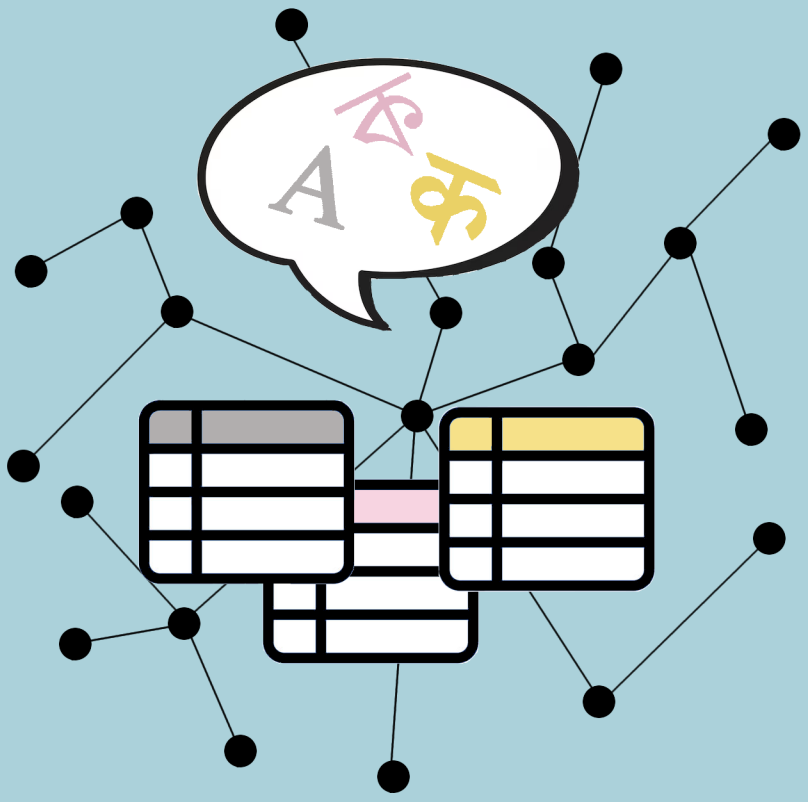
Information Seeking over Semi-Structured Tabular Data



Vaishali Pal

Information Seeking over Semi-Structured Tabular Data

Vaishali Pal

Information Seeking over Semi-Structured Tabular Data

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. P.P.C.C. Verbeek
ten overstaan van een door het College voor Promoties ingestelde
commissie, in het openbaar te verdedigen in
de Aula der Universiteit
op woensdag 3 september 2025, te 11.00 uur

door

Vaishali Pal

geboren te Sheoraphuli

Promotiecommissie

Promotor:	prof. dr. M. de Rijke	Universiteit van Amsterdam
Co-promotor:	prof. dr. E. Kanoulas	Universiteit van Amsterdam
Overige leden:	prof. dr. T. Blanke	Universiteit van Amsterdam
	dr. M.S.L. Cochez	Vrije Universiteit Amsterdam
	prof. dr. P.T. Groth	Universiteit van Amsterdam
	dr. M. Hulsebos	CWI
	prof. dr. C. Monz	Universiteit van Amsterdam
	prof. dr. F.A.H. van Harmelen	Vrije Universiteit Amsterdam

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

The research was carried out at the Information Retrieval Lab of the University of Amsterdam and in part during an internship at Naver Labs Europe. The research at the University of Amsterdam was supported by Elsevier's Discovery Lab.

Cover by Vaishali Pal
Printed by Proefschriftspecialist, Zaandam

ISBN: 978-94-93431-68-3

Acknowledgements

The decision to pursue a PhD was conceived during the COVID-19 lockdown in India and was as sudden as the complete lockdown. However, it is a decision I have found to be worthwhile. I had initially thought that the PhD would not be too difficult, having completed a rigorous Master by Research program in India. But adjusting to a new country in lockdown was not the easiest. I would like to thank all my IRLab colleagues who had made me feel welcome and made it easier.

I thank all the committee members, Tobias, Michael, Paul, Frank, Madelon, and Christof, for their time and patience to review my thesis. I would like to thank Maarten and Michael for being considerate and understanding during the PhD selection process and providing me the opportunity to pursue research at IRLab and Elsevier. Thank you Maarten for your advice and supervision during the PhD journey. I am especially grateful for your advice on organizing my research ideas into actionable items, writing papers, paying attention to details, providing feedback on presentations, and helping me to become a better researcher. I would also like to thank you for listening to my bicycle woes and for your general advice on bicycles. I still remember that I took you outside the IRLab container to help me figure out how to use a bike pump. Secondly, I would like to thank Evangelos for his supervision. I really appreciate our weekly meetings and conversations about not only research, but general things in life. Your advice about living in the Netherlands really made things easier when I first moved here.

I would like to thank all my colleagues. Thank you Maartje for showing me around Amsterdam and helping me with the little things when I first arrived. I would like to thank Clemencia for our conversations on any topic and for always listening. I really enjoyed our Barcelona trip and wish we could have done some more. Thank you Shashank for bringing the familiarity of India, for forcing me to Madras Diaries and The Yeti trips, and for being there for a random chat. I thank Ruben for being such a great friend and being one of the few NLP people in IRLab. I thank Weijia for all the collaborations and discussions we had in the last 4 years. I also thank Clara, Dylan, Jasmin, and Jingfen for the random but pleasant conversations in the office. I would like to thank all my colleagues at IRLab from whom I have learned so much during my PhD: MdR, Evangelos, Petra, Pablo, Ivana, Ali, Amin, Amy, Ana, Andrew, Antonis, Arezoo, Barrie, Chang, Chen, Chuan, Dan, Daniel, David, Fen, Gabriel, Gabrielle, Georgios, Hongyi, Ilias, Jia-Hong, Jie, Jin, Jingwei, Julien, Kidist, Lu, Maarten Marx, Maartje, Mohammad, Maria, Mariya, Maryam, Maurits, Maxime, Ming, Mohanna, Mounia, Mozhdeh, Olivier, Panagiotis, Philipp, Pooya, Romain, Roxana, Sam, Sami, Shaojie, Simon, Svitlana, Teng, Thilina, Thong, Siddharth Mehrotra, Siddharth Singh, Vera, Yangjun, Yibin, Yixing, Yuanna, Yongkang, Yougang, Yubao, Yuyue, Zahra, Zhirui, Zihan, and Ziming. I would also like to thank all my colleagues at Elsevier for their support and advice throughout the last 5 years: Anita, Daniel, Dimitris, Frank, Paul, Thom, Rinke, Romana, Masoud, Michael, Philip, Georgios, and Wytze.

Finally, I thank my parents Probir Kumar Pal and Supriya Banik Pal for always being there. I thank my husband, Prateek, for bearing with me during the last 5 years. I apologize to anyone I have missed and thank you for your help during my PhD journey.

Vaishali Pal
Amsterdam, June 2025

Contents

1 Introduction

Information seeking systems [8, 31, 178] aim to provide precise answers to natural language questions. Such systems are ubiquitous in digital assistants and in everyday devices that engage users to meet their information needs, such as smart speakers and chatbots that provide goal-oriented customer service [62] or chit-chat [18, 140]. Information seeking systems are comprised of two main constituent components [19]: a *retriever* [230] and a *reader* [6, 93, 198]. The retriever focuses on fetching relevant knowledge sources with respect to the user question which are then used by the *reader* to localize precise answers from the top-k retrieved sources. This *retriever-reader* approach allows complementary exploration directions to achieve the user-goal. The user-goal of the *retriever-reader* framework is to provide a natural and immersive user experience while providing a precise answer to the user question [228]. To create a natural and effective experience, user interfaces must accommodate various input and output modalities based on user preferences, such as structured lists or tables in addition to unstructured text. Moreover, support for diverse modalities of the retrieved context, responses in an extractive or generative manner, and the ability to cater for closed-domain question answering (QA) [4] versus open-domain QA [164] are necessary for an immersive experience in an unconstrained setup. This thesis studies these challenges with a specific focus on the *retriever* and *reader* over structured and unstructured contexts.

Although research in information seeking systems has advanced rapidly in recent years, the focus has mostly been concentrated on unstructured text [34]. Heterogeneous sources of information such as relational databases, spreadsheets [78], charts [135] etc. have been under-explored. This thesis aims to narrow the research gap with respect to one such structured information source: tables. Tables are collections of facts, entities and numbers, and are not grammatically well-formed sequences [55, 137]. Recent advances in information seeking systems that handle unstructured text are mostly due to the introduction of pre-trained language models [1, 10, 40, 120, 136]. However, pre-trained language models experience a distribution shift [50] when handling tables, as they are typically trained on unstructured text data. Additionally, tables introduce novel challenges [226] to language models that are absent in text-based knowledge sources, such as structure perturbations [76, 223], structure understanding [201], and table manipulation [219]. The challenges of free-flow text become even more pronounced when applied to tables. For instance, fact-verification [23, 137] is pivotal to table

processing instead of the fluency of the language model. Furthermore, long-tail entities are common in tables, such as polysemic [134] entities that have the same surface-form but different semantic forms. This leads to difficulties in word and phrase sense disambiguation [177] of entities [229]. For example, the Wikipedia table on Mayors in Amsterdam, New York[1] lists John Carmichael as elected mayor in 1885, which is ambiguous with the Australian pianist, composer, and music therapist John Carmichael,[2] due to a similar surface-form. A table with only the lexical mention of John Carmichael without additional contextual information of the surrounding text or without the meta-data from entity linking [182, 183] makes disambiguation challenging for language models. Moreover, many tables store temporally evolving data, such as a relational database of students' grades in a course. These aspects of tables make it difficult for language models pre-trained on unstructured text to reason on tabular data.

This thesis proposes to close this research gap by investigating structured tables as the source of knowledge for the *reader* to address user information needs. Table question answering (tableQA) is the task of addressing user information needs by providing precise answers to user questions by reasoning over tabular data. Research on tableQA is under-studied with recent work [64, 207, 220, 227] based on a single tabular context and answer-span classification of factoid questions [14, 81, 180]. To address this limitation, this thesis investigates generative approaches by studying different output modalities of tableQA, such as generative answer generation [42, 187], table generation [109, 197] and summary generation [53, 96, 100]. In addition, multiple table contexts are introduced to improve the breadth and complexity of queries that can be answered by existing tableQA models. Lastly, a low-resource [133] tableQA setting is investigated, focusing on the challenges of dataset/model scarcity and poor alignment of language models to culturally relevant facts in a low-resource language.

The last part of this thesis explores the *retriever* module of an information seeking system. Specifically, a sparse neural retriever [47] as a first-stage retriever is studied from the point of view of parameter-efficient adaptation on various text corpora.

The next section describes the research questions that are answered in this thesis. The research questions highlight some of the contributions of this thesis in addressing important challenges of tableQA discussed above and investigating first-stage retrievers in a parameter-efficient setting.

1.1 Research Outline and Questions

This thesis focuses on the two components of information seeking systems: the reader or the machine comprehension module (Chapter 2, 3, 4, 5) and the retriever (Chapter 6) which focuses on retrieval from text-based inputs. The thesis contains two broad research themes:

> *How to design and develop machine comprehension reader over semi-structured tables to aid the information needs of users?*

and

[1] https://en.wikipedia.org/wiki/Amsterdam,_New_York#Mayors
[2] https://en.wikipedia.org/wiki/John_Carmichael_(composer)

How to design an efficient and effective retriever to aid question answering on textual data?

Our research questions are grouped under these two themes. The first, "reader" theme is addressed through four related research questions.

(RQ1) How do language models compare on the generative question answering task when the context is unstructured text or semi-structured tables?

Prior work on generative QA has only focused on textual contexts [42], while QA over structured tables has been explored primarily for span classification of factoid-based answers [64, 227]. As a result, there is little research [141] on tableQA with generative answers. To address this research gap, Chapter 2 investigates the task of question answering on structured tables or unstructured text with generative answers. With the usage of parameter-efficient adapters [68], this chapter studies the effectiveness-efficiency trade-off of using different modalities in the input context of QA.

(RQ2) How can we leverage multiple tabular contexts to perform complex tabular reasoning to address user needs?

Real world questions over tables span across multiple tabular contexts [16, 208, 209], such as relational tables in a database, related web tables in a Wikipedia page, etc. However, prior work [64, 121, 207, 227] has only focused on reasoning over single tables for meeting the information needs of a user. This limits the complexity and scope of questions and operations that can be addressed by the reader model. Chapter 3 addresses this limitation by introducing the task of multi-table QA. As there exists no dataset or model for this task, a methodology for scalable dataset creation and effective model training is designed. Further, metrics to evaluate the trained models are also proposed. Experimental results show the efficacy of the proposed training scheme and dataset on different datasets and domains.

(RQ3) How to generate summaries over multiple tables for conversational agents?

While RQ2 introduces the multi-table QA task, the tabular output modality is inconvenient for real-world information seeking agents such as chatbots and conversational agents [92]. Further, text-based summaries provide a more human-readable description than large output tables. This motivates the exploration of query-focused multi-table summarization in Chapter 4. Specifically, two different prompting methodologies are investigated for this task: direct summarization and reason-then-summarize. Moreover, different aspects of generation quality such as faithfulness, completeness, and fluency are explored. A comparison of different automatic evaluation metrics for summarization is done with human evaluation to investigate a structure-compatible evaluation technique. Experimental results on fine-tuned and prompt-based in-context learning demonstrate the effectiveness of our proposed summarization methodology over state-of-the-art baselines.

(RQ4) How to adapt tableQA for low-resourced languages?

Chapter 5 introduces the task of low-resource tableQA focusing on two Indo-Aryan languages: Bengali and Hindi. Prior work [85] on low-resource tableQA has only been limited to one language, Korean, where the authors provide a dataset and describe the dataset creation methodology for Korean. There is a lack of general methodology for low-resource tableQA that can be applied to any language. In addition to well-known challenges that come with working in low-resource settings [61, 84, 176], such as data scarcity and the poor alignment of neural models, semi-structured tables introduce new challenges that further complicate machine reading comprehension. As tables are a collection of facts, long-tailed entities and culturally relevant facts are often under-represented in high-resource datasets and models. To address all these challenges, Chapter 5 investigates an automatic dataset generation methodology for low-resource languages that can be applied to any language with a Wikipedia presence. The result of this methodology is a large-scale tableQA dataset for Bengali and Hindi. Additionally, large language models (LLMs) and small encoder-decoder models are trained on the generated dataset to explore various architectures and training processes. Experimental results on manually annotated test sets demonstrate the models' effectiveness while an analysis on different mathematical operations and a zero-shot cross-lingual transfer setting demonstrate the generalizability of the trained models.

Under our second, "retriever" theme, we address the following research question:

(RQ5) How can we balance the efficiency-accuracy trade-off to leverage efficient sparse neural models as first-stage rankers?

Prior work on efficient neural retrieval models has focused only on dense bi-encoder [71] models, which are often used for second-stage ranking [124, 144]. There has been little attention to studying sparse neural models [47] often used as a first-stage ranker. Chapter 6 is a first study of the use of parameter-efficient adapters on sparse neural models. This study investigates whether parameter-efficient transfer learning performs comparable to memory-heavy fine-tuning for efficient adaptation of the sparse neural model SPLADE [47]. Specifically, the efficiency-accuracy trade-off of full fine-tuning is compared to parameter-efficient adapter-tuning. In addition, the effectiveness of adaptation to new domains is explored. Further, analysis of the different adapter-layers is performed to investigate their effect on performance. Lastly, experiments are conducted to transfer knowledge between cross-encoder [172] re-rankers and bi-encoder first-stage rankers.

1.2 Main Contributions

The main contributions of this thesis are arranged into different categories: datasets, models, and methods, analyses, and assessments. We list the contributions of each category below as follows:

1.2.1 Dataset Contributions

- *Multi-table question answering (MultiTabQA):* Chapter 3 develops a collection of datasets to aid the task of question answering over multiple tables. As Chapter

3 introduces the multi-table QA task, there were no existing datasets available for the task. Chapter 3 addresses this research gap and follows a curriculum learning paradigm and designs two types of pre-training datasets: single table pre-training datasets and multi-table pre-training datasets. The pre-training datasets help in table generation and understanding of complex formal queries and structured input tables. Further, fine-tuning datasets are developed to aid the language models in understanding complex natural language questions. To summarize, the following datasets are produced:

  - Pre-training datasets: A single-table pre-training dataset focusing on table generation for SQL query answering over a single table and a multi-table pre-training dataset comprising of $132,645$ samples focusing on SQL query answering over multiple tables, are available at the Hugging Face model hub. Each sample comprises of a formal SQL query, the corresponding table(s), and an answer table.
  - Fine-tuning datasets: A collection of multi-table QA datasets is publicly available at the Hugging Face model hub. Each sample in the datasets comprises a natural language question, the associated tables and an answer table. The datasets are built over existing relational databases and span across different domains such as airlines information system (ATIS [30, 165]), geography facts (GeoQuery [212]), and cross-domain facts (Spider [208]). The Atis multi-table QA dataset comprises 384 training samples, 45 evaluation samples and 86 test samples. The GeoQuery multi-table QA dataset comprises 530 training samples, 49 validation samples and 253 test samples. The Spider multi-table QA dataset comprises $6,715$ training samples and 985 validation samples.

- *QFMTS:* Chapter 4 develops a query-focused multi-table summarization dataset over the MultiTabQA fine-tuning dataset [151] for the task of query-focused multi-table summarization. The dataset comprises $4,909$ query-summary pairs and the corresponding context tables.

- *Bengali table question answering (BanglaTabQA) and Hindi table question answering (HindiTabQA):* Chapter 5 creates two low-resourced tableQA datasets over Wikipedia tables for the two Indo-Aryan languages Bengali and Hindi. The BanglaTabQA dataset comprises 19K Wikipedia tables, 2M training, 2K validation, and 165 test samples. The HindiTabQA dataset contains 2K Wikipedia tables, 643K training, 645 validation, and 125 test samples. Each sample in the dataset comprises a natural language question, the corresponding Wikipedia table, and the answer table.

1.2.2 Model Contributions

- *MultiTabQA:* Chapter 3 develops a collection of models for the task of question answering over multiple tables.

  - Multi-table query answering: Models that answer an SQL query over multiple tables.

  - Multi-table question answering: Models that answer a natural language question over multiple tables. These fine-tuned models have been trained on datasets created and discussed in Chapter 3.

- *BanglaTabQA and HindiTabQA:* Chapter 5 introduces low-resourced tableQA models for the two Indo-Aryan languages Bengali and Hindi. Each model was trained on a natural language question, the corresponding Wikipedia table, and the answer table in the respective language.

1.2.3 Methods, Analyses, and Assessments

- *Analysis of the effectiveness of QA models with a new modality of semi-structured tables:* Chapter 2 introduces tables as a new input modality that comes with a massive input domain shift to language models performing generative QA task. Prior works [42, 98, 101] on generative QA focused only on unstructured text-based context.

- *Analysis of trade-off between efficacy and efficiency:* Chapter 2 also analyzes the effects of ablating adapter layers in both the encoder and the decoder in a generative setup. This analysis shows that comparable effectiveness can be achieved using only 1.5% trainable parameters and with only the last few encoder layers.

- *Metrics for multi-table QA:* Chapter 3 introduces table generation metrics to evaluate multi-table QA task. Previous work [64, 113] used text QA metrics such as denotation accuracy as the output modality were either factoids or summaries. The proposed task over complex questions often leads to tables as output, necessitating metrics that evaluate table generation.

- *Analysis of summarization metrics for query-focused multi-table summarization:* Chapter 4 analyzes existing text-based summarization metrics and shows that these are insufficient for query-focused table summarization as they do not evaluate the factuality of the summaries and focus only on fluency.

- *Budget-friendly scalable dataset generation methodology:* Chapter 5 introduces a methodology for large-scale automatic generation of low-resource tableQA data in a budget-constrained manner. As one of the challenges of the low-resource setting is data scarcity, designing an automatic dataset generation methodology is crucial. Chapter 5 is the first work to introduce such a method for salable data creation for any low-resource language for tableQA.

- *Analysis of trade-off between fine-tuning and adapter-tuning for sparse retriever model SPLADE:* Chapter 6 analyzes the efficiency-accuracy trade-off of parameter-efficient fine-tuning with adapters on the sparse retriever model SPLADE. Further, exploration is done on how much each adapter layer ablation affects retrieval effectiveness.

- *Generalizability of sparse retriever model:* Chapter 6 also studies whether parameter-efficient adapters are effective for improving generalizability of neural

sparse neural retriever to out-of-domain datasets. Additionally, studies on whether adapters can share knowledge between re-rankers and sparse first stage rankers are also conducted.

1.3 Thesis Overview

This thesis comprises various chapters studying the two broad themes of table question answering systems: the *retriever* and *reader*.

Chapter 2 explores techniques for addressing question answering from multiple modalities such as unstructured text or semi-structured tables. This chapter compares training techniques such as full fine-tuning and parameter-efficient adapter-tuning on the different data modalities and analyzes the trade-off different trainable parameter sizes have on efficiency and efficacy.

Chapter 3 introduces the task of question answering over multiple tables. This chapter describes the task, discusses a scalable dataset creation methodology, a training mechanism for better convergence, and introduces evaluation metrics suitable for the task.

Chapter 4 explores multi-table summarization over user queries. This chapter explores data creation with large language model (LLM), effective prompting mechanisms over different LLMs and studies different evaluation metrics.

Chapter 5 discusses table question answering over low-resourced languages. This chapter discusses challenges introduced by low-resourced languages such as the under-representation of cultural entities and facts, and a dearth of resources such as datasets and trained models. This chapter studies two Indo-Aryan languages, Hindi and Bengali, and explores potential solutions to address some of the issues of the low-resourced setting.

Chapter 6 explores retrievers and re-rankers using parameter-efficient adapters. Specifically, this chapter deviates from the *reader* module and focuses on the *retriever* module of an information seeking system. This chapter explores the sparse retriever SPLADE [46] and studies the trade-off between fine-tuning and adapter-tuning on retrieval effectiveness, out-of-domain adaptation, and knowledge sharing between re-rankers and first-stage rankers.

All of the chapters are self-contained and can be read in any order.

1.4 Origins

The research chapters in this thesis are built upon previously published work. Below we list the origins of the chapters in the thesis as well as the contributions made by the authors involved.

Chapter 2 is based on the following paper:

- V. Pal, E. Kanoulas, and M. de Rijke. Parameter-efficient abstractive question answering over tables or text. In *Proceedings of the Second DialDoc Workshop on Document-grounded Dialogue and Conversational Question Answering*, pages

41–53. Association for Computational Linguistics, May 2022. URL https://aclanthology.org/2022.dialdoc-1.5.

VP: Conceptualization, Investigation, Methodology, Resources, Validation, Visualization, Writing – Original Draft Preparation, Writing – Review & Editing. EK: Supervision, Writing – Review & Editing. MdR: Funding Acquisition, Supervision, Writing – Review & Editing.

Chapter 3 is based on the following paper:

- V. Pal, A. Yates, E. Kanoulas, and M. de Rijke. MultiTabQA: Generating tabular answers for multi-table question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6322–6334, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.348. URL https://aclanthology.org/2023.acl-long.348.

  VP: Conceptualization, Data Curation, Investigation, Methodology, Resources, Software, Validation, Visualization, Writing – Original Draft Preparation, Writing – Review & Editing. AY: Writing – Review & Editing. EK: Supervision, Writing – Review & Editing. MdR: Funding Acquisition, Supervision, Writing – Review & Editing.

Chapter 4 is based on the following paper:

- W. Zhang, V. Pal, J. Huang, E. Kanoulas, and M. de Rijke. QFMTS: generating query-focused summaries over multi-table inputs. In U. Endriss, F. S. Melo, K. Bach, A. J. B. Diz, J. M. Alonso-Moral, S. Barro, and F. Heintz, editors, *ECAI 2024 - 27th European Conference on Artificial Intelligence, 19-24 October 2024, Santiago de Compostela, Spain - Including 13th Conference on Prestigious Applications of Intelligent Systems (PAIS 2024)*, volume 392 of *Frontiers in Artificial Intelligence and Applications*, pages 3875–3882. IOS Press, 2024. doi: 10.3233/FAIA240951. URL https://doi.org/10.3233/FAIA240951.

