma et al. (2022a). Then we break down tables into rows and use the reranking skill to keep only top 200 rows. Then for each row, expanded query retrieval and linking skills are used to find the second-hop passages, where we keep top 10 passages from every expanded query retrieval and top 1 passage from every linked entity. Finally, we apply the same heuristics, as done in Ma et al. (2022a) to construct the final top 100 evidence chains.

For HotpotQA, single retrieval and linking are used jointly to find the first-hop passages where we keep top 200 passages from single retrieval and top 5 passage from each linked question entity. The combined set is then reranked to keep the top 30 first-hop passages. Then expanded query retrieval and passage entity linking are applied to these 30 passages, where we keep top 50 passages from expanded query retrieval and top 2 passages from every linked passage entity. Next, another round of reranking is performed on the newly collected passages and then we sort the evidence passage chains based on the final aggregated score and keep top 100 chains. Since all of the baselines on HotpotQA adopt a large passage path reranker, we also trained such a model following Zhu et al. (2021) (discussed in Appendix D.3) to rank the top 100 passage chains to get the top 1 prediction.

The hyperparameters for OTT-QA and HotpotQA inference are selected such that the total number of evidence chains are comparable to previous works (Ma et al., 2022a; Xiong et al., 2020b).

	Dev						Test					
	Ans		Sup		Joint		Ans		Sup		Joint	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
MUPPET (Feldman and El-Yaniv, 2019)	31.1	40.4	17.0	47.7	11.8	27.6	30.6	40.3	16.7	47.3	10.9	27.0
CogQA (Ding et al., 2019)	37.6	49.4	23.1	58.5	12.2	35.3	37.1	48.9	22.8	57.7	12.4	34.9
GoldEn Retriever (Qi et al., 2019)	-	-	-	-	-	-	37.9	49.8	30.7	64.6	18.0	39.1
Semantic Retrieval (Nie et al., 2019)	46.5	58.8	39.9	71.5	26.6	49.2	45.3	57.3	38.7	70.8	25.1	47.6
Transformer-XH (Zhao et al., 2020)	54.0	66.2	41.7	72.1	27.7	52.9	51.6	64.1	40.9	71.4	26.1	51.3
HGN (Fang et al., 2020)	-	-	-	-	-	-	59.7	71.4	51.0	77.4	37.9	62.3
GRR (Asai et al., 2020)	60.5	73.3	49.2	76.1	35.8	61.4	60.0	73.0	49.1	76.4	35.4	61.2
DDRQA (Zhang et al., 2021b)	62.9	76.9	51.3	79.1	-	-	62.5	75.9	51.0	78.9	36.0	63.9
MDR (Xiong et al., 2020b)	62.3	75.1	56.5	79.4	42.1	66.3	62.3	75.3	57.5	80.9	41.8	66.6
IRRR+ (Qi et al., 2021)	-	-	-	-	-	-	66.3	79.9	57.2	82.6	43.1	69.8
HopRetriever-plus (Li et al., 2021b)	66.6	79.2	56.0	81.8	42.0	69.0	64.8	77.8	56.1	81.8	41.0	67.8
TPRR (Zhang et al., 2021a)	67.3	80.1	60.2	84.5	45.3	71.4	67.0	79.5	59.4	84.3	44.4	70.8
AISO (Zhu et al., 2021)	68.1	80.9	**61.5**	**86.5**	45.9	**72.5**	**67.5**	**80.5**	61.2	**86.0**	44.9	**72.0**
COS	**68.2**	**81.0**	61.1	85.3	**46.4**	72.3	67.4	80.1	**61.3**	85.3	**45.7**	71.7

Table D.3: End-to-end QA results on Hotpot-QA.

D.3 Question Answering Results

D.3.1 Training Details

We follow descriptions in Kedia et al. (2022) for re-implementation of FiE model and the model is initialized from Electra-large (Clark et al., 2020). For NQ, we train the model for 5,000 steps with the effective batch size of 64, the learning rate of 5e-5, the layer-wise learning rate decay of 0.9, the max answer length of 15, the max question length of 28, the max sequence length of 250, and 10 global tokens. Note that although Kedia et al. (2022) reports that training with 15,000 steps leads to better performance, we actually found it to be the same as 5,000 steps. Thus we train with fewer steps to save computation. For OTT-QA, we used the same set-up of hyperparameters except that the max sequence length is changed to 500.