  VP and WZ share first authorship. WZ: Data Curation, Resources, Methodology, Conceptualization, Validation, Writing – Original Draft Preparation, Writing – Review & Editing. VP: Methodology, Model training, Validation, Visualization, Software, Conceptualization, Investigation, Visualization, Writing – Original Draft Preparation, Writing – Review & Editing. JH: Supervision, Writing – Original Draft Preparation, Writing – Review & Editing, EK: Supervision, Writing – Review & Editing. MdR: Funding Acquisition, Supervision, Writing – Review & Editing.

Chapter 5 is based on the following paper:

- V. Pal, E. Kanoulas, A. Yates, and M. de Rijke. Table question answering for low-resourced Indic languages. In Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in*

Natural Language Processing, pages 75–92, Miami, Florida, USA, Nov. 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.5. URL https://aclanthology.org/2024.emnlp-main.5.

VP: Conceptualization, Data Curation, Formal Analysis, Investigation, Methodology, Resources, Software, Validation, Visualization, Writing – Original Draft Preparation, Writing – Review & Editing. AY: Writing – Review & Editing. EK: Supervision, Writing – Review & Editing. MdR: Funding Acquisition, Supervision, Writing – Review & Editing.

Chapter 6 is based on the following paper:

- V. Pal, C. Lassance, H. Déjean, and S. Clinchant. Parameter-efficient sparse retrievers and rerankers using adapters. In J. Kamps, L. Goeuriot, F. Crestani, M. Maistro, H. Joho, B. Davis, C. Gurrin, U. Kruschwitz, and A. Caputo, editors, *Advances in Information Retrieval*, pages 16–31, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-28238-6.

  VP: Conceptualization, Formal Analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – Original Draft Preparation, Writing – Review & Editing. CL: Supervision, Formal Analysis, Investigation, Methodology, Software, Validation, Writing – Original Draft Preparation, Writing – Review & Editing. HD: Investigation, Methodology, Software, Writing – Original Draft Preparation, Writing – Review & Editing. SC: Supervision, Investigation, Writing – Review & Editing.

The writing of the thesis also benefited from work on the following publications:

- G. Penha, S. Vakulenko, O. Dusek, L. Clark, V. Pal, and V. Adlakha. The seventh workshop on search-oriented conversational artificial intelligence (SCAI 2022). In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 3466–3469, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450387323. doi: 10.1145/3477495.3531700. URL https://doi.org/10.1145/3477495.3531700.

- Y. Luo, F. Lu, V. Pal, and D. Graus. Enhancing resume content extraction in question answering systems through T5 model variants. In *RecSys in HR'23: The 3rd Workshop on Recommender Systems for Human Resources*, CEUR Workshop Proceedings, 2023.

- E. Kanoulas, P. Eustratiadis, Y. Li, Y. Lyu, V. Pal, G. Poerwawinata, J. Qiao, and Z. Wang. Agent-centric information access. *arXiv preprint arXiv:2502.19298*, 2025. URL https://arxiv.org/abs/2502.19298 (under review).

2

Parameter-Efficient Abstractive Question Answering over Tables or Text

In this chapter, RQ1 is addressed with studying the generative question answering task over either unstructured text or structured tables. Previous work on generative QA does not process structured tables and only processes an unstructured textual context. This chapter aims to reduce this research gap by exploring both modalities of context in generative question answering, structured tables, and unstructured text. To enable processing of both modalities, two task-adapters, text adapter and table adapter, were used to process the unstructured textual context and the structured tabular context, respectively. Further, detailed experimentation and analysis was done not only to compare the performance of full fine-tuning with adapter-tuning, but also to compare the importance of each adapter-layer with adapter ablation.

2.1 Introduction

Information seeking systems over diverse contexts require model capabilities to reason over unstructured and structured data such as free-form text, tables, and images [3, 34, 70, 193, 216, 227]. Such systems might have the additional requirement of generating natural language responses if deployed as task-oriented conversational agents [17, 168, 170, 195]. Recent work on open-domain question answering (QA) predominately addresses these challenges by fine-tuning massive pre-trained language models on different modalities such as tables and text [64, 65, 91, 141, 207]. However, each model trained on a specific input type is incompatible with other modalities and requires modality-specific fine-tuning. For example, in tabular QA [64], the structure of the table is learnt by training additional position embeddings (row and column identifiers) to identify which row and column a table cell belongs to. This renders such modality specific models incompatible with free-form text-based models. Multi-modal models [227] can reason over both tables and text by concatenating the textual context and the flattened table, leading to longer input sequences and limiting the length of the context

This chapter was published as V. Pal, E. Kanoulas, and M. de Rijke. Parameter-efficient abstractive question answering over tables or text. In *Proceedings of the Second DialDoc Workshop on Document-grounded Dialogue and Conversational Question Answering*, pages 41–53. Association for Computational Linguistics, May 2022. URL https://aclanthology.org/2022.dialdoc-1.5.

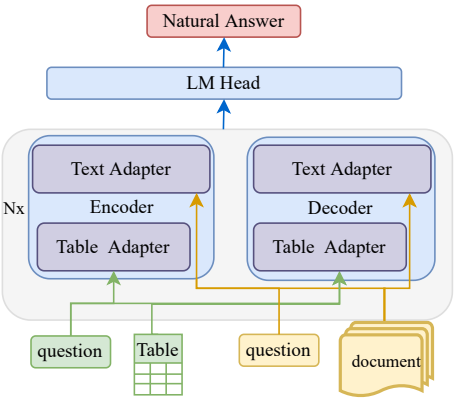


Figure 2.1: Parameter-efficient transfer learning using modality-specific (table/text) adapters for abstractive question answering.

that can be encoded.

To address these challenges, we study parameter-efficient transfer learning for abstractive QA over tables and over text. We are motivated to use adapter-layers that inject small bottle-neck layers between frozen pre-trained transformer layers as they achieve comparable performance to fine-tuning on a variety of tasks such as multilingual translation [54, 161, 162], classification [68], text-to-text generation [116], domain-adaptation in dialogue state tracking, and response generation [72].

Ablation studies on adapter layers [175] on masked language models such as BERT-base and RoBERTa over the GLUE benchmark demonstrate that removing beginning adapter layers leads to a minimal drop in performance. Extending adapter layer ablation over separate encoder and decoder modules is non-trivial as the conventional approach of sequential pruning of layers does not extend to consecutive encoder and decoder modules. Our work explores the interaction of adapter layers from both modules in the context of abstractive QA.

Lin et al. [116] explore the impact of the adapter bottle-neck dimension for various language generation tasks over an auto-regressive model such as GPT-2 [166]. They do not study tabular data nor ablate adapter layers, which is crucial in understanding impact of individual adapters in sequential transformer module architectures such as encoder-decoder. Our analysis is complementary to [116] as we ablate adapter layers to study parameter-performance trade-off whereas they only focus on adapter bottle-neck size. Also, we generalize beyond the text-to-text setting and explore language generation from structured or unstructured input such as tables and text. This introduces domain-shift in both the *task* and *structure* of the downstream data.

We propose a system, named **P**arameter, **E**fficient, **A**bstractive **Q**uestion **A**nswering (PeaQA), shown in Figure 2.1, which learns to reason over unstructured and structured input using a *shared* pre-trained language model and modality-specific adapter layers. We automatically transform hierarchical tables to regular tables to have a uniform representation without breaking associations between table cells. In addition, we extend the study of ablating adapter layers over both encoder and decoder modules.

Our main contributions are summarized as follows:

(1) We perform parameter-efficient abstractive question answering over multi-modal context using only additional 1.5% of trainable parameters for each modality. Our adapter-tuned model outperforms existing work by a large margin on tabular QA datasets and achieves comparable performance on a textual QA dataset.

(2) We study tabular QA as a new modality that introduces massive input domain shift to pre-trained language models. We propose a 2-step transformation of hierarchical tables to sequences, which produces a uniform representation to be used by a single, shared pre-trained language model and modality-specific adapter layers. To the best of our knowledge, this is the first work that explores tabular QA question answering in a parameter-efficient manner.

(3) We ablate adapter layers in both encoder and decoder modules to study their impact and show that beginning layers from both encoder and decoder can be eliminated without significant drop in performance. We also demonstrate that the last encoder adapter layers are indispensable and have a greater contribution than the decoder layers at the same level.

2.2 Related Work

Tabular question answering Tabular QA systems aim to answer questions from structured tables, which can be regular or hierarchical. Hierarchical tables can have header cells and body cells spanning across multiple rows and columns [25]. In most tabular QA systems [64, 91, 227], the structure of the table is encoded in the embedding layer of large language models by introducing table-specific position information such as row id and column id. Concurrent to our work, abstractive QA over tables [25, 141] poses additional challenges of generating natural answers by reasoning and aggregating discontinuous facts from the table.

Textual question answering Question answering over text measures a system's ability to comprehend free-form text in the user question and context passage(s) and predict an answer. The predicted answer can be extractive in nature, where the system identifies short text spans in the context passage to answer the user query [106, 159, 167, 179], or it can be abstractive, where it is required to generate a free-form answer [7, 138, 171, 206].

Transfer learning Transfer learning techniques, such as fine-tuning pre-trained models for downstream tasks, require a new set of parameters to be learnt for each new task. To avoid such memory-intensive transfer learning methods, adapters have been proposed as a parameter-efficient method of adapting to new domains [68, 161]. Adapters have been extended to language generation in a variety of generative tasks such as translation, summarization, multi-turn dialogue, and task-oriented natural language generation [116].

Our work combines all the aforementioned aspects to generate abstractive answers from *both* tables and text with only 0.7%–1.0% trainable parameters without compromising performance.

2.3 Model

We focus on encoder-decoder models for the task of abstractive question answering. We use a BART [107] encoder-decoder architecture which comprises a bidirectional encoder and an auto-regressive decoder. The input sequence consists of the question, the context title and context sequence preceded with prompts indicating the beginning of each sub-sequence. Formally, the input sequence is represented as $<$*question*$>$ q_0 q_1 $\ldots$ q_m $<$*title*$>$ t_1 t_2 $\ldots$ t_p $<$*context*$>$ c_0 c_1 $\ldots$ c_n, where q_i is the i^{th} question token, t_j is the j^{th} title token, and c_k is the k^{th} context token. The context can either be a text passage or a flattened table. The parameters of the pre-trained BART model are frozen during training. Modality-specific adapter layers added to the model are trained on either tabular context or textual context to generate natural answers.

2.4 Textual Question Answering

To study multi-modal abstractive QA, we first focus on free-form text as context to the system. We train adapter layers for textual context on the NarrativeQA dataset [98]. NarrativeQA is a complex abstractive question answering dataset over stories. The dataset contains $32,747$ samples in the training set, $3,461$ samples in the validation set, and $10,557$ samples in the test set. For our task, we have selected the input context passage to be the human annotated *summary* of each sample, which is the Wikipedia page summary of the story and represented as a paragraph. The input to the model is the *question*, *title* and *summary* of each passage, and the target is the abstractive answer.

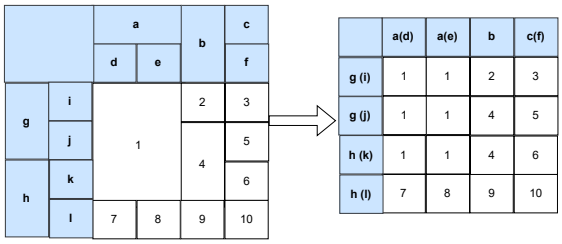
2.5 Tabular Question Answering

We study tabular QA as a new modality which introduces massive input domain shift to pre-trained language models. Tables enforce structural constraints in their representation, which is incompatible with the expected input format of pre-trained language models. To achieve our goal of parameter efficiency by utilizing a uniform pre-trained language model, we only train table-specific adapter layers while keeping the pre-trained model frozen. However, this necessitates a uniform input representation for both tables and text. An additional challenge is introduced to maintain uniformity across different table types (regular, hierarchical).

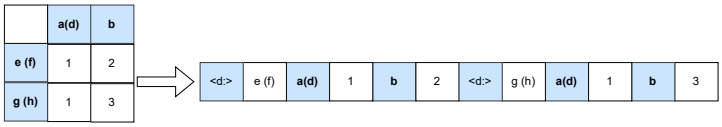
For our task, we explore two tabular QA datasets, namely, Tablesum [216] and FeTaQA [141]. Tablesum consists of 200 unique Wikipedia tables on which questions and abstractive answers are manually annotated; 40% of the samples are questions over hierarchical tables, but the tables in their released data are missing information in the hierarchical cells, and their work does not handle hierarchies. We address this issue by extracting the wikitables from the respective Wikipedia pages and release a clean version of the dataset.[1]

FeTaQA [141] is a larger abstractive tabular QA dataset consisting of question and free-form answers over $10,330$ regular tables. The dataset consists of $7,326$ samples in

[1]The cleaned data and code can be found at `https://github.com/kolk/Pea-QA`



(a) A multi-span table to a regular one.



(b) Linearize regular table to a sequence of *key*:*value* pairs.

Figure 2.2: Table representation.

the training set, $1,001$ in the validation set, and $2,003$ in the test set. FeTaQA consists of human-annotated answers containing explanations involving entities and relations.

2.5.1 Table Representation

For our work, we choose to represent all tables uniformly in a two-step process: (1) Transformation of a hierarchical table into a regular table; and (2) Linearization of a regular table into a flattened sequence which can be encoded with a language model.

Linearize hierarchical table headers Hierarchical table headers are linearized into a single row of headers by the following process. A header cell spanning multiple columns is duplicated and split into multiple cells. The cell values over which this header spans are then concatenated with the entire split. Repeating this process over all header rows flattens the hierarchical header into a sequential one. We depict this process in Figure 2.2a, which yields a linear header $a(d)$, $a(d)$, b, $e(f)$.

Linearizing table body Multi-span table body cells are parsed differently from headers. Each table body cell is replicated with one or multiple header cells depending on its span across columns. Cells that span across multiple rows are replicated with all the spanned rows. This process leads to a regular table. We flatten the regular table in row-major form, concatenating rows sequentially. Each row is a sequence of (*key, value*) pairs where a key is a column header and the value is the cell value of that column as depicted in Figure 2.2b.

2.6 Experimental Setup

We seek to answer the following research questions with our experiments: (RQ1) How does adapter-tuning perform compared to fine-tuning in the context of multi-modal

Dataset	Params	Adapter-Tuning	Fine-Tuning
All	scheduler	linear	linear
	batch size	32	32
	seed	6	6
	max epochs	15	15
Tablesum	learning rate	6e-4	4e-5
	input length	200	200
FeTaQA	learning rate	6e-4	8e-4
	input length	100	100
NarrativeQA	learning rate	1e-4	2e-5
	input length	50	50

Table 2.1: Hyper-parameters for training. **All** indicates all 3 datasets.

input? (RQ2) Do all adapter layers across the encoder and decoder contribute equally to performance across tasks/modalities?

2.6.1 Fine-Tuning

We perform all our experiments on the *large* variant of the BART model. We fine-tune the BART-large model over the 3 datasets as the state-of-the-art fine-tuned models utilize different architectures for different datasets making comparison with adapter-tuning difficult. We treat our fine-tuned BART models on the 3 datasets as baselines. We sweep learning rates from $\{8e^{-4}$, $6e^{-4}$, $3e^{-4}$, $1e^{-4}$, $5e^{-5}$, $4^{e}-5$, $3e^{-5}$, $2e^{-5}$, $1e^{-5}\}$ and select the best-performing learning rate for each dataset. We select $4e^{-5}$ for fine-tuning on Tablesum, $8e^{-4}$ on FeTaQA datasets and $2e^{-5}$ to fine-tune NarrativeQA. We use a batch size of 4 and a gradient accumulation of 8 to emulate an effective batch size of 32. The maximum target sequence length is set to 200 for tabular QA datasets and to 100 for the textual QA dataset. On the Tablesum dataset, we follow 5-fold cross validation as described in the original work to evaluate our models. On FeTaQA and NarrativeQA, we utilize the test split for evaluating our models. We train the model on each dataset for 15 epochs and evaluate on Rouge-2, Rouge-L and sacreBLEU metrics.

2.6.2 Adapter-Tuning

We perform adapter-tuning as a parameter-efficient alternative to adapt BART-large model to the abstractive question answering task in different modalities. We first freeze all layers of the pre-trained BART-large model which was trained on text reconstruction as mentioned in the original BART paper [107]. We add bottle-neck adapter layers from the Houlsby adapter configuration [68] which are trained to adapt to the downstream abstractive question answering task and also to the modality-specific input context. Each adapter layer has a bottle-neck embedding size of 64. As mentioned in Section 2.6.1, we sweep the learning rates and select the best performing learning rate for each dataset. We select $6e^{-4}$ for the tabular QA datasets Tablesum and FeTaQA, and select

Dataset	Model	Training	Rouge-1	Rouge-2	Rouge-L	BLEU
Tablesum [216]	GPT2	fine-tune	0.272	0.073	0.200	5.35
	T5		0.362	0.143	0.276	**10.43**
	Ours (Pea-QA)	fine-tune	**0.400**	**0.186**	**0.316**	6.30
		Adapter-tune	0.393	**0.186**	0.312	6.75
FeTaQA [141]	T5-small	fine-tune	0.550	0.330	0.470	21.60
	T5-base		0.610	0.390	0.510	28.14
	T5-large		0.630	0.414	0.530	30.54
	Ours (Pea-QA)	fine-tune	0.632	0.415	0.534	30.81
		Adapter-tune	**0.651**	**0.436**	**0.553**	**33.45**
NarrativeQA [98]	Masque [143]	fine-tune	–	–	**0.547**	–
	Ours (Pea-QA)	fine-tune	0.518	0.268	0.515	21.07
		Adapter-tune	0.510	0.270	0.500	20.08

Table 2.2: Results: Scores obtained on the Tablesum, FeTaQA and NarrativeQA datasets.

$1e^{-1}$ to train the textual QA dataset NarrativeQA. We use the same batch size and maximum target sequence length as fine-tuning for effective comparison. A summary of hyper-parameters is mentioned in Table 2.1.

2.6.3 Ablation Study: Adapter Pruning

Adapter-layer pruning has been explored on the GLUE benchmark in [175], which demonstrates that removing adapter layers from the beginning of BERT-base and RoBERTa models leads to minimal performance drop. We extend adapter layer ablation to encoder-decoder architectures and hypothesize that this phenomenon should be observed on both the encoder and decoder modules. However, it is non-trivial how the adapter-layers in the encoder and decoder interact with each other and contribute to performance. Previous studies [175] on adapter ablation prune consecutive adapter layers in masked language models. This approach does not extend directly to sequential modules of encoder-decoder where intra-module adapters not only contribute to their respective objective of encoding and decoding but also contribute to intermodule interaction and performance. To measure the impact of the adapter layers in different modules, we perform adapter ablation in both the encoder and the decoder. First, we uniformly remove adapter layers from both encoder and decoder modules starting from the beginning layers of both modules and finally deleting all layers. This leads to 12 experiments, corresponding to the elimination of 12 encoder and 12 decoder adapter layers. To study interaction across inter-module adapters at different levels, we conduct 36 experiments of different configurations of adapter elimination from the last 6 levels of encoder and decoder. We analyze the performance of each configuration in Section 2.7.3.

2.7 Results

We compare the results of our baseline fine-tuned models with the state-of-the-art fine-tuned models in Section 2.7.1. We address (RQ1) "How does adapter-tuning perform compared to fine-tuning in the context of multi-modal input?" in Section 2.7.2 and (RQ2) "Do all adapter layers across the encoder and decoder contribute equally to performance across tasks/modalities?" in Section 2.7.3.

2.7.1 Fine-Tuned Models

We study the results of our baseline fine-tuned models with the state-of-the-art fine-tuned models for the 3 datasets. The results of the experiments are shown in Table 2.2. We observe that for the Tablesum dataset, our fine-tuned model outperforms the best state-of-the-art T5 model in Rouge-1 by 3.8%, Rouge-2 by 4.3%, and Rouge-L score by 4%. This can be attributed to fine-tuning our model on the clean version of the dataset. Our fine-tuned models perform comparably to the state-of-the-art T5-large on the FeTaQA dataset, i.e., 0.2% on Rouge-1, 0.01% higher on Rouge-2 and 0.04% higher on Rouge-L. Our fine-tuning results on NarrativeQA are lower than state-of-the-art models trained with sophisticated reasoning architecture. The focus of this work was primarily on comparing fine-tuning and adapter-tuning and hence we leave explicit reasoning as part of future work.

2.7.2 Adapter-Tuned Models

We address (RQ1) by comparing the performance of the adapter-tuned models to our baseline fine-tuned models. For Tablesum, as observed in Table 2.2 fine-tuning (baseline) marginally outperforms adapter-tuning with 0.7% higher Rouge-1 and 0.4% higher Rouge-L scores while having the same Rouge-2 score. For FeTaQA, adapter-tune shows a larger performance gain with 1.9% on Rouge-1 and Rouge-L and 2.1% on Rouge-2 compared to fine-tuning. The insignificant gains in fine-tuning over adapter-tuning for tabular QA can be attributed to catastrophic forgetting [21, 48, 95] induced by differences in the distribution of downstream tabular data format from the original text data format of pre-training.

To explore this phenomenon further, we analyze examples from FeTaQA dataset in Table 2.3 where adapter-tuning outperforms fine-tuning. We observe that the fine-tuned model is unable to disambiguate surface-form similarities from the column semantics in the first example. The intended semantics of the named-entity *Akhila Kishore* in the question is *Actor*. While the surface-form is similar to the column value *Akhila*, the intended semantics is that of the column header *Role*. The fine-tuned model wrongly predicts the second and third rows of the tabular context as correct grounding of information, while adapter-tuning is able to disambiguate and predicts information from the first 2 rows as answer. We observe that the fine-tuned model also predicts information from the wrong column *Director* instead of *Cast* in the second example. Adapter-tune correctly identifies the column, but partially generates the required information in the prediction. The third example depicts both non-factual and non-fluent prediction by the fine-tuned model.

Question: What and when were Akhila Kishore's first two films?
Target: akhila kishore made her debut in the kannada film padhe padhe (2013), and appeared in kathai thiraikathai vasanam iyakkam (2014).

Table:

Year	Film	Role	Language
2013	Padhe Padhe	Kanchana	Kannada
2014	Kathai Thiraikathai Vasanam Iyakkam	Daksha	Tamil
2015	Inimey Ippadithaan	Akhila	Tamil
...	...	...	

Adaper-tune: akhila kishore made her debut in the kannada film padhe padhe (2013) and kathai thiraikathai vasanam iyakkam (2014).
Fine-tune: kathai thiraikathai vasanam iyakkam (2014) and inimey ippadithaan (2015) were kannada films.

Question: Who is the starring actor of Aastik?
Target: aastik is a 1956 hindi film starring shahu modak, paro devi and meenakshi.

Table:

Title	Director	Cast
...	...	...
Aastik	S. P. Kalla	Shahu Modak, Paro Devi, Meenakshi, B. M. Vyas, Praveen Paul
Alam Ara	Nanubhai Vakil	Daljeet, Chitra, Tiwari, Niranjan Sharma, Minu Mumtaz,...
...	...	...

Adaper-tune: aastik is a 1956 bollywood film starring shahu modak.
Fine-tune: a directed by s. p. kalla.

Question: What were the three films directed by Yakub and when were they released?
Target: yakub directed three films: sagar ka sher in 1937, uski tamanna in 1939, and, in 1949, aiye.

Table:

Year	Film	Director
...	...	...
1937	Sagar Ka Sher (Lion of Sagar)	Yakub
...	...	...
1939	Uski Tamanna (Her Last Desire)	Yakub
...	...	...
1949	Aiye	Yakub
...	...	...

Adaper-tune: yakub directed three films: sagar ka sher (lion of sagar) in 1937, uski tamanna (her last desire) in 1939 and aiye in 1949.
Fine-tune: y directed by yakub.

Table 2.3: Samples where adapter-tuning outperforms fine-tuning.

We demonstrate an example of a hierarchical table of Tablesum in Table 2.4. The question requires aggregation on the table cells and has various summary-like targets associated with it. The hierarchical table mentions *Ed Sheeran* 3 times, but the actual number of occurrences is 10 times, from *Week 3* to *Week 9*, *Week 31* and from *Week 42* to *Week 43*. Our table transformation process handles this to produce a regular table with 10 cells containing *Ed Sheeran* as a value. The models can simply aggregate over the mentions. As shown in Table 2.4, both models generate long answers summarizing the information from the context table. However, since the models do not explicitly handle cell aggregation, we observe factual mistakes in both adapter-tuned and fine-tuned models. The models find Tablesum samples challenging, even though the generated

Question: how many times was ed sheeran listed as the performer?
Targets:

- Ed Sheeran was listed as a performer twice in the table documenting the top hits of 2014 in Sweden. Other English-Language top performers included Bruce Springsteen, Sam Smith, and Coldplay, implying that English-Language music has significant success in Sweden.
- According to the table, in 2014, Ed Sheeran was only listed as the performer one time. It was for the song that he performed that is called "I See Fire", which was out in January and February of 2014

Table:

Week	Date	Song title	Performer	Album title	Performer
1	3 Jan 2014	"Timber"	Pitbull feat. ...	True	Avicii
2	10 Jan 2014				
3	17 Jan 2014	"I See Fire"	Ed Sheeran	High Hopes	Bruce ...
4	24 Jan 2014				
5	31 Jan 2014			True	Avicii
6	7 Feb 2014			Christer Sjögren sjunger Sinatra	Christer Sjögren
7	14 Feb 2014				
8	21 Feb 2014			True	Avicii
9	28 Feb 2014				
...	...	...	...		
31	31 July 2014	Prayer in C...	Lilly Wood &...	X	Ed Sheeran
32	7 Aug 2014			Honky Tonk Rebels	Lasse Stefanz
...	...			...	...
42	16 Oct 2014	"The Days"	Avicii	X	Ed Sheeran
43	23 Oct 2014				
44	30 Oct 2014			Songs for Daddy	Jill Johnson
...	...	...	...		

Adapter-tune: I found a table of the year 2014. It shows the week of the week, the song title, the artist, the title of the song, the album title, and the performer. There are 11 times that Ed Sheeran was listed as the performer in the year. The song title is "Timber" and the album is True.
Fine-tune: I found a table of Ed Sheeran's year in 2014. He was listed as the performer 14 times in the year 2014. The first time he was listed was on 3 January 2014 with the song "Timber" and the last time was on 4 April 2014 with "I See Fire".

Table 2.4: Example from the Tablesum dataset.

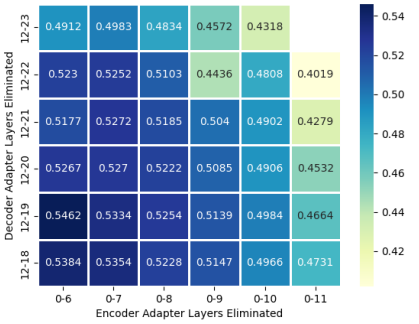
language is fluent and readable.

For textual QA, on the NarrativeQA dataset, adapter-tuning performs comparable to fine-tuning with the adapter-tuned model achieving 0.8% lower Rouge-1, 1.8% higher Rouge-2 and 1.5% lower Rouge-L scores than fine-tuning.

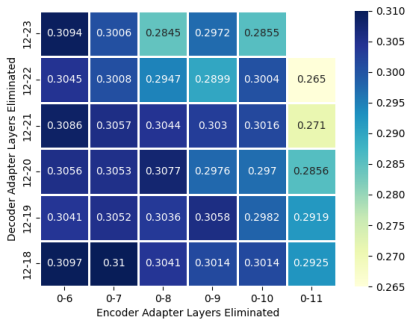
We conclude that adapter-tuning performs better than fine-tuning for out-of-domain tabular data and comparable performance on in-domain text.

2.7.3 Ablation of Adapter Layers

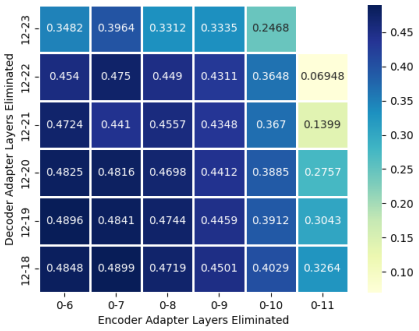
We study (RQ2) by ablating adapter layers in both encoder and decoder modules. We uniformly eliminate successive adapter layers from both encoder and decoder starting from the first layer in both modules and finally deleting all layers. This leads to 12 experiments corresponding to 12 encoder and 12 decoder adapter layers. We number the encoder adapter layers from 0–11 and the decoder adapter layers from 12–23. We



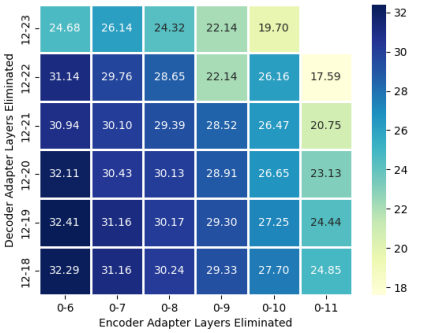
(a) FeTaQA Rouge-L scores



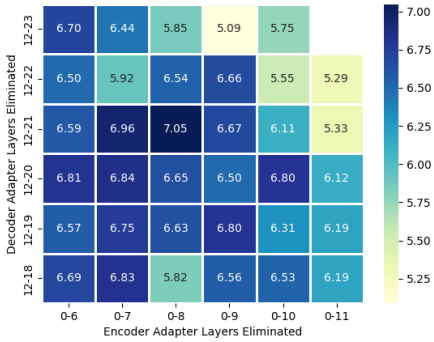
(b) Tablesum Rouge-L scores



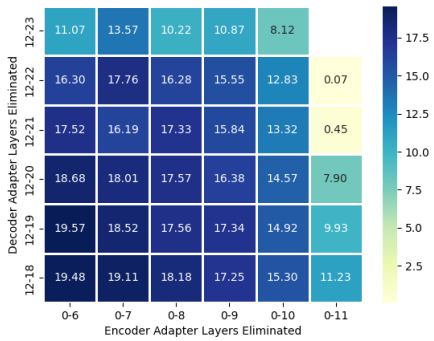
(c) NarrativeQA Rouge-L scores



(d) FeTaQA sacreBLEU scores

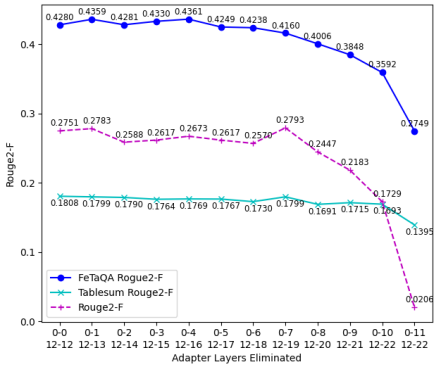


(e) Tablesum sacreBLEU scores

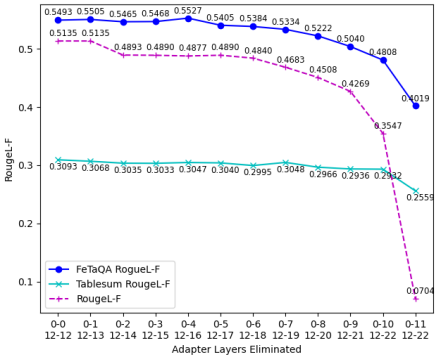


(f) NarrativeQA sacreBLEU scores

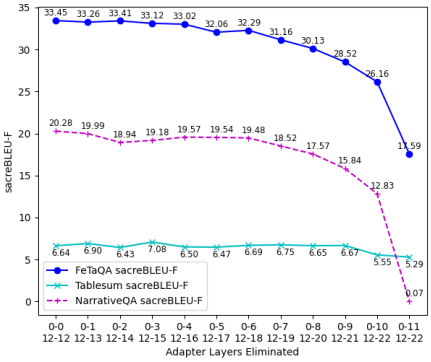
Figure 2.3: Adapter layer ablation scores. The X-axis represents range of encoder adapter layers deleted, the Y-Axis represents range of decoder adapter layers deleted. x-y implies all adapter layers from x to y inclusive. There are 36 model ablation configurations displayed. The ablation starts from 0 to 6 encoder adapter layers removal and 12 to 18 decoder adapter layer removal represented by the bottom left cell ((0–6), (12–18)) and progressively increases deletion of encoder adapter layers along the X-axis and decoder adapter layers along the Y-axis.



(a) Adapter layer ablation Rouge2 F-scores. The X-axis depicts encoder-adapter layers (0–11) and decoder adapter layers (12–23) deleted progressively. Each $\binom{x-y}{r-s}$ represents F-score with encoder layers p to q deleted and decoder layers r to s deleted.



(b) Adapter layer ablation Rouge-L scores. The X-axis depicts encoder-adapter layers (0–11) and decoder adapter layers (12–23) deleted progressively. Each $\binom{x-y}{r-s}$ represents F-score with encoder layers p to q deleted and decoder layers r to s deleted.



(c) Adapter layer ablation sacreBLEU F-scores. The X-axis depicts encoder-adapter layers (0–11) and decoder adapter layers (12–23) deleted progressively. Each $\binom{x-y}{r-s}$ represents F-score with encoder layers p to q deleted and decoder layers r to s deleted.

Figure 2.4: Ablation of adapter layers.

measure the performance of the models using Rouge-2, Rouge-L[2] and sacreBLEU[3] scores. The F-scores for each dataset (NarrativeQA, Tablesum, FeTaQA) are shown in Figure 2.4a, 2.4b and 2.4c, respectively. We observe that as more adapter layers are eliminated, performance drops across all datasets. However, the performance drop is minimal until the last adapter layers are also deleted. The inflection point varies

[2] https://pypi.org/project/rouge-score/
[3] https://github.com/mjpost/sacreBLEU

Adapter-tune		#Trainable parameters
Encoder adapters removed	**Decoder adapters removed**	
–	–	$6,343,680$ (1.56%)
0–2	12–14	$4,757,760$ (1.17%)
0–4	12–16	$3,700,480$ (0.91%)
0–6	12–18	$2,643,200$ (0.65%)
0–8	12–20	$1,585,920$ (0.39%)
0–10	12–22	$528,640$ (0.13%)
0–11	12–22	$264,320$ (0.07%)
fine-tune		$406,291,456$ (100%)

Table 2.5: Trainable parameters in the encoder and decoder. Encoder adapter layers are numbered from 0–11 and decoder adapter layers are numbered from 12–22. x–y implies all adapter layers from x to y inclusive.

across datasets but is limited to the last 2 layers of the encoder and decoder. For the NarrativeQA dataset, this point is when all layers till the second last adapter layer from both the encoder and decoder are deleted. For the FeTaQA and Tablesum datasets, the performance drops sharply only when the last encoder and decoder layers are removed.

To analyze the contribution of the i^{th} adapter layer of the encoder and decoder to the performance, we perform ablation of the adapter layers (0–6), (0–7), . . . , (0–11) from the encoder and adapter layers (12–18), (12–19), . . . , (12–23) from the decoder (decoder layers are numbered 12–23). This leads to 36 configurations where a configuration (p–q, r–s) represents the removal of all encoder adapters from the p^{th} to q^{th} layer and all decoder adapters from r^{th} to s^{th}. The results are shown in Figure 2.3. We observe that performance remains comparable as we progressively remove adapter layers from encoder and decoder until the last few layers. The performance drops steeply when we remove the last encoder and decoder adapter layer depicted towards the top-right corner of the RougeL scores in Figures 2.3a, 2.3b, and 2.3c and the BLEU scores in Figures 2.3d, 2.3e, and 2.3f. This implies that the last adapter layer learns most of the domain information.

We also observe that the last encoder and decoder layers contribute differently to performance. Removing the last encoder layer (column 0–11) leads to a substantial score drop across all decoder layers. This indicates that the last encoder layer is indispensable. Keeping only the last decoder adapter (row 12–23) is comparable to keeping the last two last encoder layers (column 0–10). We also observe that retaining only the last 50% of adapter layers from both the encoder and decoder increases parameter efficiency by 0.7% of the parameters as summarized in Table 2.5 without significant performance compromise.

2.8 Conclusion

We are the first to study parameter-efficient transfer learning over tables and text for abstractive question answering using adapters. We demonstrate that parameter-efficient adapter-tuning outperforms fine-tuning on out-of-domain tabular data and achieves comparable results on in-domain textual data.

We propose a transformation from hierarchical tables to regular ones and further into a sequential form compatible with pre-trained models. We extend an existing ablation study of adapter layers to encoder-decoder setting and demonstrate that adapter layers from the end of the encoder are indispensable to encoding modality-specific information than decoder adapter layers at the same level.

Our results are useful for exploring the scalability of QA models in memory-constrained situations with comparable performance while scaling across modalities using light-weight adapters.

One of the limitations of our work is that our models do not explicitly reason and aggregate over the table cells. This might lead to fluent but factually incorrect answers on the challenging Tablesum dataset. Addressing this limitation is left for future work.

Returning to RQ1, *How do language models compare on the generative question answering task when the context is unstructured text or semi-structured tables?*, which motivated the work in this chapter, we find that language models pre-trained on unstructured text can be adapted to structured tables with either fine-tuning or adapter-tuning, with adapter-tuning, with a fraction of training parameters, achieves comparable results compared to fine-tuning. Further, ablation studies demonstrate the impact of each adapter layer, depicting that further layer-pruning leads to negligible drop in tableQA performance with significant reduction in trainable parameters.

Chapter Appendices

We provide further details on the statistics of the datasets used (Appendix 2.A) and on the Rouge-2 scores for an encoder-decoder adapter layer ablation study (Appendix 2.B).

2.A Dataset Statistics

Statistics of the three datasets, i.e., Tablesum, FeTaQA and NarrativeQA are listed in Table 2.A.1. Tablesum has the longest answer length. The answers are summary-like, often, describing aspects of the table contents. The FeTaQA dataset contains answers of mostly single sentences and targeted towards specific facts asked in the question. The NarrativeQA dataset focuses on questions from stories. The answer lengths vary from single words to long sentences. For the tabularQA dataset, Tablesum contains larger tables than the FeTaQA dataset even though it is limited to 200 unique tables over which questions are asked. The FeTaQA dataset's tables contain more columns on average than Tablesum.

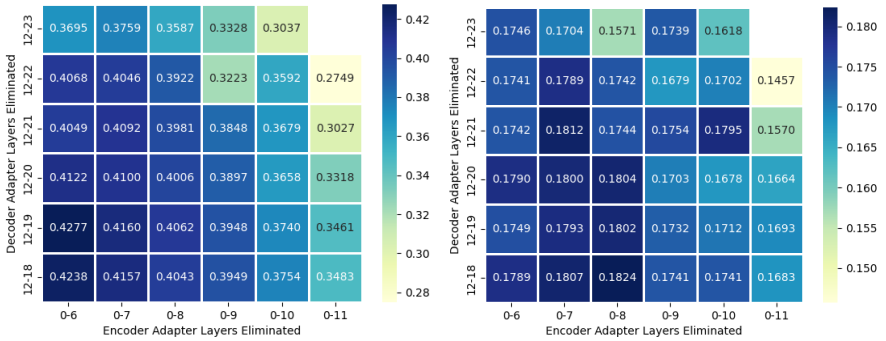
Tablesum	
Domain	Open
Modality	Table
Table-type	Regular
Training samples	798
Validation samples	200
Test samples	–
Max question length	114
Max target length	1,579
Max table row	155
Max table column	8
FeTaQA	
Domain	Open
Modality	Table
Table-type	Hybrid
Training samples	7,326
Validation samples	1,001
Test samples	2,003
Train max question length	165
Train max target length	338
Train max table rows	34
Train max table columns	30
Val max question length	182
Val target length	325
Val max table rows	34
Val max table columns	22

Test max question length	193
Test max target length	295
Test max table lows	34
Test max table columns	22
NarrativeQA	
Domain	Stories
Modality	Text
Training samples	65,494
Validation samples	6,922
Test samples	21,114
Train max question length	175
Train max target length	171
Train max context length	6,045
Val max question length	158
Val target length	187
Val max context length	6,033
Test max question length	1,220
Test target length	224
Test max context length	6,090

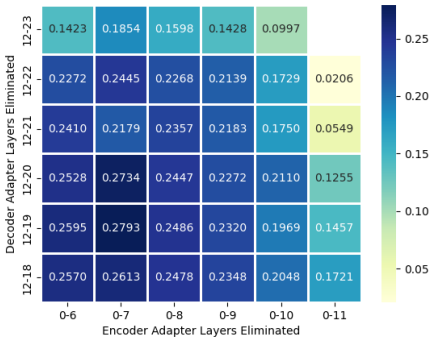
Table 2.A.1: Dataset Statistics

2.B Encoder-Decoder Adapter Layer Ablation Rouge-2 Scores

Ablation results (Rouge-2 F-scores) of 36 configurations of adapter layers deleted from the later half of the encoder and decoder. Deleting the last encoder adapter layers leads to massive drop in performance as observed in the last three columns of Figures 2.A.1a, 2.A.1b and 2.A.1c. However, deleting the last decoder adapter layers results in better performance in comparison to the encoder layers at the same level as observed from the top 3 rows.



(a) FeTaQA (b) Tablesum



(c) NarrativeQA

Figure 2.A.1: Adapter layer Rouge-2 ablation scores. The X-axis represents range of encoder adapter layers deleted, the Y-Axis represents range of decoder adapter layers deleted. x-y implies all adapter layers from x to y inclusive. There are 36 model ablation configurations displayed. The ablation starts from 0 to 6 encoder adapter layers removal and 12 to 18 decoder adapter layer removal represented by the bottom left cell ((0–6), (12–18)) and progressively increases deletion of encoder adapter layers along the X-axis and decoder adapter layers along the Y-axis.

3

MultiTabQA: Generating Tabular Answers for Multi-Table Question Answering

The previous chapter (Chapter 2) discussed the task of answering user questions over different context types, such as structured tables or unstructured text. This chapter focuses on answering user questions only over structured tables. Prior work on table question answering has focused only on single tables for the table question answering task. However, real-world questions are often complex and over multiple tabular context; where answering the question requires utilizing the information spanning over multiple tables. This limits the scope of the questions and reduces the challenge of the task. To reduce this research gap, this chapter introduces the multi-table question answering task, i.e., the task of answering questions over multiple tables. As there are no existing resources for the task, this chapter focuses on formalizing the task methodology, dataset creation methodology, and model development. Further, detailed qualitative and quantitative analysis is done to study the impact of multiple tables on model performance. More specifically, RQ2 is addressed by introducing the task of question answering over multiple tables.

3.1 Introduction

Question answering (QA) over multiple tables aims to provide exact answers to natural language questions with evidence from one or more tables [83]. This is in contrast to single-table QA, which has been the focus of tabular QA research to date [64, 121, 141, 227]. Even though groups of related tables are ubiquitous in real-world corpora, such as relational databases or tables in a web page, multi-table QA remains a largely unexplored area.

To address this gap, we propose a new task of answering questions over multiple

This chapter was published as V. Pal, A. Yates, E. Kanoulas, and M. de Rijke. MultiTabQA: Generating tabular answers for multi-table question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6322–6334, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.348. URL https://aclanthology.org/2023.acl-long.348.

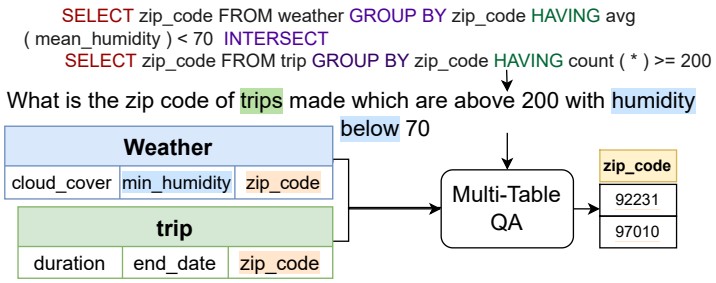


Figure 3.1: Multi-table QA. The QA model generates a tabular answer from either a natural language question or an SQL query and one or more tables as input context.

tables. Our multi-table QA model, MultiTabQA,[1] addresses the novel challenges introduced by multi-table context. These include complex queries involving chains of reasoning, disambiguation of relevant table names at each reasoning step, and generating a final table as the answer. It also leads to novel question-types that are unnatural in a single-table setting. For instance, questions involving operations specific to multiple tables, such as Cartesian products (*outer joins*, *inner joins*) and set operations (such as *intersect*, *union*, *in*), are unique to and common in a multi-table scenario. Furthermore, such multi-table operations often result in a tabular answer and they necessitate table generation capabilities of the QA model.

Figure 3.1 depicts an example of a question involving two tables, *I would like to know the zip code of trips taken above 200 with humidity below 70*, and its associated input tables, *Weather* and *trip*. A multi-table QA model is expected to disambiguate records from different tables (the question phrase *zip code of trips* grounds the column *zip_code* of Table *trip*; the question phrase *humidity below 70* grounds column *min_humidity* of Table *Weather*), learn associations among inter-table columns (*zip_code* in both tables) and intra-table columns (*min_humidity* and *zip_code* in the *Weather* table), and finally compute the required operations (*intersect*, *count*) and generate the tabular answer.

Recent work on tabular QA can be categorized into two major directions: (i) Semantic parsing-based techniques [16, 157, 224] which have been the dominant approach to answering multi-table complex questions. Such methods transform a natural question to a logical form, which is used to query a relational database to extract the answer. However, these techniques are restricted to relational databases and cannot be applied to tables from other sources such over web tables, tables in text documents, and any non-normalized tables. Additionally, they require expensive and expert human annotations [105, 208] formulating SQL queries from natural questions. (ii) Modeling the problem as a sequence generation/classification task [25, 64, 83, 121, 131, 141, 149, 207, 216, 227], where an end-to-end trained neural model is responsible for not only question/query understanding but also table reasoning. Existing end-to-end neural models are either classification-based [64, 227], where the model detects the answer span and classifies one table operator associated with the span, or they are sequence generation-based [121, 141, 216], where the model generates the answer as a span of text in an auto-regressive manner.

[1]Code and data are at: `https://github.com/kolk/MultiTabQA`

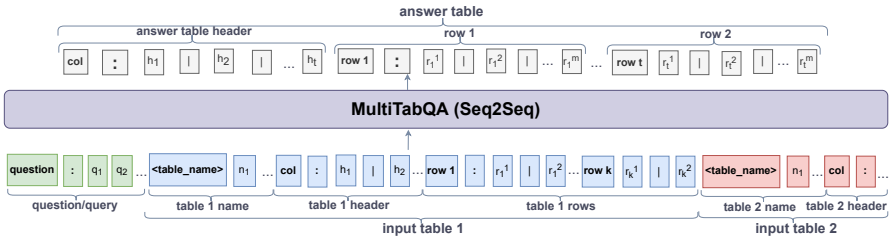


Figure 3.2: Architecture of MultiTabQA model. Given a natural language question/SQL query and the associated tables as an input sequence, the seq2seq model performs tabular reasoning and generates a tabular answer. Start of an input table is identified with keyword <**table_name**> which also indicates that the next tokens comprises the table name. **col:** indicates that the next tokens are table headers. Rows in a table are identified with keyword **row i:**, columns are separated by —.