For HotpotQA path reranker and reader, we prepare the input sequence as follows: "[CLS]Q [SEP]yes no [P] P1 [P] P2 [SEP]", where [P] is a special token to denotes the start of a passage. Then the input sequence is encoded by the model and we extract passage start tokens representations $p_1, ...p_m$ and averaged sentence embeddings for every sentence in the input $s_1, ...s_n$ to represent passages and sentences respectively. The path reranker is trained with three objectives: passage ranking, supporting sentence prediction and answer span extraction, as we found the latter two objectives also aid the passage ranking training. For answer extraction, the model is trained to predict the start and end token indices as commonly done in recent literature (Xiong et al., 2020b; Zhu et al., 2021). For both passage ranking and supporting sentence prediction, the model is trained with the ListMLE loss (Xia et al., 2008). In particular, every positive passage in the sequence is assigned a label of 1, and every negative passage is assigned 0. To learn a dynamic threshold, we also use the [CLS]token p_0 to represent a pseudo passage and assign a label of 0.5. Finally, the

loss is computed as follows:

$$L_{\text{p}} = -\sum_{i=0}^{m} \log \frac{\exp(p_i W_p)}{\sum_{p' \in \mathcal{P} \cup \{p_i\}} \exp(p' W_p)}. \tag{D.4}$$

where $\mathcal{P}$ contains all passages representations that have labels smaller than p_i. $W_p \in \mathrm{R}^d$ are learnable weights and d is the hidden size. In other words, the model learns to assign scores such that positive passages $>$ thresholds $>$ negative passages. The supporting sentence prediction is also trained using Equation D.4. Overall, use the following loss weighting:

$$L_{\text{path}} = L_p + L_a + 0.5 \times L_s \tag{D.5}$$

where L_a is the answer extraction loss and L_s is the supporting sentence prediction loss.

During training, we sample 0-2 positive passages and 0-2 negative passages from the top 100 chains returned by `COS`, and the model encodes at most 3 passages, *i.e.,* the passage chain structure is not preserved and the passages are sampled independently. We train the model for 20,000 steps with the batch size of 128, the learning rate of 5e-5, the layer-wise learning rate decay of 0.9, the max answer length of 30, the max question length of 64, and the max sequence length of 512. For inference, the model ranks top 100 passage chains with structure preserved. We sum the scores of the two passages in every chain and subtract the dynamic threshold score and sort the chains based on this final score.

Next, we train a reader model that only learns answer extraction and supporting sentence prediction. We only train the model using the two gold passages with the following loss weighting.

$$L_{\text{reader}} = L_a + 0.5 \times L_s \tag{D.6}$$

The model uses the same set of hyperparameters as the path reranker except that the batch size is reduced to 32. At inference time, the model directly read the top 1 prediction returned by the path reranker. Both models here are initialized from Electra-large.

D.3.2 Results

The NQ results are presented in Table D.2. Overall, our model achieves a similar performance as our own FiE baseline. FiE baseline uses the reader data released by the FiD-KD model, which has an R100 of 89.3 (vs 90.2 of `COS`). Considering that the gap between our method and FiD-KD model's top 100 retrieval recall is relatively small, this result is not surprising.

The HotpotQA results are shown in Table D.3. Overall our results are similar to previous SOTA methods on the dev set. At the time of the paper submission, we have not got the test set results on the leaderboard.

We adopted DPR evaluation scripts [1]for all the retrieval evaluations and MDR evaluation scripts [2] for all the reader evaluations.

[1]`https://github.com/facebookresearch/DPR`

[2]`https://github.com/facebookresearch/multihop_dense_retrieval`

D.4 Computation

Our COS has 182M paramteres. For COS pretraining, we use 32 V100-32GB GPUs, which takes about 3 days. For COS finetuning, we used 16 V100-32GB GPUs which takes about 2 days. Our reader model FiE has 330M parameters. We used 16 V100-32GB GPUs for training which takes about 1.5 days. For HotpotQA, both the path reranker and the reader have 330M parameters. We used 16 V100-32GB GPUs for training, the path reranker takes about 12 hours and the reader takes about 4 hours to train. We train all of our models once due to the large computation cost.

D.5 Licenses

We list the License of the software and data used in this paper below:

- DPR: CC-BY-NC 4.0 License
- MDR: CC-BY-NC 4.0 License
- Contriever: CC-BY-NC 4.0 License
- BLINK: MIT License
- NQ: CC-BY-SA 3.0 License
- HotpotQA: CC-BY-NC 4.0 License
- OTT-QA: MIT License
- EntityQuestions: MIT License
- SQuAD: CC-BY-SA 4.0 License
- WebQuestions: CC-BY 4.0 License