Our work focuses on the latter direction of research. We train a neural model to mimic a semantic parser and generate the answer. A clear distinction of our work compared to existing end-to-end models is that our proposed model, MultiTabQA, does not operate in the constrained setting of a single input table, but can accommodate one or more tables in the input and the associated multi-table operators. Additionally, MultiTabQA performs the task of structured table generation, which imposes structure aspects to the generated output, such as table schemas, alignments of rows and columns, and relationships between column-headers and column values. Generating structured tables as output requires table-specific evaluation metrics, which we define and use to evaluate the generated tables. To effectively train the model, we generate a pre-training dataset with multi-table SQL queries and tabular answers built over an existing semantic parsing dataset, Spider [208]. Our dataset consists of $132,645$ samples comprising SQL queries, associated natural language questions, input tables, and tabular answers. To the best of our knowledge, this is the first work to address the task of multi-table QA and generate tabular output.

Our main contributions can be summarized as follows:

(1) We fill in the gap of existing tabular QA methods, which operate only on single tables, by proposing a new task of answering questions over multiple tables. Our work increases the breadth of question types that can be handled by single tabular QA methods.

(2) Our proposed multi-table QA model generates structured tables imposed by multi-table operations. Table generation introduces generation challenges such as maintaining row-column alignment, table-header generation, etc.

(3) We release a multi-table pre-training dataset comprising of $132,645$ samples of SQL queries and tabular answers.

(4) We introduce table generation metrics that capture different levels of granularity and strictness to evaluate our proposed model.

3.2 Related Work

Tabular QA is a research direction in the broader topic of table understanding [80, 185] in natural language processing. Recent advances in table representation [41] and pre-training [24, 122], table fact verification [52, 226], table numeric reasoning [181, 225], table-to-text generation [2], text-to-table generation [197], table summarization [20, 77, 216], and table question answering [25, 64, 83, 121, 131, 141, 149, 207, 216, 225, 227] have shown the adaptability of language models to table processing.

3.3 Methodology

We frame multi-table question answering as a sequence-to-sequence task and train an auto-regressive transformer encoder-decoder model to generate the tabular answer. Given a question Q consisting of a sequence of k tokens $q_1, q_2, \ldots, q_k$ and a set of N tables, $T_N = \{t_1, t_2, \ldots, t_n\}$, the goal of the multi-table QA model is to perform chains of *operations* over T_N, constrained by Q, and generate a table T_{out}. The model always generates a table, T_{out}, which can be single celled for scalar answers, single rowed or columned for list-based answers, and multiple rows and columns for tabular answers. In all cases, the model also generates column headers revealing important semantics associated with the generated values.

Training approach We follow a curriculum learning approach [12] by sequentially increasing the complexity of tasks to train MultiTabQA. The first stage of training is a pre-training step where the training objective is two-fold: (i) learn to generate correct tabular answers from SQL, and (ii) understand the associations between related input tables. The final training stage is fine-tuning where the model learns to understand natural language questions with their inherent ambiguity, in addition to retaining its ability of reasoning over tables and generating a tabular answer. We discuss the training process in detail in Section 3.5.

Model input/output The input to the model is a sequence consisting of the query or the natural language question, followed by a sequence of input tables, represented by the table name and the corresponding flattened table. Table names are important for disambiguating tables in multi-table QA setting. Specifically, the input sequence is represented as $question\ [table_1\ rep]\ [table_2\ rep] \ldots [table_n\ rep]$ where $[table_i\ rep]$ is the representation of the i^{th} table. As depicted in Figure 3.2, the i^{th} table is flattened in row-major format and represented as

$$\text{<}\textbf{table\_name}\text{>:}\ n_1\ n_2\ \mid \textbf{col:}\ h_1 \mid h_2 \mid \ldots \mid h_k \quad \textbf{row 1:}\ r_1^1 \mid \ldots \mid r_1^m \ldots \textbf{row k:}\ r_k^1 \mid \ldots \mid r_k^m,$$

where $n_1, \ldots, n_2$ is the sequence of table name tokens, h_j is j^{th} column header, r_m^i is the i^{th} row and m^{th} column cell. The boldface words are keywords specifying the semantics of the next tokens. The output of the model is also a flattened table in row-major format, i.e., $[table_{ans}\ rep]$, but without a table name. As depicted in Figure 3.2,

the generated table, $[table_{ans}\ rep]$, is of the form:

col: $h_1 \mid h_2 \mid \ldots \mid h_k$ **row 1:** $r_1^1 \mid \ldots \mid r_1^m$
row 2: $r_2^1 \mid \ldots \mid r_2^m \ldots$ **row k:** $r_k^1 \mid \ldots \mid r_k^m$.

3.4 Dataset

To effectively train a multi-table QA model, the dataset needs to cover three aspects: (i) multi-table context, (ii) tabular answers, and (iii) natural questions. Given the absence of large-scale datasets covering all three aspects, we transform existing semantic parsing and single-table QA datasets to focus on a single aspect before training with samples covering all three aspects.

3.4.1 Single Table Pre-training Dataset

One of the sub-tasks of pre-training is to generate tabular answers. We hypothesize that tuning the model to generate tables may lead to a warm-start and better convergence in a multi-table QA setting. To enable such experiments, we modify the large-scale single-table QA Tapex pre-training dataset [121] to accommodate tabular answers. The dataset contains $1,834,419$ samples of query, input table and factoid answers. The tables in the dataset are not named, as there is no need for table disambiguation in a single table setting. The SQL queries are semi-formal (do not contain the `FROM` clause with a table name) and cannot be used to query a real SQL database. We insert a placeholder table name in the queries and the corresponding input tables to extract the tabular answer from the database. Transforming the factoid answers to tables leads to single-celled or single-rowed tables. The modified dataset helps the model to understand simple tables and reason over semi-formal queries to generate simple tables.

3.4.2 Multi-table Pre-training Dataset

We develop a multi-table pre-training dataset over the database of Spider [208]. Spider is a cross-domain complex semantic parsing dataset for text-to-SQL translation. It consists of $10,181$ questions and $5,693$ SQL queries. The questions are over 200 databases of multiple tables covering 138 different domains. The training, development, and test splits do not contain overlapping databases to test a model's generalizability to new databases.

We first adapt the existing samples of Spider for our task. We use the ground-truth SQL queries of Spider as input query for pre-training over multiple tables. We automatically extract all input table names from the SQL query and retrieve the input tables[2] from the relational database. The query, extracted table names, and retrieved tables are inputs to our multi-table QA model. We extract the answer table with the SQL query by querying the relational database. Answer table headers reveal important semantics of the associated column values such as the numeric operation (*average*, *sum*, etc.), numeric scales (million, thousand, kms, meters, etc.), or entity facets (name, date,

[2]We use SQLite3 and pandas for extracting tables.

etc.). This process generates 3816 samples comprising *query*, *question*, *table_names*, *tables*, and *answer*.

We further augment the modified Spider dataset with $132,645$ samples of synthetic queries. This leads to an augmented multi-table pre-training dataset of $136,461$ unique training samples comprising of 3816 Spider samples and $132,645$ synthetic samples. The validation set comprises of 536 samples from the Spider validation set pre-processed as described above to adapt to our task.

Existing work on semantic parsing [184, 210] has utilized hand-crafted templates to generate large-scale corpora of synthetic queries, but is constrained in their coverage with no multi-table operations [184] or limited coverage with no table *joins* and lacking diversity in *set* operations [210]. This motivates us to generate our augmented pre-training dataset for multi-table QA using multi-table SQL templates.

Our synthetic queries are generated from 45 manually crafted templates over the Spider database and hand-crafted rules for operation types. The query templates have placeholders for aggregation, relational operations, table name, and headers that are randomly assigned during the query generation process. For example, to generate multi-table *join* queries, we instantiate the templates by randomly choosing tables from a database with at least one common header. For *set* operations, all tables participating in a multi-table query require all table headers to match. We design SQL templates in increasing order of complexity starting with simple SQL templates and progressively adding components, increasing its complexity. For example, for single-table queries, we use the simplest template "*SELECT * FROM* {*table_name*}" whereas for multi-table templates such as *joins*, the simplest template is "*SELECT T1.*{*table1_cols*}*, T2.*{*table2_cols*}* *FROM* {*table_name1*}* as T1 JOIN* {*table_name2*}* as T2 ON T1.*{*common_col*}* = T2.*{*common_col*}". We progressively add SQL components such as aggregations, *where* conditions, *group by* and *having* clauses to generate templates of increasing complexity. This process results in 14 templates for *joins*, 4 templates for each set operation: *intersect*, *union* and *except*. To avoid catastrophic forgetting for single table queries, we also instantiate 14 single-table queries with increasing complexity.

Quality control We ensure the correctness of the synthetic samples by discarding SQL queries that executes to an error or empty table. We also apply the process on the modified Spider, Atis and GeoQuery data to discard SQL query and the corresponding natural language question to ensure that all questions are answerable.

3.4.3 Multi-table QA Dataset

We fine-tune and evaluate our model on the natural language questions of semantic parsing datasets: Spider, GeoQuery [212], and Atis [30, 165]. GeoQuery is a semantic parsing dataset to query into a database of United States geography.[3] Atis is a semantic parsing dataset[4] with a collection of $4,379$ questions, corresponding SQL queries, and a relational database to a flight booking system [73]. Similar to the Spider dataset processing described in Section 3.4.2, we first extract the input table names from the

[3]This data is made available under under GPL 2.0 license.

[4]This data is made available under MIT license.

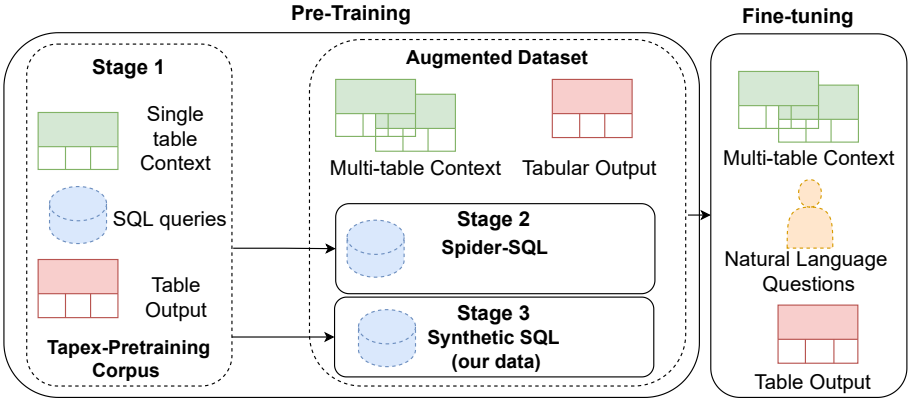


Figure 3.3: Four stage training procedure. The first three stages are pre-training, followed by fine-tuning.

available SQL queries and query the relational database for the input tables.[5] We also extract the tabular answers using the SQL queries. We discard any samples that executes to an error or an empty table. We use the corresponding natural language question for each SQL query as the user utterance for fine-tuning. This results in $6,715$ training samples and 985 validation samples for Spider. We also process the 600 GeoQuery samples provided in [73] to create a subset of 530 training samples, 49 validation samples, and 253 test samples. We process and generate an Atis subset of 384 training samples, 45 evaluation samples, and 86 test samples. We discard Atis queries with very large input tables (with $> 10,000$ rows). This restriction enables us to correctly evaluate question answering capabilities of a model by ignoring samples with truncated input sequences including entire input tables from the second table onward. The truncation of tables leads to incorrect answers for any numeric operation such as *average*, *intersect* and the evaluation scores would no longer reflect reasoning capabilities of the model.

3.5 Training

We follow a curriculum learning approach by sequentially training the model on sub-tasks of increasing complexity as depicted in Figure 3.3. Broadly, we first pre-train the seq2seq model to mimic a SQL parser and further fine-tune it on the downstream multi-table QA task. Pre-training the model on unambiguous SQL queries leads to better convergence and warm-start for the closely related downstream multi-table QA task. We further segregate the pre-training by first addressing the simpler sub-task of generating tables from single table queries. This is immediately followed by pre-training on multi-table query answering where complex SQL queries are utilized to train the model to learn multi-table associations from unambiguous complex queries, reason over the tables, and generate tabular answer. The final stage of training is the downstream

[5]We preprocess the Atis and GeoQuery data samples available at `https://github.com/sriniiyer/nl2sql/tree/master/data`.

multi-table QA from natural language questions. Natural language introduces ambiguity, ellipses and co-references which increases complexity and is thus the final stage of training. For each stage, we choose the model with the best table exact match accuracy on the corresponding validation set, defined in Section 3.6, as the initialization for training the next stage.

3.5.1 Pre-training

Pre-training of MultiTabQA is conducted in two stages in a curriculum learning fashion: Stage 1 is single table QA where the model learns to generate tabular answers from relatively simple SQL queries. Stage 2 is multi-table QA where the model trained in Stage 1 is further tuned for multi-table SQL QA.

Stage 1 We first train MultiTabQA on the task of generating tables from SQL queries over single tables. The tabular answer to be generated is simple and single-columned. For this stage, we use the modified Tapex pre-training corpus described in Section 3.4.1. We train the model on $1,834,419$ samples for two epochs. This stage provides a good initialization for multi-table QA in the next stages.

Stage 2 + Stage 3 We further pre-train the model on multi-table QA. For this, we tune our model on SQL queries from the modified Spider and synthetic dataset. We tune with only the modified Spider SQL samples *Stage 2*, and tuning with only the synthetic dataset *Stage 3*. We utilize the larger augmented dataset comprising of the modified Spider SQL (Stage 2) and our synthetic samples (Stage 3) as described in Section 3.4.2 to train the final pre-trained model for 30 epochs. We call this setting *Stage 2+3*. We compare these three multi-table pre-training settings in Section 3.7.

3.5.2 Fine-tuning

The final stage of training is fine-tuning the pre-trained model on natural language questions. Natural questions are ambiguous compared to formal SQL and used at the last stage of training. We fine-tune the pre-trained model on the $6,715$ natural questions, extracted input and output tables for Spider as described in Section 3.4 and evaluate on 985 samples of the validation set. To observe the performance of the pre-trained model on out-of-domain database tables, we also fine-tune the pre-trained model on Atis and GeoQuery datasets. For all the fine-tuning datasets, we train for 60 epochs.

3.6 Evaluation Metrics

While denotation accuracy has been widely used in semantic parsing [16, 157, 224], it is not directly applicable for our task where tabular input encoding, reasoning, and generation are performed by the same model. Evaluating the answer table requires matching not only the generated values, but also the table structure. Moreover, tables store factual information such as named entities, dates, numbers, etc. in an ordered manner. This makes lexical metrics measuring surface form overlap more suitable

Data-set	Model	Table	Row EM (%)			Column EM (%)			Cell EM (%)		
		EM (%)	P	R	F1	P	R	F1	P	R	F1
Spider	T	18.99	17.28	19.83	18.27	19.75	19.39	19.57	23.15	27.71	25.03
	M	**25.19***	**22.88**†	**24.64***	**23.70***	**26.86***	**26.76***	**26.81***	**28.07**†	**31.23***	**29.55***
GeoQ	T	39.84	22.43	30.74	24.89	39.48	39.76	39.62	21.98	30.88	24.67
	M	**52.22***	**72.39***	**46.90***	**41.38***	**52.10***	**52.22***	**52.16***	**37.16**†	**46.92***	**41.33***
Atis	T	72.20	**57.07**†	57.69	**55.08**	**72.20**†	72.20	72.20	**57.07**†	57.69	**54.48**
	M	**73.88**†	38.29	**92.19***	54.36	69.55	**75.24**†	**72.29**	38.16	**92.56***	54.16

Table 3.1: Average scores of models fine-tuned on 5 different seeds with Multitable-Natural Questions (NQ) datasets. T is short for `tapex-base`, which is used as baseline while M is short for `MultiTabQA`, which is our fine-tuned model. Table EM indicates table exact match accuracy. For all other table units (row, column, and cell), P is Precision, R is Recall, and F1 is F1 score for exact match metric. A * denotes a significant difference at $p < 0.005$ and a † denotes a significant difference at $p < 0.05$ using a t-test.

than semantic metrics measuring the underlying meaning of paraphrased sequences. Moreover, table components such as rows, columns, and cells are standalone units which capture different levels of semantics and relationships with the surrounding table component. For example, rows capture data records while columns capture the features of each record. Cells capture the lowest level of self-contained facts and require complete match with the target. For example, a cell with the entity "United Kingdom" should not be partially matched with the predictions "United Nation", "United" or "Kingdom". Similarly, a numeric value such as "123.45" should not be partially matched with "12.45", "23.45" or "12". Numeracy poses a challenge to seq2seq models [145, 148], especially in the extrapolation setting where semantic match of unseen numbers may not be an ideal. Considering all these factors, we focus on the lexical match to measure model effectiveness.

Table exact match We define *table exact match Accuracy (Table EM)* as the percentage of predicted tables that exactly matches the target tables. Table exact match evaluates ordering of rows, columns, and table headers and the exact lexical matching of the table values. It is a strict binary measure that treats partial matches as incorrect. However, many queries do not impose ordering among columns or rows, and strict table exact match may not be the ideal indication of model efficacy. To measure partial correctness, we treat rows, columns, and cells as units at varying levels of granularity, which have ordered associations among the values within the unit. We evaluate partial correctness with exact match of rows, columns, and cells.

Row exact match To relax the strict criterion of table exact match, we first measure correctness on table rows. Row exact match does not consider the ordering of rows in the generated table but requires the ordering of values within the row. We define a correctly generated row to be a predicted row that exactly matches any target rows in

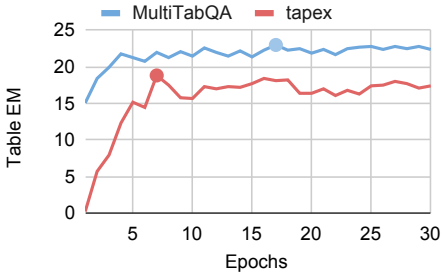


Figure 3.4: Validation table exact match scores of `MultiTabQA` vs. `tapex-base` on Spider evaluation set natural language questions during fine-tuning. The points are highest validation scores for each model.

the target table. *Row exact match precision* is the percentage of correctly generated rows among all the predicted rows in the evaluation dataset. *Row exact match recall* is the percentage of correctly generated rows among all the target rows in the evaluation dataset.

Column exact match Unlike rows, which represent records in relational databases, columns represent attributes where column header provides semantic meaning to the values. Hence, a correct column is defined as a generated column that first exactly matches a target column header and further the column values. Column exact match measures the ordering of values within a column. *Column exact match precision* is the percentage of columns generated correctly among all columns generated in the evaluation set. *Column exact match recall* is the percentage of columns generated correctly among all target columns in the evaluation set.

Cell exact match *Cell exact match* is the most relaxed measure of model efficacy at the lowest level of granularity (cells) where table structure is not measured. A cell is correct if it matches any cell in the corresponding target table. *Cell exact match precision* is the percentage of correctly predicted cells among all predicted cells in the dataset. *Cell exact match recall* is the percentage of cells predicted correctly among all target cells in the dataset.

3.7 Experimental Setup and Results

We use `tapex-base` [121] as the base model for all our experiments. `tapex-base` is a single table question answering model (140M parameters) trained to approximate table reasoning by pre-training to mimic an SQL parser. For both the pre-training and fine-tuning process, we use a batch size of 8 and a gradient accumulation of 32 to emulate an effective batch size of 256, a learning rate is $1e^{-9}$. The maximum sequence length of both the encoder and decoder is set to 1024. We run all our pre-training experiments on four A6000 48GB GPUs and fine-tuning on one A6000 GPU.

We observe from Figure 3.4 that the three-stage pre-training leads to a warm-

Pre-training stages	Query type	Table	Row (%)			Column (%)			Cell (%)		
		EM(%)	P	R	F1	P	R	F1	P	R	F1
2	SQL	21.46	18.60	18.88	18.74	21.98	21.90	21.94	24.19	25.89	25.01
1+2	SQL	20.52	14.13	20.06	16.58	18.87	20.87	19.82	19.24	25.83	22.05
1+2+3	SQL	**29.10**	**23.15**	**25.62**	**24.32**	**31.66**	**31.50**	**31.58**	**29.95**	**32.92**	**31.36**
2	NL	19.41	16.51	19.48	17.87	20.13	20.11	20.12	21.12	26.55	23.52
1+2	NL	20.12	11.67	21.09	15.03	19.54	19.97	19.76	16.26	29.22	20.90
1+2+3	NL	**24.49**	**24.95**	**24.87**	**24.91**	**26.80**	**26.91**	**26.86**	**28.44**	**31.06**	**29.69**

Table 3.2: Ablation on datasets in our multi-stage pre-training processes for 1 run of experiments. The two sections show scores for different question types: SQL queries (top) and natural language (NL) questions (bottom). In a section each row shows a training process with different stages: Pre-training on Stage 2, pre-training on Stages 1+2, and all pre-training Stages 1+2+3. Table EM is table exact match accuracy; P is Precision; R is Recall; and F1 is F1 score for exact match of row, column, and cell.

start for fine-tuning and better convergence compared to the baseline `tapex-base`. For our experiments, we compare the effectiveness of the MultiTabQA model with fine-tuned `tapex-base` on the $6,715$ natural questions from Spider. The fine-tuned `tapex-base` acts as a baseline for studying the adaptability of the state-of-the-art single table model to a multi-table setting. We report the mean scores of 5 training runs initialized with different seeds in Table 3.1. We conduct statistical significance test (t-test) on the mean scores of the 5 runs and report the significance with $p < 0.05$ and $p < 0.005$. We observe that our multi-stage training process leads to improvement in scores on all table exact match accuracy across all datasets compared to fine-tuned `tapex-base`. The difference in table exact match is highest for GeoQuery where MultiTabQA outperforms `tapex-base` by 12.38%, Spider by 6.20% and Atis by 1.68%. For F1 and Recall scores on row, column and cell exact match, a similar pattern is observed where MultiTabQA outperforms `tapex-base` on all datasets. MultiTabQA outperforms `tapex-base` by 5.43% on row F1, 7.24% on column F1, and 4.52% on cell F1 for Spider. On GeoQuery, MultiTabQA outperforms by 16.49% on row F1, 12.54% on column F1 and 16.66% on cell F1 scores. All results on Spider and GeoQuery are significant with a p-value less than a critical value of 0.05 indicating strong evidence that MultiTabQA is a superior model. On Atis, we observe that MultiTabQA underperforms on precision but outperforms on recall by a large margin. The difference in recall is larger than precision, indicating that MultiTabQA generates more target rows, columns, and cells of Atis correctly (higher recall) and hallucinates spurious rows and cells (lower precision). However, the F1 scores are better for MultiTabQA. `tapex-base` is unable to correctly generate target rows, cells, and columns (lower recall), but the few generated ones are correct (higher precision). The low number of test samples (85) of Atis and variations in the hallucinations in different runs makes the precision scores statistically non-significant. However, the recall scores provide very strong evidence ($p < 0.005$) of the superiority of MultiTabQA

in generating correct table units compared to `tapex-base`.

Qualitative analysis Multi-table QA models must perform numeric reasoning, understand multi-table schemas and comprehend natural language. A success case also depicts this. For the question *how many likes does kyle have?* with 2 input tables:

highschooler

id	name	grade
1510	jordan	9
...	...	...
1934	**kyle**	12
1661	logan	12

likes

student_id	like_id
1689	1709
...	...
1501	**1934**
1934	1501

with

target:

count(*)
1

and

prediction:

count(*)
1

,

MultiTabQA identifies inter-table association of column *id* of table *highschooler* and column *student_id* of table *likes*. It correctly disambiguates the lexical occurrence of 1934 in columns *like_id* and *student_id* and correctly performs *count*.

A failure case also illustrates the challenges: for the question *find the average weight for each pet type* with input table:

PetID	PetType	pet_age	weight
2001	cat	3	12.0
2002	dog	2	13.4
2003	dog	1	9.3

with

target:

avg(weight)	PetType
12.0	cat
11.35	dog

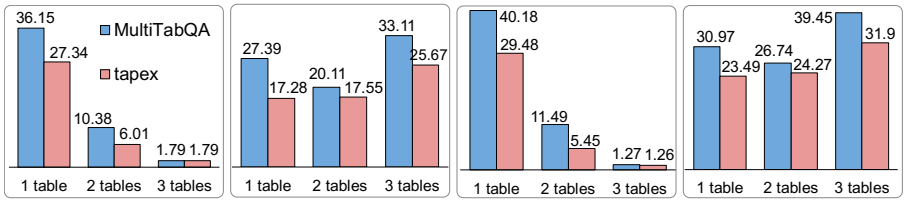
and

prediction:

PetType	avg(weight)
cat	12.0
dog	13.4

,

MultiTabQA swaps the ordering of the 2 columns and fails to compute *average* leading to an incorrect measure by table exact match. The column, row and cell metrics (precision, recall and F1) measure correctness of individual table units without measuring the ordering. Column metrics measure predicted column *PetType* as correct and *avg(weight)* as incorrect without measuring ordering of the 2 columns. Row *cat — 12.0* is measured as correct, while *dog — 13.4* is measured as incorrect without measuring the ordering among them. Out of the 4 target cells, *cat*, *dog*, *12.0* are measured as correct.



(a) Table EM Acc (b) Row EM F1 (c) Column EM F1 (d) Cell EM F1

Figure 3.5: Evaluation results on Spider evaluation samples segregated by number of input tables. Acc is the Accuracy and F1 is the F1 score.

Impact of the number of input tables The number of input tables increases the complexity of the questions and directly impacts the effectiveness of the models. We segregate evaluation on Spider validation set on the basis of number of input tables and compare the results to study the impact of input table number. We observe from Figure 3.5 that the effectiveness reduces as the number of tables increases for both MultiTabQA and `tapex-base`. However, MultiTabQA fares better than `tapex-base` when the number of input tables increases. MultiTabQA generates whole tables, rows, columns, and cells better than `tapex-base` as observed in Figure 3.5a, 3.5b, 3.5c and 3.5d. The gain of MultiTabQA in table exact match for one-table context is around 8.81%, for two-tables context around 4.37%, and it performs similar to `tapex-base` for three-tables context. It also has a significant higher score on rows, columns and cells, on both single and multi-tabular context.

We also observe that while the column and table EM decreases dramatically when using several tables (Figure 3.5a and 3.5c), the row and cell EM does not (Figure 3.5b and 3.5d). This indicates that MultiTabQA can generate rows and cells as effectively in single and multiple input tables settings but fails to do so for columns and consequently for the whole table. This is due to the fact that certain columns in the answer, particularly ones with numbers such as floats, are challenging to generate. The errors from the incorrect columns propagate and are accumulated in the table EM leading to a significant drop in performance for multi-table queries.

Ablation on training stages We perform ablation on the pre-training stages to analyze the contribution of each dataset. The simplest setting is to pre-train with Spider SQL queries, i.e., Stage 2. To evaluate the effectiveness of single table Tapex pre-training samples, the next setting comprises stages 1 and 2, i.e., pre-train with Tapex pre-training and Spider SQL dataset. The final comparison is with the three-stage pre-training as described in Section 3.5.1. The results for one run of the experiments are displayed in Table 3.2. We observe that table exact match is highest for both pre-training and fine-tuning for the three-stage training. Stage 2 fares better than Stage 1+2 on table exact match, and generally has better precision and F1 scores, but lower recall. The three-stage pre-training with our synthetic data augmented with Spider outperforms the other settings and confirms the effectiveness of our synthetic data samples in boosting model efficacy.

3.8 Conclusion

In this chapter, we have proposed a new task of multi-table question answering without intermediate logical forms to fill the gap of existing end-to-end table QA research which focused only on single-table QA. We release a pre-training dataset of $132,645$ samples to effectively train a seq2seq model. We fine-tune and evaluate our model, MultiTabQA, on natural language questions of three datasets: Spider, GeoQuery and Atis, to test the efficacy in a multi-table setting. As many multi-table questions result in tables, we train the model to generate tables. This necessitates table-specific metrics at various levels of granularity, which we design to evaluate the effectiveness of our model. We demonstrate that such metrics are insightful in understanding model behavior. MultiTabQA outperforms existing state-of-the-art single table QA model fine-tuned to adapt to a multi-table QA setting.

This chapter answers RQ2 which inquires *How can we leverage multiple tabular contexts to perform complex tabular reasoning to address user needs?* The development of a large-scale pre-training dataset aids in training models capable of handling complex multi-table questions. The pre-training dataset, the proposed training methodology that outperforms regular fine-tuning, and the development of task-specific evaluation metrics completes the multi-tabular question answering setup for addressing user needs. This setup provides the first step in the direction of reasoning over multi-tabular context and addresses RQ2.

Chapter Appendices

3.A Limitations

Our synthetic pre-training dataset was automatically generated from manual templates, which inspite of dataset creation scalability and low cost, may limit the diversity of the generated SQL queries. Our model, MultiTabQA, requires improvement in numeracy understanding and numeric operations. Real numbers are especially challenging, and the model may not be able to correctly generate all the digits of the number correctly, rendering the generated cell incorrect. Furthermore, large input tables pose a challenge as the input sequence may get truncated beyond the model's maximum sequence length. This has practical limitation in the size and number of input tables which the model can accommodate before truncation which leads to incorrect answers.

3.B Ethical Considerations

The task and model proposed in the paper is aimed at broadening the scope of the TabularQA research. All the datasets used in this research, apart from our synthetic data, are publicly available in peer-reviewed articles and are referenced in this paper. The synthetic SQL dataset we release was generated over a standard benchmark database which has been annotated by 11 Yale students, as mentioned in the original paper. Our synthetic samples use templates annotated by the authors of this work and do not use any user-specific data or information. We will be providing open access to our datasets for use in future research under the MIT License. All datasets, including the synthetic pre-training dataset and all datasets adapted for multi-table QA will be released. Our model is built over `tapex-base` which in turn has been trained over `bart-base`. Our work did not explicitly handle any bias which exists in the aforementioned pre-trained models.

4

QFMTS: Generating Query-Focused Summaries over Multi-Table Inputs

The previous chapter (Chapter 3) addresses multi-table question answering task where the neural model addresses complex queries over multiple tables and generates a structured table as the user response. In this chapter, we address RQ3 by studying query-focused summary generation over multiple tables, where the neural model generates a fluent and factual summary in response to the user query. Prior work on query-focused summary generation focuses only on single tables as the knowledge source, whereas real-world queries are often complex spanning across multiple tables. To address this limitation of prior work and to provide a human-readable summary in response to the user question, this chapter introduces the task of generating summaries over multiple tables called QFMTS. In addition to formalizing the task, this chapter focuses on the creation a dataset to aid the task, designing a methodology for effective multi-table reasoning, and analyzing summarization metrics to evaluate various aspects of query-focused multi-table summarization, such as faithfulness, completeness, and fluency.

4.1 Introduction

Table summarization involves the distillation of information from tabular data into a succinct format, typically a clear and human-readable description or textual table summary. This process aims to capture the key insights or trends encapsulated within the table, allowing for easier comprehension and interpretation by humans [22, 104, 118, 123, 188]. Traditional methods of table summarization take a single table as input and produce a fixed textual summary [77, 216]. However, the fixed nature of table summaries generated by these traditional approaches often falls short of meeting users'

This chapter was published as W. Zhang, V. Pal, J. Huang, E. Kanoulas, and M. de Rijke. QFMTS: generating query-focused summaries over multi-table inputs. In U. Endriss, F. S. Melo, K. Bach, A. J. B. Diz, J. M. Alonso-Moral, S. Barro, and F. Heintz, editors, *ECAI 2024 - 27th European Conference on Artificial Intelligence, 19-24 October 2024, Santiago de Compostela, Spain - Including 13th Conference on Prestigious Applications of Intelligent Systems (PAIS 2024)*, volume 392 of *Frontiers in Artificial Intelligence and Applications*, pages 3875–3882. IOS Press, 2024. doi: 10.3233/FAIA240951. URL https://doi.org/10.3233/FAIA240951.

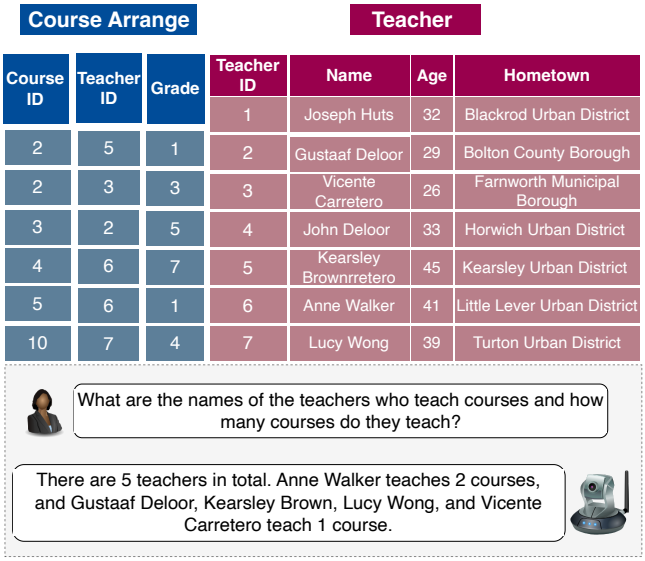


Figure 4.1: An example of query-focused multi-table summarization. Summarization models should combine the information from the two tables to produce a summary tailored to the query.

information and quality requirements adequately. The quality of a summary plays a pivotal role in its utility. For instance, in business contexts, table summaries often play a critical role in shaping future business strategies, with their quality directly influencing the judgments of decision-makers. Poor-quality summaries may fail to capture essential aspects, leading to a misrepresentation of the data and conveying an inaccurate perspective.

To enhance the effectiveness of conventional table summarization, Zhao et al. [221] proposes to leverage textual queries as a starting point to generate query-dependent table summaries aligned with users' information needs. Their approach accepts a single table and a user-specified textual query as inputs, generating a description or statement tailored to the query's focus as output. However, in Zhao et al. [221], the authors presume that fulfilling the information requirements of a given query depends solely on data from a single table. This assumption overlooks the complexity of real-world scenarios, which frequently demand information from multiple data sources. Consequently, the intricate nature of real-world queries highlights the necessity for integrating information from multiple tables to address them effectively.

Let's consider a practical scenario depicted in Figure 4.1 to understand how humans address complex real-world queries. For instance, the query "*What are the names of the teachers who teach courses and how many courses do they teach?*" entails two distinct information requirements – teachers' names and their course teaching details. While the `Teacher` table in Figure 4.1 provides the teachers' names, relying solely on this table is insufficient to generate a complete query-focused table summary. To completely address the query, we require data from another table, such as the `Course Arrange`

table in Figure 4.1, which contains information about course arrangements for teachers listed in the `Teacher` table. By performing table *join* (i.e., multi-table reasoning) and *count* (i.e., arithmetic reasoning) operations, we can determine the number of courses taught by each teacher. Thus, addressing such a common real-world complex query involves employing multi-table reasoning and arithmetic reasoning. This practical and challenging scenario remains largely unexplored, underscoring the necessity for further exploration and advancement in this domain.

Motivated by the above observation of how humans handle such a scenario and aiming to bridge this research gap, we propose a new method designed to tackle the aforementioned practical and challenging situation: query-focused multi-table summarization. Our proposed approach involves taking multiple tables and a user-defined query as inputs, aiming to generate a query-relevant textual summary based on the inputs. The proposed approach primarily comprises a table serialization module, a summarization controller, and a large language model (LLM). The flowchart illustrating our proposed method is provided in Figure 4.2 to aid comprehension. Additionally, in light of the absence of an existing dataset for our introduced task, and to validate the effectiveness of our proposed query-focused multi-table summarization method, we create a query-focused multi-table summarization (QFMTS) dataset. This dataset comprises $4,909$ query-summary pairs, each pair associated with multiple tables. For further details about our proposed dataset, please refer to Section 4.4.

In our comprehensive experiments, we assess the effectiveness of the proposed query-focused multi-table summarization method using our devised QFMTS dataset. The experimental findings demonstrate the superiority of our approach over baseline methods, shedding light on the challenges encountered by existing models in performing complex table reasoning to produce precise table summaries. To the best of our knowledge, we are the first to address the task of query-focused multi-table summarization.

Contributions Our primary contributions in this chapter can be summarized as follows:

- **Introduction of query-focused multi-table summarization:** We introduce a novel method tailored for query-focused multi-table summarization, aiming to overcome limitations present in current approaches that predominantly target single-table summarization. Our proposed method comprises a table serialization module, a summarization controller, and a large language model (LLM), offering a structured framework to tackle the complexities inherent in the task of query-focused multi-table summarization.
- **Development of a comprehensive dataset:** We develop a QFMTS dataset specifically tailored for the query-focused multi-table summarization task. The QFMTS dataset consists of $4,909$ query-summary pairs, each intricately linked with multiple tables. It will serve as a valuable resource for researchers in this field, facilitating the exploration and validation of proposed methods in the future.
- **Extensive experiments:** We conduct comprehensive experimental validation of our proposed query-focused multi-table summarization method using the introduced QFMTS dataset. Our experimental results demonstrate its effectiveness

over baseline methods and provide insights into the challenges encountered in complex table reasoning for precise summarization.

4.2 Related Work

In this section, we briefly review recent developments in the related areas of table summarization and query-focused text summarization. We also highlight our unique contributions in contrast to these existing studies.

Table summarization Table summarization involves generating a concise and informative summary from a given table. To address this task, prior research [77, 123, 216] has primarily focused on summarizing the entire table without explicitly addressing users' specific information needs. However, in real-world applications, users frequently seek targeted information from table segments, underscoring the importance of generating personalized table summaries. Although Zhao et al. [221] introduced the initial human-annotated dataset for query-focused table summarization, their study restricts to single-table scenarios, lacking consideration of multi-table reasoning such as operations involving table *join* and *union*. In contrast, our work presents two primary distinctions: First, we leverage LLMs to assist in the data annotation; second, our proposed method addresses complex queries that necessitate the integration of information across multiple table contexts.

Query-focused text summarization Query-focused text summarization is designed to generate textual summaries based on a specific query and a collection of relevant documents. This research field has been extensively investigated with textual inputs [32]. Traditional studies have faced challenges due to the scarcity of large-scale datasets, often resorting to distant supervision signals from adjacent fields, such as generic summarization [199, 200] to enhance summarization performance. Additionally, recent efforts have been directed towards creating synthetic large-scale datasets [99, 102, 126]. Despite these advancements, the application of query-focused summarization to tabular data remains relatively unexplored [221], especially in scenarios where queries span multiple tables. This study seeks to bridge this gap by exploring the efficacy of query-focused summarization in multi-table contexts.

4.3 Methodology

This section formulates the task of query-focused multi-table summarization and details our proposed approach. As illustrated in Figure 4.2, our approach comprises two primary components: a table serialization module and a summarization controller.

Task formulation Query-focused multi-table summarization, denoted as QFMTS, involves generating a coherent and informative summary aimed at addressing a user query across multiple tables. Specifically, given a natural language query q and a set of input tables $\mathcal{T} = t_1, \ldots, t_k$, a query-focused multi-table summarization model

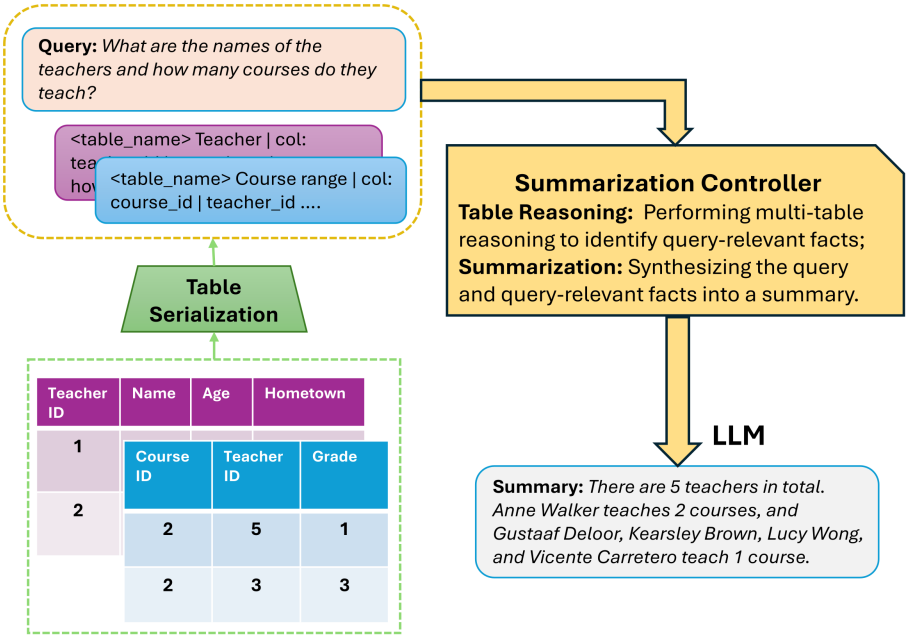


Figure 4.2: Overview of the proposed approach. Our approach integrates a table serialization module with a summarization controller. Initially, a set of tables is transformed into textual representations through the table serialization module. Subsequently, the summarization controller directs an LLM to perform table reasoning and produce a summary specifically tailored to a given query.

systematically engages in table-based reasoning across the contents of $\mathcal{T}$ related to q, aiming to produce a textual summary s that effectively resolves the user query while maintaining factual accuracy and comprehensiveness.

Table serialization Given that our approach is based on LLMs that process only textual data, it necessitates a table serialization to transform input tabular data into a textual format suitable for processing. In this work, we utilize a technique known as table linearization, which is widely used in table-to-text generation tasks [118, 121, 141, 149]. This technique transforms a table into a textual, sequential format using designated sentinel words. Specifically, a table identified by its name, $name$, consisting of m rows and n columns, is linearized as follows:

$$\begin{aligned} &\textbf{<table\_name>:}\ name \qquad \textbf{col:}\ h_1 \mid \ldots \mid h_n \\ &\textbf{row 1:}\ c_{1,1} \mid \ldots \mid c_{1,n} \ldots \quad \textbf{row m:}\ c_{m,1} \mid \ldots \mid r_{m,n}. \end{aligned}$$

where h_j and $c_{i,j}$ stand for the j^{th} column header and i^{th} row and j^{th} column cell, respectively.

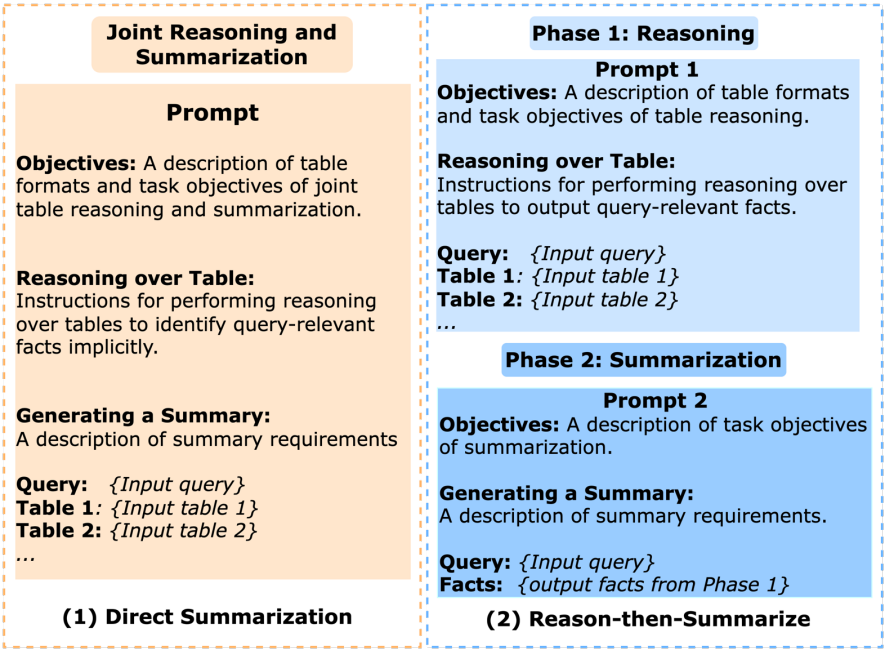


Figure 4.3: Two distinct approaches of the summarization controller: (1) Direct Summarization (2) Reason-then-Summarize. The former tackles the proposed task in an end-to-end manner. In contrast, the latter tackles the task across two independent phases.

4.3.1 Summarization Controller

Given a query and the corresponding linearized tables, our proposed summarization controller efficiently generates a comprehensive summary. This is achieved by integrating an LLM, such as GPT-3.5 [147], with a carefully designed prompting-based method. In this work, we explore two distinct methods: *direct summarization* and *reason-then-summarize*, which are depicted in Figure 4.3.

Direct summarization Direct summarization (DirectSumm) enables the LLM to jointly perform table reasoning and summarization in an end-to-end manner. We first present the linearized structure of table formats as outlined in Section 4.3, aiming to improve the LLM's understanding of tabular data. Given the query and linearized tables, we then prompt the LLM to perform table reasoning across multiple tables to implicitly identify query-relevant facts, i.e., these facts are not explicitly generated by the LLM. It is worth noting that the facts include numerals and entities that address the information needs of the query. Furthermore, we adopt chain-of-thought (CoT) style prompting methods [97, 194] using the directive *"Let's think step by step"*. This enhances the LLM's reasoning ability. Lastly, we instruct the LLM to synthesize the query with the identified query-relevant facts into a comprehensive summary, while adhering to constraints such as summary length.

Dataset	Statistics			Reasoning	
	# Examples	# Tables per example	# Words in summary	Numeric	Multi-table
ROTOWIRE [104]	4,953	1.0	337.1	✗	✗
SciGen [139]	1,338	1.0	116.0	✓	✗
NumericNLG [188]	1,355	1.0	94.2	✓	✗
QTSUMM [221]	7,111	1.0	68.0	✓	✗
QFMTS	4,909	1.83	58.5	✓	✓

Table 4.1: Comparisons between our dataset and existing table-to-text generation datasets. Our dataset is the only one tailored for query-focused multi-table summarization, supporting both numeric and multi-table reasoning.

Reason-then-summarize Inspired by the established *retrieve-then-generate* paradigm [75, 108], we introduce a novel approach termed *reason-then-summarize* (Reason-then-Summ). This approach tackles the proposed task in sequential phases. The first phase focuses exclusively on enabling the LLM to perform table reasoning across multiple tables based on the query and linearized tables. As a result, the LLM identifies and extracts query-relevant facts from the tables. In the second phase, we prompt the LLM to synthesize a comprehensive summary based on the query and the previously extracted query-relevant facts.

4.4 Dataset Construction

In order to validate the effectiveness of our proposed method, we create a novel dataset specifically tailored for this query-focused multi-table summarization task. This section details the dataset construction process, including data annotation and quality verification.

Source data We built our dataset on top of the Spider dataset [208]. The Spider dataset, originally designed for semantic parsing and text-to-SQL tasks, comprises $10,181$ natural language queries. These queries are paired with complex SQL queries and one or more tables from various relational databases. In this work, we construct input queries and tables from original textual queries and their corresponding tables to develop our dataset. We utilize SQL queries for data annotation, as detailed in the following subsection. Additionally, our analysis shows that over 50% of the queries in the original dataset corresponded to only a single table. This high proportion of single-table inputs raises concerns about the dataset's efficacy to provide sufficient challenges for multi-table scenarios. To address this, we selectively down-sampled these single-table examples to allow multi-table examples to predominate in the dataset. Since the test set from the original dataset is not publicly available, we randomly allocated 10% of the original training set to serve as our validation set, with the remaining 90% forming the new training set. The original validation set was repurposed as our test set.

This results in $3,871$ training, 430 validation, and 608 test examples, forming the basis for our dataset.

4.4.1 Data Annotation

LLMs as data annotators The objective of data annotation is to produce high-quality, comprehensive, and accurate summaries tailored to the associated input queries. Prior research in the field of query-focused summarization from single tables, such as QTSUMM [221], has predominantly been dependent on human experts to annotate summaries based on a input query and a input table. This reliance is primarily due to complex table reasoning, making summary annotation a challenging task. However, manual annotations are not only time-consuming but also costly. Recent studies [37, 60, 211, 222] have revealed that LLMs can match the annotation quality of crowd-sourced workers while being significantly more cost-effective and efficient. Motivated by these findings, we have employed LLMs as data annotators for our summary annotation, namely LLM_{Anno}. Although LLM_{Anno} does not yet match the table reasoning capabilities of human experts, we have designed a simplified table-to-text annotation task avoiding complex table reasoning. Specifically, for each textual query in our dataset, we follow Pal et al. [151] to first extract the *output table* by executing the corresponding SQL query over the associated rational database. It is worth noting that the execution table contains the query-relevant facts/entities required to construct a summary. Subsequently, rather than relying on input tables, we use the execution table as the basis for our annotation. We also have identified that relying solely on the output table frequently results in summaries that lack essential contextual information. Our observations indicate that this missing context can effectively be retrieved directly from the input query. For instance, consider the example output table below:

semester_name	semester_id
summer 2010	2

It lacks contextual information to confirm that the `Summer 2010` semester has the *most registered students* in response to the corresponding query *"What is the semester in which most students registered? Show both the name and the id."*. To address this, we have incorporated the given query as a supplementary input for summary annotation. This strategy enhances the overall comprehensiveness and relevance of the annotated summaries. In this work, we employ `gpt-3.5-turbo-0613` as LLM_{Anno} via the public OpenAI API.[1]

Instruction design The effectiveness of the instruction prompt is crucial in determining the quality of annotated summaries. To this end, we carefully design the instruction to ensure the summary quality, in which the structure of the instruction is shown in Prompt 4.4.1. The prompt comprises three components: a comprehensive annotation guideline, few-shot demonstrations, and input data. The annotation guideline outlines the expected discourse structure and the summary's length requirements. We have found that a more precise guideline significantly improves generation quality. To provide further clarity to LLM_{Anno}, we manually write summaries for a few examples as few-shot

[1] `https://platform.openai.com`

demonstrations (we use 5-shot in our experiments). Lastly, we include both the input query and the execution table directly in the prompt, specifically requesting LLM_{Anno} to write a summary based on these inputs. Similarly, we leverage table serialization as described in Section 4.3 to obtain the linearized execution table.

Prompt 4.4.1: Summary Annotation

Instruction: A comprehensive annotation guideline.

Demonstrations:
Few-shot human-written demonstrations.

Query: {*Input query*}
Table: {*Linearized execution table*}

4.4.2 Dataset Analysis

Our dataset comprises a total of $4,909$ examples, segmented into $3,871$ training examples, 430 validation examples, and 608 test examples. The dataset is composed of 32.8% single-table examples, 52.6% double-table examples, and 14.6% examples that incorporate three or more tables. Notably, more than 67% of the examples include at least two tables, highlighting the dataset's efficacy in facilitating research in multi-table scenarios. We present a comparative analysis of our dataset against existing table-to-text generation datasets in Table 4.1. Our dataset averages approximately two input tables per example, in contrast to the prevailing datasets which predominantly focus on single-table scenarios. The summaries from our dataset are sufficiently informative, with an average length of 58.5 words, aligning closely with the norms of existing datasets. Additionally, our dataset is characterized by a rich variety of operations. It includes basic numeric operations such as *sum* and *average*, and extends to more complex multi-table operations like *join* and *union*, which are absent in the QTSUMM dataset, as it is tailored exclusively towards single-table contexts.

4.4.3 Quality Verification and Control

To assess the quality of annotated summaries effectively, we develop a comprehensive evaluation encompassing both automated and manual evaluations. We define three primary desiderata for quality verification:

- **Faithfulness:** Each statement within the summary must be factually consistent with the facts presented in the execution table.
- **Completeness:** The summary should address all information needs in the user query, representing all facts from the execution table.
- **Fluency:** The summary needs to be articulate, clear, and easily understandable for human readers.

In our experiments, we utilize standard sequence similarity metrics to evaluate completeness. Given the lack of definitive metrics for faithfulness and fluency, we rely on human evaluations to assess these aspects. The results are presented in Table 4.2.

Split	Automated Evaluation	Human Evaluation†	
	Completeness* (%)	Faithfulness	Fluency
Training	91.45	0.98	4.73
Validation	91.75	0.98	4.81
Test	90.75	0.96	4.68
Total	91.48	0.97	4.74

Table 4.2: Quality evaluation of our dataset. * indicates that the completeness is quantified using average ROUGE-L recall scores. † indicates that we randomly sample 100 examples from the training, validation, and test set to measure faithfulness (0–1) and fluency (1–5), respectively.

Automated evaluation We initially assess the completeness of the annotated summary solely based on the execution table. However, as delineated in subsection 4.4.1, the execution table, by itself, proves inadequate for a comprehensive evaluation of summary completeness due to its limited contextual information. Consequently, we extend our evaluation by incorporating the query with the table, thereby enabling a more robust measure of completeness. Specifically, we extract facts, including numerals and entities, from the execution table. These facts are then combined with the query to construct a reference sequence. The completeness of the annotated summary is estimated against this reference sequence. In this work, we quantify the completeness using the lexical similarity metric ROUGE-L [111], a standard metric in table-to-text generation assessments [115, 221]. As our primary focus lies in assessing the presence of information from the query and the execution table within the summary, we focus on the recall scores of ROUGE-L. As shown in Table 4.2, the ROUGE-L recall scores exceed 90, affirming that the annotated summaries proficiently include not only the facts from the corresponding execution tables but also the contextual information from the queries.

Human evaluation To assess the faithfulness and fluency of annotated summaries, we randomly select 100 examples from each of the training, validation, and test sets. We engage three annotators, each proficient in SQL and English, to evaluate the summaries in relation to the corresponding SQL queries, input tables, and execution tables. Annotators assign a binary label to assess faithfulness, a common method in table-to-text generation tasks [23, 221]. Summaries that accurately represent the execution tables without any hallucinated content are labeled as 1, while those that do not are labeled as 0. The annotators are also provided with the query and input tables to enhance their understanding and judgment of the summaries. Following the methodology described by Zhao et al. [221], we measured fluency using a 5-point Likert scale, ranging from 1 (least fluent) to 5 (most fluent). The average score from the three annotators determined the faithfulness and fluency rating for each summary. Table 4.2 presents the results: the summaries achieved an average faithfulness score of 0.97 and a fluency score of 4.74, indicating that over 97% of the summaries are faithful to the corresponding execution tables and are deemed sufficiently fluent by the annotators. To measure the inter-annotator agreement, we employed the Fleiss Kappa scores [45], achieving Kappa

scores of 0.97 for faithfulness and 0.80 for fluency. These scores indicate almost perfect agreement and substantial agreement, respectively.

Human post-correction Despite our quality verification confirming the high quality of the dataset, we acknowledge the need for further correction of the validation and test sets to ensure their accuracy, as they play a critical role in selecting optimal model checkpoints and measuring model performance, respectively. Additionally, biases may arise from using output summaries produced by LLM_{Anno} to construct these sets. This risk is particularly pronounced if the same LLM_{Anno} is employed as the baseline model, potentially leading to artificially enhanced performance results. To address these concerns, we have implemented a rigorous post-correction process on the annotated summaries within both sets. This involves a detailed manual review to identify and rectify any missing information or hallucinated content in the summaries, based on the corresponding query and execution table. Furthermore, we have undertaken to rephrase each summary in a manner that more closely resembles human expression, thereby reducing potential biases.

4.5 Experiment

In this section, we outline baseline models selected for performance comparison. We then provide the implementation details of the baseline models and our proposed method. Lastly, we describe the evaluation protocols employed in our experiments.

4.5.1 Baseline Models

In this work, we conduct experiments to evaluate two distinct neural network architectures: encoder-decoder models and decoder-only LLMs. Given that encoder-decoder models typically have significantly fewer parameters than LLMs, we fine-tune these models using our training dataset to facilitate a fair comparison. In contrast, we utilize the LLMs as backbone models for our proposed methods without updating their parameters in the experiments. We benchmarked against the following state-of-the-art models:

BART [107] represents a pre-trained encoder-decoder architecture known for its efficacy in text generation tasks. We have fine-tuned two variants of BART, namely `bart-base` with 139 million parameters and `bart-large` with 406 million parameters [107], which are referred to as BART-base-FT and BART-large-FT, respectively.

TAPEX [121] is a table-to-text generation model, trained using a large-scale synthetic dataset that includes executable SQL queries and their corresponding outputs. In our experiments, we employ the version that utilizes `bart-large` as the backbone.

OmniTab [82] is based on the same architecture as TAPEX but has been additionally trained on a synthetic dataset designed for table question answering tasks. Like TAPEX, our implementation leverages `bart-large` as its backbone.

MultiTab [151] is a table-to-text generation model that has been additionally trained on a synthetic, multi-table question answering dataset. In our experiments, we uti-

lize `bart-base` as its backbone, as the released version exclusively supports this backbone.

Llama-2 [192] includes a set of open-source LLMs that have been pre-trained across vast datasets. We explore two adaptations of Llama-2. First, we fine-tune it on our dataset, namely Llama-2-FT. Second, we utilize llama-2 as the backbone for our approach.

GPT [146, 147] comprises a family of LLMs developed by OpenAI, demonstrating their remarkable text generation capabilities across numerous tasks. In our experiments, we deploy versions `gpt-3.5-turbo-0613` and `gpt-4-0613`, applying these models as the backbones for our approach.

4.5.2 Implementation Details

Fine-tuning We fine-tune encoder-decoder models on the training set using the AdamW optimizer [128]. This is conducted over 32 epochs with a learning rate of $1e^{-4}$, batch size of 256, and the maximum sequence length of 1024. For fine-tuning Llama-2 models, we adopt the QLoRA algorithm [35] to fine-tune `llama-2-chat-7B` due to the computation restrictions. The maximum sequence length is 4096. Input sequences for the fine-tuned models were composed by concatenating the query with all linearized input tables. For instance, the final input sequence for an example with k tables is represented as $query\ [table_1] \ldots [table_k]$, where $[table_i]$ is the linearized representation of the *i*-th input table. Model performance is evaluated by selecting the best checkpoints based on the loss from the validation set. All experiments are conducted on a single A6000 GPU.

Few-shot prompting We prepend 3-shot demonstrations into the prompts to facilitate in-context learning [15] for our method. We set the temperature, top-p, and maximum output tokens to 0.1, 0.95 and 400, respectively. Due to budget constraints, we only report the results of the Reason-then-Summ method with the backbone of GPT-4. Details of the prompts can be found in **??**.

4.5.3 QFMTS Evaluation

To assess model performance, we employ automated and human evaluations. For automated evaluation, we measure the quality of the generated summary based on the corresponding reference summary and the execution table, as detailed in Subsection 4.4.1. For human evaluation, we focus on evaluating two key aspects: *faithfulness* and *fluency*.

Text-based automated evaluation We first evaluate the quality of a generated summary w.r.t. the corresponding reference textual summary by estimating the similarity between them in general aspects, such as fluency and accuracy. Following Zhao et al. [221], we adopt two lexical-based metrics, SacreBLEU [154] and ROUGE-L (longest common sub-sequences) [112], along with a semantic-based metric, BERTScore [217]. We report the F1 versions for both ROUGE-L and BERTScore. We use `deberta-xlarge-mnli` [59] as the backbone for BERTScore.

Table-based automated evaluation In contrast to text-based evaluations, table-based evaluations focus more on specific aspects, such as completeness and faithfulness of the generated summary. To assess these aspects, we employ two metrics: Sting Exact Match (STR-EM) [186] and PARENT [36]. STR-EM quantifies the proportion of facts or entities from the execution table that are accurately represented in the generated summary. PARENT evaluates summary completeness by integrating both the reference summary and the execution table. PARENT has demonstrated a significant correlation with human judgments.

Human evaluation In addition to the automated evaluations, we conduct human evaluations, specifically targeting *faithfulness* and *fluency*. These evaluations follow the detailed annotation guidelines specified in Subsection 4.4.3. For each model, 100 generated summaries are randomly sampled from the test set, and their quality was assessed by three expert annotators.

4.6 Results and Analysis

Main results We present the summarization performance of various models in Table 4.3. We observe that our Reason-then-Summ methods markedly outperforms the DirectSumm approach in both text-based and table-based evaluations. For instance, when we use GPT-3.5 as the backbone, Reason-then-Summ surpasses DirectSumm by about 5 points regarding BERTScore (64.98 vs. 60.18). This is because the Reason-then-Summ method tackles the task into sequential phases of table reasoning and summarization. This specialized focus on multi-table reasoning in the initial phase enables the generation of more query-relevant facts and entities, thus enhancing the LLM's reasoning ability. Consequently, the subsequent summarization task benefits from generated more relevant facts, yielding superior performance compared to the DirectSumm approach, which jointly addresses table reasoning and summarization tasks.

Additionally, our results indicate that baseline models fine-tuned on our training dataset largely surpass the DirectSumm methods employing GPT-3.5 or Llama-2 as backbones in text-based evaluations. In comparison, our Reason-then-Summ approach using GPT-3.5 and GPT-4 as backbones demonstrates competitive performance alongside the BART-large-FT and MultiTab-FT. This also indicates the effectiveness of Reason-then-Summ. Furthermore, the Llama-2-FT model achieves the highest performance. Particularly in text-based evaluation, when compared to the Reason-then-Summ with the backbone Llama-2, the fine-tuned Llama-2-FT shows significant improvements, underscoring our dataset's efficacy as a robust training resource for query-focused multi-table summarization scenarios. Conversely, in table-based evaluations, however, the trend reverses. For instance, our Reason-then-Summ method employing GPT-4 as the backbone, largely outperforms the leading Llama-2-FT model. This discrepancy indicates that while smaller, fine-tuned models may produce plausible summaries, they lack proficiency in table reasoning across multiple tables to gather query-relevant facts, thereby leading to inferior performance on table-based metrics. In comparison, our Reason-then-Summ methods produces more facts relevant to the queries, demonstrating

Model	Backbone	Text-based metric			Table-based metric	
		SB	RL	BS	STR-EM	PARENT
Fine-tuned						
BART-base-FT	BART-base	39.74	63.14	62.38	24.63	13.51
BART-large-FT	BART-large	43.40	64.84	66.06	33.54	18.66
TAPEX-FT	BART-large	43.99	65.12	66.43	38.78	22.16
MultiTab-FT*	BART-base	44.41	65.68	67.13	42.70	24.83
OmniTab-FT	BART-large	45.58	67.19	68.76	44.60	26.46
Llama-2-FT	Llama-2-7B	**54.06**	**71.82**	**73.66**	61.71	28.69
Our prompting-based						
DirectSumm	Llama-2-7B	12.49	32.63	21.85	45.32	7.45
DirectSumm	GPT-3.5	33.58	57.02	60.18	53.93	22.21
Reason-then-Summ	Llama-2-7B	16.45	37.13	25.13	48.16	12.17
Reason-then-Summ	GPT-3.5	40.84	62.68	64.98	56.24	24.36
Reason-then-Summ	GPT-4	42.32	64.36	67.36	**66.83**	**32.37**

Table 4.3: Summarization performance of our approaches with various backbones and fine-tuned models on the test set of our dataset. "FT" stands for the fine-tuned version of the corresponding model. "SB" is short for SacreBLEU. "RL" is short for ROUGE-L. "BS" is short for BERTScore. The best results are highlighted in **bold**. * indicates that the released version only supports the backbone of BART-base.

superior performance in terms of table reasoning ability.

Human evaluation Table 4.4 illustrates the results of sampled human evaluation on the test set. Despite lower scores on automated text-based evaluation metrics such as SacreBLEU and ROUGE scores, our Reason-then-Summ method, which integrates GPT-4, significantly outperform the fine-tuned OmniTab-FT in human evaluations, particularly in terms of faithfulness (0.56 vs. 0.19). This disparity highlights the superior reasoning ability of our methods over fine-tuned baseline models. Furthermore, our findings reveal a mismatch between text-based automated metrics and human evaluations, aligning with the observations made by Zhao et al. [221]. In contrast, our table-based metrics demonstrate a strong correlation with human judgments concerning faithfulness. This indicates that table-based evaluations are complementary to text-based evaluations, enabling a more comprehensive evaluation for system performance comparison.

Single- vs. multi-table To enhance our understanding of the challenges presented in multi-table scenarios, we conducted a performance comparison between single-table and multi-table inputs from our test set, as illustrated in Table 4.5. It is worth noting that approximately 30% of the examples in our dataset are characterized by single-table inputs. Our analysis reveals that the presence of multiple input tables significantly deteriorates the performance across all evaluated metrics for every model tested. This decline in performance is particularly significant for the smaller OmniTab-FT, whereas

Model	Backbone	Faithfulness	Fluency
OmniTab-FT	BART-large	0.19	4.79
Reason-then-Summ	GPT-3.5	0.28	4.72
Reason-then-Summ	GPT-4	**0.56**	**4.84**

Table 4.4: Human evaluations of representative models on the test set. Three expert annotators are recruited to evaluate 100 random examples for each model. The best results are in **bold**.

Model	Number of tables					
	Single-Table			Multi-Table		
	R-L	BSc	PA	R-L	BSc	PA
OmniTab-FT (BART-large)	70.73	72.11	37.73	65.67	67.20	21.56
Reason-then-Summ (GPT-3.5)	64.95	67.04	28.92	61.61	64.18	19.29
Reason-then-Summ (GPT-4)	66.03	69.23	38.11	62.58	64.98	29.87

Table 4.5: Comparisons between single-table and multi-table examples on the test set. R-L, BSc, and PA stand for ROUGE-L, BERTScore, and PARENT, respectively.

it is least noticeable for our method utilizing GPT-4 as the backbone. For instance, considering PARENT scores, the decrease observed with OmniTab-FT is approximately 16 points, moving from 37.73 to 21.56. In contrast, our Reason-then-Summ shows a more modest decrease of about 8 points, dropping from 38.11 to 29.87. This reduction is nearly half that observed with OmniTab-FT. These findings suggest that while multi-table reasoning poses greater challenges compared to single-table scenarios, increased model capacity can effectively narrow this performance gap.

Qualitative analysis To enhance our understanding of the strengths of our approach and challenges within the task, we conduct a manual analysis of the summaries generated by Reason-then-Summ with the backbone of GPT-3.5 on the test set, including success and failure cases. We observe that our method successfully performs arithmetic and multi-table operations in some cases. A success case illustrates the strengths of our approach. For the query "*Which employee received the most awards in evaluations? Give me the employee name.*" over two input tables:

Employee

ID	Name	Age
1	**George Chuter**	23
2	Lee Mears	29
...	...	...

Evaluation

ID	Year_awarded	Bonus
1	2011	**3000**
2	2015	3200
1	2016	**2900**
...	...	...

With the reference summary "*The recipient of the most awards in evaluations is George Chuter.*", our method reasons over the 2 tables, performing complex table operations,

such as *count* and *join*. Specifically, the method finds two records of awards of George Chuter in the table `Evaluation` and aggregates the total number of awards. After joining the two tables, the method accurately identifies George Chuter as the person with the most awards, generating "*The employee who received the most awards in evaluations is George Chuter.*"

A failure case illustrates the challenges of multi-table scenarios. Consider the query "*What are the names of all European countries with at least 3 manufacturers?*" over three input tables:

Continents

Cont Id	Continent
1	America
2	**Europe**
3	Asia
4	Africa
5	Australia

Countries

Country Id	Country Name	Cont-inent
2	**Germany**	**2**
3	**France**	**2**
1	USA	1
8	**Korea**	**3**
...	...	...

Car Makers

Id	Maker	Full Name	Country
2	Volkswagen	Volkswagen	**2**
3	bmw	BMW	**2**
...	...	...	...
7	citroen	Citroen	**3**
...	...	...	...
14	opel	Opel	**2**
15	peugeaut	Peugeaut	**3**
16	renault	Renault	**3**
...	...	...	...
22	kia	Kia Motors	**8**

With the reference summary "*There are 2 European countries with at least 3 manufacturers. The names of these countries are France and Germany.*", in which the correct country names are marked in blue. The method incorrectly generates "*There are 2 European countries that have at least 3 manufacturers. Their names are France and Korea.*", in which the incorrect country name is marked in red. Even though this generated summary exhibits a high degree of *fluency*, it is only partially *faithful* and *complete* due to the incorrect inclusion of *Korea*, a country which is not located in Europe. This case exemplifies the complexity and challenges of multi-table operations since the proposed approach struggles to combine information from the three tables based on the corresponding column headers, `Country ID` and `Cont Id`.

Additional examples of summarization with GPT-4 by the Reason-then-Summ approach are shown in Table 4.11. In the first example, for the question *What are the names of high schoolers who have a grade of over 5 and have 2 or more friends?*, GPT-4 erroneously counts 15 high school students instead of the correct answer of 5. The second example with the question *Show names of teachers and the courses they are arranged to teach.* is interesting. GPT-4 correctly summarizes the names of the teachers and the corresponding course taught by the teacher, but counts the total number

Query	Answer Table	Reference	GPT-4-MS
What are the names of high schoolers who have a grade of over 5 and have 2 or more friends?	**name** Alexis Kris Jordan Gabriel Andrew	There are 5 high schoolers who have a grade of over 5 and have 2 or more friends. Their names are Alexis, Kris, Jordan, Gabriel, and Andrew.	There are 15 high schoolers who have a grade of over 5 and have 2 or more friends. These students are Jordan, Gabriel, Tiffany, Cassandra, Andrew, Kris, Brittany, Alexis, Austin, Gabriel, Jessica, Jordan, John,Kyle, and Logan.
Show names of teachers and the courses they are arranged to teach.	**Name** / **Course** Kearsley Brown / Math Vicente Carretero / Math Gustaaf Deloor / Science Anne Walker / History Anne Walker / Bible Lucy Wong / Musi	There are 6 teachers in total. The names of the teachers and the courses they are arranged to teach are as follows: Kearsley Brown teaches Math, Vicente Carretero teaches Math, Gustaaf Deloor teaches Science, Anne Walker teaches History, Anne Walker teaches Bible, and Lucy Wong teaches Music.	There are five teachers in total and each is arranged to teach a specific course. Kearsley Brown and Vicente Carretero are both arranged to teach Math. Gustaaf Deloor is arranged to teach Science. Anne Walker is arranged to teach History and Bible. Lastly, Lucy Wong is arranged to teach Music.
What are the names of poker players in descending order of earnings?	**Name** Maksim Botin Aleksey Ostapenko Teodor Salparov Semen Poltavskiy Yevgeni Sivozhelez	There are 5 poker players in total. The names of the poker players in descending order of earnings are Maksim Botin, Aleksey Ostapenko, Teodor Salparov, Semen Poltavskiy, and Yevgeni Sivozhelez.	There are 5 poker players listed in descending order of earnings. These players are Maksim Botin, Aleksey Ostapenko, Semen Poltavskiy, Teodor Salparov, and Yevgeni Sivozhelez.
For each stadium, how many concerts play there?	**Name** / **count(*)** Stark's Park / 1 Glebe Park / 1 Somerset Park / 2 Recreation Park / 1 Balmoor / 1	There are 5 stadiums in total. The number of concerts played at each stadium are as follows: Stark's Park - 1 concert, Glebe Park - 1 concert, Somerset Park - 2 concerts, Recreation Park - 1 concert, and Balmoor - 1 concert.	The number of concerts played at each stadium varies. Stark's Park hosts 1 concert, Somerset Park hosts 2 concerts, while Bayview Stadium, Hampden Park, Forthbank Stadium, and Gayfield Park **do not host** any concerts. Recreation Park, Balmoor, and Glebe Park each host 1 concert.

Table 4.11: Selected samples with generated summaries by `GPT-4`. Note that we omit input tables since they are too large to display.

of unique teachers instead of the total number of teacher listed. Hence, Anne Walker, teaching two different courses, is counted only once. The reference summary, on the other hand, counts the total number of teachers listed in the answer table. The third example demonstrates GPT-4 correctly generating the summary with respect to the user question *What are the names of poker players in descending order of earnings?*. The fourth example with the question *For each stadium, how many concerts play there?* demonstrates that GPT-4 generates additional information that is not present in the reference summary, such as the list of stadiums that do not have any concerts playing in them.

4.7 Conclusion

In conclusion, this chapter has addressed the shortcomings of current table summarization techniques through the introduction of an innovative method for query-focused multi-table summarization. Our proposed method leverages user queries and analyzes multiple tables to generate summaries that directly cater to users' information needs. Additionally, we make a significant contribution to the field by presenting a comprehensive dataset tailored explicitly for this query-focused multi-table summarization task, thereby enabling further research.

Extensive evaluations conducted demonstrate the superior performance of our method compared to existing baselines, underscoring the complexities associated with accurate summarization in the context of intricate table reasoning. Overall, our work not only propels advancements in query-focused multi-table summarization but also offers valuable insights to guide future exploration and development in this field.

This chapter answers RQ3, *How to generate summaries over multiple tables for conversational agents?*, by introducing the task of query-focused multi-table summarization. To aid the task, a query-focused multi-table summarization (QFMTS) dataset was created and a summarizer controller designed for generation of fluent and faithful summaries over multiple tables and the user question. Multiple experimental results demonstrate the effectiveness of our designed summarizer and dataset.

4.A Prompts

Complete prompts used in the paper are shown in the Tables 4.A.1, 4.A.2 and 4.A.3.

Appendix 4.A.1: Complete prompt for stage 1

Instruction: You will be given a question along with one or more tables to complete the task below. Each table contains a name and content with multiple rows and columns, formatted as follows:
col: <column header 1> | <column header 2> | ... | <column header n> row 1: <value 1,1> | <value 1,2> | ... | <value 1,n> row 2: <value 2,1> | <value 2,2> | ... | <value 2,n> ... row m: <value m,1> | <value m,2> | ... | <value m,n>.

Task: Answering the Question from the Tables.
Your task is to answer the question using only the information from the tables, such as numerical data and entities. This may involve performing arithmetic calculations and combining data from multiple tables if necessary. Please begin your response with 'Answers:' and enumerate all discovered answers one by one, separating them with commas ','. Let's think step by step.

Demonstrations:
Question: Show the name for regions not affected.
Table 1: Name: region; Content: col : Region_id | Region_code | Region_name row 1 : 1 | AF | Afghanistan row 2 : 2 | AL | Albania row 3 : 3 | DZ | Algeria row 4 : 4 | DS | American Samoa row 5 : 5 | AD | Andorra row 6 : 6 | AO | Angola row 7 : 7 | AI | Anguilla row 8 : 8 | AQ | Antarctica row 9 : 9 | AG | Antigua and Barbuda row 10 : 10 | CY | Cyprus row 11 : 11 | CZ | Czech Republic row 12 : 12 | DK | Denmark row 13 : 13 | DJ | Djibouti
Table 2: Name: Affected Region; Content: col : Region_id | Storm_ID | Number_city_affected row 1 : 1 | 1 | 10 row 2 : 2 | 1 | 15 row 3 : 3 | 3 | 30 row 4 : 1 | 4 | 22 row 5 : 12 | 5 | 37 row 6 : 2 | 5 | 12
Answers: American Samoa, Andorra, Angola, Anguilla, Antarctica, Antigua and Barbuda, Cyprus, Czech Republic, and Djibouti are the names for regions not affected.
. . . . . .

Now follow the instructions and the demonstrated style above to complete the task step by step for the question and tables provided below:

Question: *input question here*
Tables: *input tables here*

4.B Limitations

The summaries on our QFMTS were automatically generated by GPT-3.5, despite being scalable and cost-effective, which may limit the diversity of the summaries regarding vocabulary or sentence structure compared to expert annotators. For few-shot prompting baselines, we used fixed few-shot demonstrations, which are easy to implement yet sub-optimal. Advanced demonstration selection methods, such as retrieval-augmented methods [119, 174], have the potential to enhance generation capabilities. Furthermore, these baselines do not explore re-verifying the correctness of the answers before summary generation. Such a verification mechanism may boost the

Appendix 4.A.2: Complete prompt for stage 2

Instruction: You will be given a question along with its one or more answers to complete the task below.
Task: Writing a Summary for the Answers.
Your task is to write a concise, fluent, and accurate summary based on the answers generated in the first task. This summary should begin with the word "Summary:" and follow the guidelines as follows: 1) Introduction: Begin by using a numeral to indicate the total number of answers if there are two or more; Then, rephrase the question as a declarative statement while retaining all relevant keywords. 2) Body: Present all discovered answers one by one. The summary should be a standard paragraph format without using lists, containing a minimum of 5 words but not exceeding 300 words in length.
You can refer to the demonstrations below. Each demonstration consists of a question, tables, and human-written answers, and a summary.

Demonstrations:
Question: Show the name for regions not affected.
Answers: American Samoa, Andorra, Angola, Anguilla, Antarctica, Antigua and Barbuda, Cyprus, Czech Republic, and Djibouti are the names for regions not affected.
Summary: There are 9 regions that are not affected. These regions include American Samoa, Andorra, Angola, Anguilla, Antarctica, Antigua and Barbuda, Cyprus, Czech Republic, and Djibouti.
. . . . . .

Now follow the instructions and the demonstrated style above to complete the task step by step for the question and answers provided below:

Question: *input question here*
Answers: *generated answers here*

faithfulness of the summaries and can be explored in the future.

4.C Ethical Considerations

The source questions and tables in QFMTS are derived from a multi-table QA [151] dataset, which is openly accessible under the MIT license. This facilitates its usage for research purposes. The baseline models used in this paper include closed LLMs accessible via the commercial OpenAI API [2] and publicly available open-source models. In particular, we leverage Copilot primarily to assist with data processing code. We use ChatGPT to mainly correct grammatical errors and ensure the paper does not contain any of the generated text directly from ChatGPT.

[2] https://openai.com/blog/openai-api

Appendix 4.A.3: Complete single-stage prompt

Instruction: You will be given a question along with one or more tables to complete two tasks step by step. Each table contains a name and content with multiple rows and columns, formatted as follows: col: <column header 1> | <column header 2> | ... | <column header n> row 1: <value 1,1> | <value 1,2> | ... | <value 1,n> row 2: <value 2,1> | <value 2,2> | ... | <value 2,n> ... row m: <value m,1> | <value m,2> | ... | <value m,n>.

Task 1: Answering the Question from the Tables.
Your first task is to answer the question using only the information from the tables, such as numerical data and entities. This may involve performing arithmetic calculations and combining data from multiple tables if necessary. Please begin your response with "Answers:" and enumerate all discovered answers one by one, separating them with commas ",". Let's think step by step.

Task 2: Writing a Summary for the Answers.
Your second task is to write a concise, fluent, and accurate summary based on the answers generated in the first task. This summary should begin with the word "Summary:" and follow the guidelines as follows: 1) Introduction: Begin by using a numeral to indicate the total number of answers if there are two or more; Then, rephrase the question as a declarative statement while retaining all relevant keywords. 2) Body: Present all discovered answers one by one. The summary should be a standard paragraph format without using lists, containing a minimum of 5 words but not exceeding 300 words in length.
You can refer to the demonstrations below. Each demonstration consists of a question, tables, and human-written answers, and a summary.

Demonstrations:
Question: Show the name for regions not affected.
Table 1: Name: region; Content: col : Region_id | Region_code | Region_name row 1 : 1 | AF | Afghanistan row 2 : 2 | AL | Albania row 3 : 3 | DZ | Algeria row 4 : 4 | DS | American Samoa row 5 : 5 | AD | Andorra row 6 : 6 | AO | Angola row 7 : 7 | AI | Anguilla row 8 : 8 | AQ | Antarctica row 9 : 9 | AG | Antigua and Barbuda row 10 : 10 | CY | Cyprus row 11 : 11 | CZ | Czech Republic row 12 : 12 | DK | Denmark row 13 : 13 | DJ | Djibouti
Table 2: Name: Affected Region; Content: col : Region_id | Storm_ID | Number_city_affected row 1 : 1 | 1 | 10 row 2 : 2 | 1 | 15 row 3 : 3 | 3 | 30 row 4 : 1 | 4 | 22 row 5 : 12 | 5 | 37 row 6 : 2 | 5 | 12
Answers: American Samoa, Andorra, Angola, Anguilla, Antarctica, Antigua and Barbuda, Cyprus, Czech Republic, and Djibouti are the names for regions not affected.
Summary: There are 9 regions that are not affected. These regions include American Samoa, Andorra, Angola, Anguilla, Antarctica, Antigua and Barbuda, Cyprus, Czech Republic, and Djibouti.
. . . . . .

Now follow the instructions and the demonstrated style above to complete the two tasks step by step for the question and tables provided below:

Question: *input question here*
Tables: *input tables here*

5

Table Question Answering for Low-resourced Languages

Although previous chapters addressed various challenges of table question answering; such as answering user questions over a table or text source (Chapter 2), answering user queries over multiple tables (Chapter 3), and generating fluent and factual summary over multiple tables (Chapter 4), all of them were focused on a high-resource language, English. This chapter studies the challenge of table question answering in the context of resource scarcity. Low-resource setting is challenging as there is a dearth of resources such as datasets and models. To reduce the research gap in a low-resource setting for the task of table QA, this chapter focuses on designing a generalized dataset generation methodology and instantiates it on two Indo-Aryan languages: Bengali and Hindi. Furthermore, as tables introduce novel challenges to question answering models, various neural models were trained on the generated datasets and studied. Experimental results further demonstrated the effectiveness of the models. Additionally, detailed analysis was conducted on various classes of mathematical operations and zero-shot cross-lingual transfer to explore the efficacy of the trained models in the low-resource setting. Specifically, this chapter addresses RQ4 by exploring the adaptation of table QA for low-resourced languages.

5.1 Introduction

Tables are ubiquitous for storing information across domains and data sources such as relational databases, web articles, Wikipedia pages, etc. [34]. Tables introduce new challenges in machine comprehension not present in text as they are are not well-formed sentences but a semi-structured collection of facts (numbers, long-tail named entities, etc.) [74, 79, 83, 91, 121, 141, 149, 227]. Additionally, tables make position (rows/columns) bias [114] and entity popularity bias [56] severe. The table question answering (tableQA) task introduces novel challenges compared to text-based

This chapter was published as V. Pal, E. Kanoulas, A. Yates, and M. de Rijke. Table question answering for low-resourced Indic languages. In Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 75–92, Miami, Florida, USA, Nov. 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.5. URL https://aclanthology.org/2024.emnlp-main.5.

question answering (textQA) [64, 121, 205, 208, 220]. In addition to the semi-structured nature of tables, a tabular context leads to a high frequency of fact-based questions, mathematical and logical operations such as arithmetic [227], set, relational [82, 121], and table operations such as table joins [151]. Effective tableQA systems not only have machine comprehension skills, but also numeracy understanding [24, 121, 220, 227], table reasoning [121, 208], table summarization [218, 221] and answer table generation ability [151].

Low-resource tableQA aims to answer questions over semi-structured tables storing cultural and region-specific facts in a low-resource language. Joshi et al. [84] show that most languages struggle to be represented and are deprived of advances in NLP research. As manual data collection is slow and expensive, low-resource languages struggle with large-scale, annotated data for effective transfer learning solutions. The low-resource setting [61, 176] exacerbates the challenges of tableQA with challenges of data sparsity, annotated data costs, and lack of trained models. In contrast to textQA, syntactico-semantic variations such as agreement and morphology are not exhibited in tables, but the high presence of culturally significant yet long-tail entities makes adapting existing high resource datasets and trained models challenging. Research on low-resource table inference [137] shows that standard approaches of translating English datasets for low-resource data creation are infeasible for tables due to high translation error as tables are not well-formed sentences.

Challenges This chapter focuses on studying the following core challenges of low-resource tableQA:

(1) Low-resource **tableQA data scarcity** and under-representation of cultural facts.

(2) Existing **neural models' poor alignment** in low-resource languages and lack of understanding of table structure.

This motivates us to explore low-resource tableQA by designing a low-cost and large-scale automatic data generation and quality estimation pipeline. We discuss the process in detail with a low-resource Indic language, Bengali (spoken extensively in Bangladesh and India, with over 230 million native speakers [88]), and explore generalizability with Hindi (570 million speakers).

Contributions Our main contributions in this chapter are as follows:

(1) We introduce the low-resource tableQA task.

(2) We design a method for automatically generating low-resource tableQA data in a scalable budget-constrained manner.

(3) We release resources to support low-resource tableQA: Large-scale tableQA datasets and models for 2 Indic languages, Bengali (Bengali table question answering (BanglaTabQA)) and Hindi (Hindi table question answering (HindiTabQA)). BanglaTabQA contains $19K$ Wikipedia tables, $2M$ training, $2K$ validation, and 165 test samples. HindiTabQA contains $2K$ Wikipedia tables, $643K$ training, 645 validation, and 125 test samples.

5.2 Related Work

TableQA aims to answer a user question from semi-structured input tables. Prior work on tableQA in English can be classified as extractive [64, 207] or abstractive [141, 149, 205, 221]. While extractive tableQA focuses on row and cell selection [64], abstractive tableQA generates various types of answers, such as factoid answers [121], summaries [218, 221], or answer tables [151]. The low-resource setting poses challenges for various NLP tasks. The low-resource corpus creation [13, 33, 58] has used automatic annotation efforts by synthesizing a large-scale dataset. Das and Saha [33] train a Bengali QA system by developing a synthetic dataset translated from standard English QA datasets. Bhattacharjee et al. [13], Hasan et al. [58] create low-resource datasets by translating English datasets to Bengali using neural models. However, these methods are unsuitable due to the semi-structured ungrammatical sequential representation of tables.

5.3 Task Definition

We formulate low-resource tableQA as a sequence generation task. Given a question Q of k tokens $q_1, q_2, \ldots, q_k$, and table T comprising of m rows and n columns $\{h_1, \ldots, h_n, t_{1,1}, t_{1,2}, \ldots, t_{1,n}, \ldots, t_{m,1}, t_{m,2}, \ldots, t_{m,n}\}$ where $t_{i,j}$ is value of the cell at the i^{th} row and j^{th} column and h_j is the j^{th} column header; the low-resource tableQA model generates an answer table T_{out}. The input sequence is the concatenated question Q and linearized input table T separated by special sentinel tokens. The answer, T_{out}, is also a linearized sequence. Henceforth, for concreteness, we will use Bengali as the example low-resource language. The input to such a model is:

$q_1\ q_2 \ldots q_k$ <কলাম> $h_i \ldots h_n$ <রো ১> $t_{1,1} \ldots t_{1,n}$ <রো i> $t_{i,j} \ldots t_{i,n} \ldots$ <রো m> $t_{m,1} \ldots t_{m,n}$.

The answer table, T_{out}, is a linearized sequence:

<কলাম> $H_i \ldots H_q$ <রো ১> $o_{1,1} \ldots o_{1,q}$ <রো i> $o_{i,j} \ldots o_{i,q} \ldots$ <রো m> $o_{p,1} \ldots o_{p,q}$

where $o_{i,j}$ is value in the i^{th} row and j^{th} column and H_j is the j^{th} column header of T_{out}.

5.4 Methodology for Dataset Generation

Effective training of low-resource tableQA requires creation of large-scale datasets of questions, input and answers tables, to align a language model to the low-resource language and adapt it to semi-structured tables and QA task. We address **Challenge 1** by designing an automatic data generation process to generate a large-scale low resource tableQA corpus of training and validation samples. We follow a three-step pipeline as follows: (i) table extraction, (ii) question generation, and (iii) answer table extraction. This pipeline applied on Bengali, as depicted in Figure 5.1, generates the **BanglaTabQA** dataset.

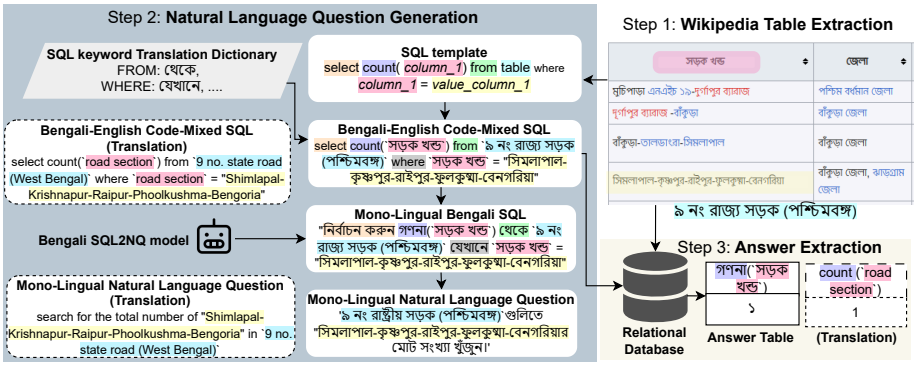


Figure 5.1: **BanglaTabQA Dataset generation**: The SQL elements and table elements are color-coordinated to represent a single SQL/table element. Dotted rectangles represent translations for accessibility to non-native readers.

5.4.1 Table Extraction

English Wikipedia with $6,751,000+$ articles is used for English tableQA datasets [157], but is insufficient for non-Latin languages with many cultural topics missing. The standard process [13, 33] of translating English datasets to low-resource languages is biased due to lack of cultural topic/fact representation in English tableQA datasets. For example, the named-entity অধরিাজ গাঙ্গুলি (Adhiraj Ganguly), exists only in Bengali Wikipedia,[1] and not in English. Further, translating English tables with machine translation models is error-prone [137] as tables are not well-formed sentences, but collections of facts. To mitigate these issues, we extract tables from the Wikipedia dump of the low-resource language.

5.4.2 Natural Language Question Generation

The question generation is a two-step process:

Code-mixed SQL query generation We automatically generate SQL queries over the extracted low-resourced tables with SQL templates from the SQUALL dataset [184]. These templates have placeholders of table components such as table name, column names, etc. which are randomly assigned with values from a Wikipedia table. For example, the template "`select count(c1) from w where c1 = value`" is instantiated by assigning a Bengali table name "৯ নং রাজ্য সড়ক (পশ্চিম বঙ্গ)" to `w`, column header "জেলা" to `c1`, and "বাঁকুড়া জেলা" to `value`. This results in an executable code-mixed query "`select count(`জেলা`) from` ৯ নং রাজ্য সড়ক (পশ্চিম বঙ্গ) `where 'জেলা' = "`বাঁকুড়া জেলা`"`", where the SQL keywords are in English but all table information is in the low-resource language (Bengali). This leads to $13,345,000$ executable Bengali code-mixed queries.

[1] https://bn.wikipedia.org/wiki/অধরিাজ_গাঙ্গুলি

Natural language question generation We formulate question generation as a sequence-to-sequence task by transforming a code-mixed SQL query into a natural language question (NQ). To the best of our knowledge, there exist no sequence generation models that translate code-mixed SQL queries to low-resource natural language questions. To train a model for this conversion, we first transform the code-mixed SQL to a monolingual SQL-like query in the low-resource language. As the only linguistic variation exhibited in the SQL templates is polysemy i.e. a dearth of one-to-one correspondence between English SQL keywords and the corresponding low-resource language translations, we employ native speakers well versed in SQL to manually create one-to-one mappings of 27 SQL keywords for linguistic transfer of SQL keywords to the corresponding low-resource language. All table-specific words are directly copied into the monolingual query. We discard `FROM` keyword and table name from the query as it is associated with a single input table. This leads to a SQL-like monolingual query in the low-resource language, which is a well-formed sentence. For example, code-mixed Bengali query "`select count ( 'জেলা' ) from ৯ নং রাজ্য সড়ক (পশ্চিম বঙ্গ) where 'জেলা' = "বাঁকুড়া জেলা"`", results in a monolingual Bengali query "নির্বাচন করুন গণনা('জেলা') যেখানে 'জেলা' = "বাঁকুড়া জেলা"". In contrast to tables which are invalid sentences, queries and NQ are well-formed sequences and effectively transformed (SQL to question) with existing encoder-decoder models. We train a SQL-to-NQ (SQL2NQ) model (`mbart-50-large` [125] backbone) by translating $68,512$ training and $9,996$ validation samples from semantic parsing datasets: Spider [208], WikiSQL [224], Atis [30, 165], and Geoquery [212] to the low-resource language. We use this SQL2NQ model to transform the queries to NQ. For example, Bengali SQL2NQ model transforms the aforementioned query to the NQ "কবার বাঁকুড়া জেলার উল্লেক আছে?".

We report the validation scores of the SQL2NQ models in Table 5.1. The Bengali SQL2NQ model scores are lower than the Hindi SQL2NQ model. Manual inspection of the generated dataset reveals that the Hindi questions and query have higher lexical overlap compared to the Bengali questions-query pairs where the questions are more natural leading to lower lexical overlap with the corresponding SQL query.

	Bengali	**Hindi**
Rouge-1	14.63	53.20
Rouge-2	5.83	24.98
Rouge-L	14.28	51.58

Table 5.1: Bengali SQL2NQ model's validation scores (%).

5.4.3 Answer Table Extraction

We dump low-resource Wikipedia tables in a relation database. The code-mixed SQL queries are executed with an SQL compiler over a relational database comprising the low-resourced Wikipedia tables to extract the answer tables. We execute the $13,345,000$ Bengali code-mixed queries to extract the corresponding answer tables.

5.4.4 Automatic Quality Control

We employ automatic quality control steps to ensure quality of the synthetic tableQA data.

Code-mixed query and answer quality control We discard all code-mixed queries that execute to an error with an SQL compiler. This process follows the quality control in [151] and discards invalid and erroneous queries and samples.

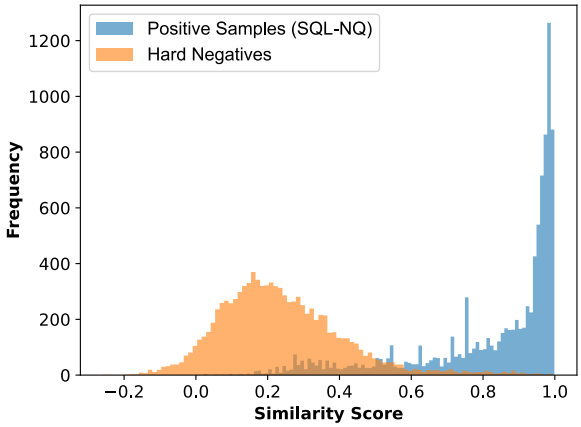


Figure 5.2: Histogram of similarity scores from fine-tuned Bengali SQL2NQSim model of $1,000$ random samples.

Natural language question quality control We evaluate the quality of the generated NQ with a sentence similarity model to discard questions that have a low similarity score with the corresponding monolingual queries. We found the standard method of quality evaluation in low-resource languages [13, 169] using the sentence similarity model, LaBse [44], incompatible for code-mixed SQL-NQ due to low discriminating ability (0.55 mean similarity score and 0.13 standard deviation for Bengali SQL-NQ). For example, LaBse assigns low score (0.43) for positive SQL-NQ pair corresponding to the Bengali query "`SELECT title ORDER BY year DESC LIMIT 1`" and Bengali NQ "`Return the most recent title corresponding to the most recent year`" (translated for non-native readers), while it assigns a high score (0.8) to negative pair "`SELECT count(*) WHERE 'work' = The World of Saudamini`" and the unrelated NQ "`How many games scored a total of 4?`". Table 5.2 shows more examples. This necessitates fine-tuning LaBse on low-resourced SQL-NQ samples. First, we use the translated semantic parsing samples ($68,512$ training and $9,996$ SQL-NQ pairs), described in Section 5.4.2, as positive pairs and in-batch negatives with multiple-negatives ranking loss. We call this the SQL2NQSim model. We select the best checkpoint by evaluating SQL2NQSim on $1,000$ randomly selected hard-negatives (unrelated/negative SQL-negative question

pairs for which pre-trained LaBse assigns a high similarity score (> 0.5)). We use that checkpoint to obtain similarity scores of the low-resourced tableQA SQL-NQ pairs and discard samples with a similarity score lower than a threshold. We select a good threshold by plotting a histogram of scores assigned by the SQL2NQSim model on $10,000$ randomly selected positives and hard-negatives and selecting the inflection point as the threshold. Figure 5.2 shows the scores' histogram for BanglaTabQA. We select a strict threshold of 0.74 (hard-negatives scores taper-off around 0.7). The final BanglaTabQA dataset, after quality control, comprises of $2,050,296$ training and $2,053$ validation samples.

Comparison of Scores of LaBSE and SQL2NQSim Models We qualitatively compare the sentence similarity models LaBse and SQL2NQSim with examples shown in Table 5.2. We observe that LaBse scores are low for positive samples of Bengali SQL queries and the corresponding Bengali question. Further, negative samples, i.e., Bengali SQL query and an unrelated Bengali question have high similarity scores. This trend is not observed for the sentence similarity model, SQL2NQSim, trained on Bengali SQL queries and the corresponding Bengali natural questions.

	Bengali SQL	**Bengali Question**	**LaBse Scores**	**SQL2NQ-Sim Scores**
+ve	নির্বাচন করুন 'বছর' দল করা 'বছর' সাজান হোক গণণা('ফলাফল') সীমা ১ (SELECT years GROUP BY years ORDER BY COUNT(result) LIMIT 1)	কোন বছরে সবচেয়ে কম ফল হয়েছে? (Which year has the least number of results?)	0.45	0.94
	নির্বাচন করুন 'শিরনাম' সাজান হোক 'বছর' অবরোহী সীমা ১ (SELECT 'title' ORDER BY 'year' DESC LIMIT 1)	সম্প্রতিকতম বছরের সাথে সম্প্রতিক শিরনাম ফেরত দিন। (Return the most recent title of the most recent year?)	0.43	0.98
-ve	নির্বাচন করুন সর্বনিম্ন('সাল') (SELECT min('year'))	কোন বছরে (২০১০, ২০১৬) সবচেয়ে বেতশ শংখ্যক পুরষ্কার জিতেছে? (In which year (2010, 2016) were the most number of awards received?)	0.51	0.31
	নির্বাচন করুন গনণা(*) যেখানে 'কাজ'=সৌদামিনীর সংসার (SELECT count(*) WHERE 'work'="The World of Saudamini")	মোট ৪ আছে এমন গেমের মোট শংখ্যা গনণা করুন। (How many games scored a total of 4?)	0.80	0.07

Table 5.2: Comparison of sentence similarity scores between LaBse and our trained SQL2NQSim models.

5.4.5 Dataset Analysis

In contrast to textQA, tableQA focuses on mathematical questions [121, 151, 227]. Following [121], we analyze BanglaTabQA dataset on question complexity, which estimates the difficulty of a question based on the corresponding SQL query. As tableQA enforces mathematical, logical and table reasoning questions, we further classify tableQA queries into different classes of table operations determined by the SQL operators present.

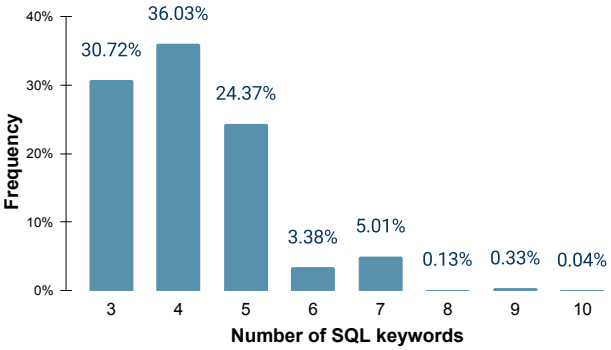


Figure 5.3: Number of SQL keywords per query histogram in the BanglaTabQA dataset.

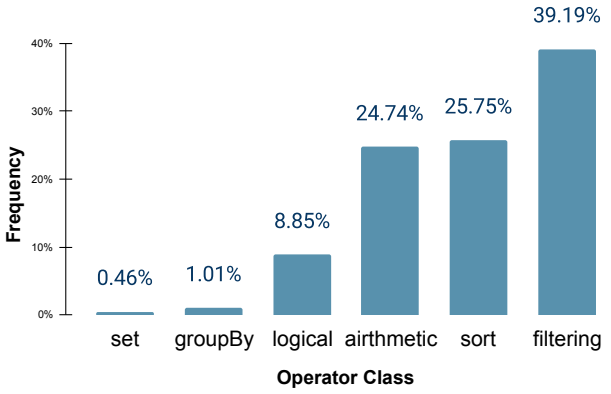


Figure 5.4: Histogram of operator classes in the BanglaTabQA dataset.

Question complexity Recent work on tableQA [121] categorizes SQL queries into difficulty levels based on the number of SQL keywords. We follow this approach and count the number of keywords for each query. Figure 5.3 shows that most of BanglaTabQA queries have 4 SQL keywords. The longest SQL queries are comprised of 10 keywords, and the shortest ones of 3 SQL keywords.

Mathematical operations We further categorize each sample based on the operators present in the question. We utilize the SQL query associated with a question to extract

all keywords for classification. We categorize data samples into 6 operator classes: arithmetic, sorting, group by, filtering, set operators, and logical operators. Arithmetic operators comprises of SQL numeric operations such as `sum`, `count`, `min`, etc. Sorting refers to ordering of the answer values in an ascending or descending order. Group by is the SQL operator of grouping rows based on a criterion. Filtering corresponds to SQL operators such as `where` and `having` used to filter the input table. Set operators involve `union`, `intersect`, and `except`. Finally, we classify logical operators to be conjunction (`and`) and disjunction (`or`) to combine filtering conditions. It also includes membership operators (`in`, `between`, etc.) and string matching operator (`like`). The classification of the operators is shown in Table 5.5. Figure 5.4 shows the distribution of the 6 operator classes for the BanglaTabQA dataset.

5.4.6 Test Set

We manually annotate test samples for evaluating low-resource tableQA models on clean data. We select unique tables not present in the training and validation set to avoid data leakage. To ensure question diversity, we select code-mixed SQL representing each of the 6 operator classes (discussed in Section 5.4.5) and distinct from the training and validation data. Three native annotators well-versed in SQL were employed for annotation. One annotator was tasked with question generation, given the synthetic SQL query, input tables, and the answer table, and asked to rewrite the code-mixed query to a natural language question. The remaining two were tasked with evaluating the question generated by the first annotator. The evaluator-annotators were provided the code-mixed query, input table, answer table, and the annotated question and asked to rate the question based on fluency. We estimate the fluency of the annotated question with a 5-point Likert scale (1-5), where a higher score indicates better fluency. The final score for each question was computed by averaging the scores of the evaluator-annotators. For BanglaTabQA, we manually annotate 165 test samples. We estimate inter-annotator agreement with Fliess's Kappa score [45] of 0.82, indicating strong agreement among the annotators. The average fluency score on the test set questions was 4.3, indicating high fluency.

5.4.7 Generalizability of Dataset Methodology

We study the generalizability of the dataset generation method by repeating the process on another Indic language: Hindi (Hi) with more than 602 million speakers. To the best of our knowledge, there are no existing tableQA data for Indic languages. Hindi text is in Devanagari script, which is different from Bengali written in Eastern-Nagari (Bengali-Assamese) script. This requires tableQA models to be trained on large-scale Hindi datasets for good alignment. Following the dataset creation process in Section 5.4, we extract $1,921$ Hindi tables from the respective Wikipedia dumps. We generate $82,00,000$ Hindi code-mixed queries automatically to extract answer tables and generate the Hindi natural language questions. The final HindiTabQA dataset comprises $643,434$ synthetic training, 645 synthetic validation samples, and 121 manually annotated test samples.

5.5 Experimental Setup

We address **Challenge 2** by studying the effectiveness of state-of-the-art models (baselines) in *Bengali table QA*. The experimental results (Section 5.6) show the need for a large-scale BanglaTabQA dataset and model training. We analyze several models' effectiveness in Bengali language, mathematical/table operations, and generalizability, thus providing a measure of the dataset quality and consequently the dataset creation methodology.

Baselines We perform 2-shot in-context learning (ICL) to adapt large language model (LLM)s to BanglaTabQA task. We further fine-tune an encoder-decoder model. The demonstrations are the concatenated question and flattened input table with the flattened answer table. We use the following models as baselines:

(1) **En2Bn:** We fine-tune an encoder-decoder model, `mbart-50-large`, with $25,000$ random samples from MultiTabQA's [151] pre-training data translated to Bengali using Google translate. MultiTabQA used SQUALL templates to generate their queries and have the same distribution as BanglaTabQA queries. However, the input tables of MultiTabQA are English wiki-tables from WikiTableQuestions dataset [157] and are not representative of Bengali cultural topics/facts.

(2) **OdiaG [156]** is Llama-7b [192] adapter-tuned (LoRA [69]) on 252k Bengali instruction set.[2]

(3) **GPT:** `GPT-3.5` [15] performs well on English tableQA [213]. `GPT-4` [146] outperforms other LLMs (Chinchilla [66], PaLM [26]) in low-resource languages, including Bengali and Hindi, on various tasks ($14,000$ multiple-choice problems on 57 subjects in a translated MMLU benchmark [63]).

BanglaTabQA models Bengali tableQA models must understand both Bengali **script** *and* **numerals**, crucial for mathematical operations. However, Bengali numbers are not present in many state-of-the-art Indic models' [29, 49][3] vocabulary. To the best of our knowledge, there is no open-access generative model which understands both the table structure and Bengali. We train the following models on BanglaTabQA as they support Bengali and Hindi numbers and text:

(1) **BnTQA-mBart:** `mbart-50-large` [125] is a multi-lingual encoder-decoder model with support for 50 languages.

(2) **BnTQA-M2M:** `m2m100_418M` [43] is a multi-lingual encoder-decoder model with support for 100 languages.

(3) **BnTQA-llama:** We train `Llama-7B`, on BanglaTabQA dataset with parameter-efficient fine-tuning (PEFT) on LoRA adapters.

[2] OdiaGenAI/odiagenAI-bengali-lora-model-v1
[3] ai4bharat/IndicBART

Model	Validation Set scores (%)				Test Set scores (%)			
	Bengali							
	Tab	**Row**	**Col**	**Cell**	**Tab**	**Row**	**Col**	**Cell**
En2Bn	0.05	3.06	0.20	3.07	0.00	4.73	0.00	4.73
OdiaG	0.00	3.89	0.00	3.89	0.69	1.77	0.69	1.42
GPT-3.5	1.14	4.81	1.67	5.14	6.04	10.06	9.12	9.84
GPT-4	0.00	13.57	5.43	14.65	26.83	**38.67**	26.74	**36.51**
				BnTQA				
-llama	**60.08**	**68.30**	**60.47**	**68.30**	9.41	12.35	9.85	11.87
-mBart	56.63	64.10	56.79	64.31	**35.88**	33.16	**35.88**	33.16
-M2M	45.31	58.07	45.29	58.04	28.05	34.55	28.05	34.55
Model	**Validation Set scores (%)**				**Test Set scores (%)**			
	Hindi							
	Tab	**Row**	**Col**	**Cell**	**Tab**	**Row**	**Col**	**Cell**
En2Hi	0.00	3.37	0.47	3.43	0.00	5.03	8.26	5.03
OpHathi	0.00	0.00	0.00	0.00	0.00	0.11	0.37	0.74
GPT-3.5	4.81	8.94	4.99	9.71	8.20	10.29	7.10	9.81
GPT-4	15.53	22.60	16.02	22.25	11.11	21.49	11.76	20.84
				HiTQA				
-llama	14.76	9.92	14.13	7.29	13.11	9.71	11.11	7.66
-mBart	92.09	87.97	92.02	87.97	33.06	43.35	33.88	43.35
-M2M	89.55	85.32	89.34	85.15	28.93	33.11	28.92	33.10
-BnTQA	92.40	88.10	92.42	88.12	**41.32**	**47.26**	**41.32**	**47.26**

Table 5.3: Baseline, BnTQA-X and HiTQA-X models' scores. -X represents the backbone architecture of a fine-tuned model and $-$ entries are for incompatible models in a low-resourced language (Bengali or Hindi).

We train `BnTQA-mBart` and `BnTQA-M2M` with 128 batch size and `BnTQA-llama` with 16 batch size, and 4-bit quantization. All models are trained with $1e-4$ learning rate on a single A6000 48GB GPU for 5 epochs with 1024 maximum sequence length.

5.5.1 HindiTabQA

We assess the generalizabiltiy of our data generation process by training and evaluating HindiTabQA models. All hyper-parameters and experimental setup are the same as Bengali.

Baselines We use the following baselines:

(1) **En2Hi:** Similar to `En2Bn`, we fine-tune `mbart-50-large` with $25,000$ random samples from MultiTabQA, translated to Hindi.

(2) **GPT**: We perform 2-shot ICL on the best LLMs on Bengali, `GPT-3.5` and `GPT-4`.

(3) **OpHathi**: We perform 2-shot ICL on `OpenHathi-7B-Hi-v0.1-Base`, an open-source LLM based on `llama-7b` and trained on Hindi, English, and Hinglish text.

HindiTabQA models We train the following models on the HindiTabQA dataset:

(1) **HiTQA-llama:** Similar to Bengali, we fine-tune `Llama-7b` on HindiTabQA dataset.

(2) **HiTQA-M2M:** Similar to Bengali, we fine-tune `m2m100_418M` on HindiTabQA dataset.

(3) **HiTQA-mBart:** Similar to Bengali, we fine-tune `mbart-50-large`, on HindiTabQA.

(4) **HiTQA-BnTQA:** `BnTQA-mBart`, trained on BanglaTabQA provides a warm start. We fine-tune it on HindiTabQA for better convergence.

5.5.2 Evaluation Metrics

The answer table requires both the table structure and content evaluation, rendering standard text similarity metrics (Rouge, BLEU, etc.) inappropriate. We instead evaluate with tableQA evaluation metrics [151]. Henceforth, F1 scores are the harmonic mean of the precision and recall scores.

(1) **Table Exact Match Accuracy (Tab)** measures the percentage of generated answer that *match exactly* to the target answer tables.

(2) **Row Exact Match F1 (Row)**: Row EM precision is the percentage of correctly predicted rows among all predicted rows. Row EM recall is the percentage of correctly predicted rows among all target rows.

(3) **Column Exact Match F1 (Col)**: Column EM precision is the percentage of correctly predicted columns and corresponding headers among all predicted columns. Column EM recall is the percentage of correctly predicted columns among all target columns.

(4) **Cell Exact Match F1 (Cell)** is the most relaxed metric. Cell EM precision is the percentage of correctly generated cells among all predicted cells. Cell EM recall is the percentage of correctly predicted cells among all target cells.

5.6 Results

Baselines As reported in Table 5.3, `GPT-4` performs the best on our test set with a table EM accuracy of 26.83%. `GPT-3.5` under-performs `GPT-4` but is better than open-sourced LLMs. Open-source LLMs, `OdiaG` is pre-trained on Bengali text data but not on structured table data. The low accuracy of `OdiaG` (0.69%) can be attributed to the models' lack of table understanding and table specific question which differs

Model	No post-processing				With post-processing			
BnTQA	**Tab**	**Row**	**Col**	**Cell**	**Tab**	**Row**	**Col**	**Cell**
-llama	0.00	0.00	0.00	0.26	5.74	17.59	5.69	15.49
-mBart	0.00	8.70	10.74	8.70	19.01	20.74	19.01	20.74
-M2M	0.00	0.00	0.00	0.00	18.18	35.80	18.18	35.80

Table 5.4: Zero-shot cross-lingual transfer scores of BnTQA models on Hindi test data.

significantly from text-based tasks on which it has been pre-trained on as shown in examples in Section 5.7. Baseline encoder-decoder model, `En2Bn`, fine-tuned on translated tableQA data, correctly generates 4.73% of rows and cells and under-performs `OdiaG`, but is better than `TableLlama`. Although fine-tuning improves Bengali understanding, the low scores can be attributed to the erroneous translations of English tables in the MultiTabQA dataset which corroborate with [137] that table translation leads to error-propagation to down-stream QA task. Further, a lack of culture-specific tables in the MultiTabQA pre-training dataset leads to downgraded performance on topics in the BanglaTabQA test set. In conclusion, `GPT-4` is able to perform table reasoning in low-resourced Bengali, but is very expensive and closed-source, limiting it's accessibility and utility. `GPT-3.5`'s and all open-access baseline models' low scores demonstrates the need for both task and language adaptation with a large-scale dataset for training accessible open-source language models for low-resourced tableQA.

BanglaTabQA models Parameter-efficient fine-tuned Llama models, `BnTQA-llama`, achieves comparable results to `GPT-3.5`. Table 5.3 shows that fine-tuned encode-decoder models, `BnTQA-mBart` and `BnTQA-M2M`, outperforms `GPT-4` on table exact match accuracy (EM) and column EM F1, but not for row and cell EM F1. This can be attributed to incorrect header generation of `GPT-4` reflecting in column and subsequently table EM scores. Apart from `GPT-4`, all other baseline models underperform BanglaTabQA encoder-decoder models by a large margin on all metrics. `BnTQA-llama` overfits to the validation set and does not generalize well to the test set. The low scores of PEFT compared to full fine-tuning (FT) can be attributed to insufficient alignment of the frozen parameters of the backbone Llama model and sub-optimal tokenization of Bengali which has been observed in SentencePiece tokenizers in non-Latin languages [5, 28]. The results establish the quality of the BanglaTabQA dataset and its effectiveness in adapting neural models to *both* language and table understanding.

HindiTabQA models We follow an experimental setup similar to the one discussed in Section 5.5. We report the results in Table 5.3. We observe that `HiTQA-BnTQA`, initialized with with `BnTQA-mbart`, outperforms all HindiTabQA models and achieves a test score of 41.32%. Similar to BanglaTabQA, `HiTQA-mBart` outperforms `HiTQA-M2M` with a table EM test score of 33.06% and 28.93% respectively. `HiTQA-llama` underperforms compared to the encoder-decoder models. All models trained on the HindiTabQA dataset outperform the two-shot in-context learning baseline models. The

Operator class	Operations
arithmetic (A)	count, sum, average, max, min
sorting (So)	ascending, descending
groupBy (G)	table column/row grouping
filtering (F)	where, having
set (Se)	union, intersect, except
logical (L)	and, or, not, in, not in, between

Table 5.5: Classification of tableQA operations.

Operator class	Bengali				Hindi			
	Tab	**Row**	**Col**	**Cell**	**Tab**	**Row**	**Col**	**Cell**
arithmetic (A)	39.66	55.64	39.67	55.64	35.06	41.71	35.07	41.71
sorting (So)	25.00	25.00	25.00	25.00	**39.05**	**42.74**	**39.05**	**42.74**
groupBy (G)	**50.00**	**76.92**	**50.00**	**76.92**	33.33	35.96	33.33	35.96
filtering (F)	37.78	35.86	37.77	35.86	23.23	26.35	23.23	21.67
set (Se)	36.11	49.10	36.11	49.10	5.00	11.11	5.00	11.11
logical (L)	34.38	13.23	34.38	13.23	25.58	27.38	25.58	27.38

Table 5.6: XTQA-mBart test set scores (%) on Operator Class (Op); X is a low-resourced language (Bn or Hi).

results follow a similar trend to BanglaTabQA models and prove that our data generation process is generalizable and the HindiTabQA dataset is able to align neural models for the tableQA task in Hindi.

5.6.1 Zero-shot Cross-lingual Transfer

We further study generalizability, by selecting the best performing language, Bengali, and evaluating the BanglaTabQA models on Hindi test set in a zero-shot setting *without* training on Hindi data. This setup allows us to study the cross-lingual transfer of BanglaTabQA models to Hindi with a different script, and evaluate how well the models generalize to new out-of-distribution input tables. BanglaTabQA models are able to perform table reasoning in Hindi indicating semantic information transfer across languages. We demonstrate some examples in the Appendix 5.8. Table headers and numbers generated from math operations are often in Bengali instead of Hindi (Example 7). The extractive questions are generated correctly (Example 8). Table 5.4 lists the zero-shot cross-lingual scores using the original predictions (named "No Post-Processing") of the BanglaTabQA models on the Hindi test set defined in Section 5.4.7. Additionally, we perform post-processing of the predictions to translate the predicted tables' values to Hindi. As translating tables, composed of numbers and entities, with machine translation systems is unreliable [137], we follow an automatic post-processing pipeline to transform the predicted answer tables to Hindi. First, all lexical occurrence of Bengali digits in predictions are replaced with Hindi digits using a dictionary. Next, all lexical occurrences of SQL keyword in Bengali in the prediction headers are replaced with a Bengali-to-SQL keyword mapping and subsequently with a SQL-to-Hindi mapping

described in Section 5.4. This fixes most of the Bengali presence in the predictions. Finally, we translate the predicted column names/values in Bengali to Hindi with Google translate. Table 5.4 shows that post-processing increases the scores, demonstrating the generalizability of BanglaTabQA models' table reasoning capabilities on out-of-domain Hindi tables with unseen cultural entities. This further demonstrates the quality and utility of the BanglaTabQA dataset and our proposed data generation method and quality of the trained models.

5.6.2 Mathematical Operator Classes

We study how BanglaTabQA and HindiTabQA datasets aid in Bengali and Hindi numeracy and math understanding by evaluating `BnTQA-mBart` and `HiTQA-mBart` on 6 categories of operator classes (Section 5.4.5). We observe in Table 5.6 that `BnTQA-mbart` performs best on *groupBy (G)* operators with a table EM accuracy of 50.00% and `HiTQA-mBart` on *Sorting (So)* operators with a table EM accuracy of 39.05%. Both models are able to generalize to unseen tables in the respective languages' test sets. This affirms that BanglaTabQA and HindiTabQA dataset aids mathematics reasoning of the trained models and enhances numeracy understanding in the low-resourced language.

5.7 BnTabQA Models Qualitative Analysis

We analyze the output of each model with an example to identify error patterns and factors that impact model predictions. The test set question কার নামে ফুটসাল সমন্বয়কারী অথবা প্রযুক্তিগত পরিচালকের অবস্থান আছে? (Who has the position of Futsal Coordinator or Technical Director?), involves logical operator `or` after extracting values for ফুটসাল সমন্বয়কারী (Fulsal Coordinator) and প্রযুক্তিগত পরিচালকের (Technical Director) from the column অবস্থান (`Position`). The input table is shown in Table 5.8 (translation of each table cell is italicized and in parentheses for non-native readers) with target (English translation italicized and in parentheses) shown in Table 5.7:

নাম (*Name)*)
মাইকেল সকুবালা (*Michael Skubala)*
লেস রিড (*Les Reed)*

Table 5.7: Example: BnTabQA Target Table. (English translation of each cell is italicized and in parenthesis)

Example 1 Baseline encoder-decoder model, `En2Bn`, fine-tuned on the translated MultiTabQA dataset, correctly extracts মাইকেল স্কুবালা (*Michael Skubala)* as the ফুটসাল সমন্বয়কারী (Fulsal Coordinator), but wrongly assigns it as the table header instead of নাম (name). Moreover, it generates the same entity twice instead of generating লেস রিড (Les Reed):

অবস্থান *(Position)*	নাম *(Name)*
সভাপতি *(Chairman)*	গ্রেগ ক্লার্ক *(Greg Clark)*
সহ-সভাপতি *(Co-Chairman)*	ডেভিড গিল *(David Gil)*
সাধারণ সম্পাদক *(General Secretary)*	মার্ক বুলিংহ্যাম *(Mark Bullingham)*
কোষাধ্য *(Treasurer)*	মার্ক বারোস *(Mark Burroughs)*
গণমাধ্যম এবং যোগযোগ পরিচালক (Media And Communications Director)	লুইসা ফিয়ান্স *(Louisa Fiennes)*
প্রযুক্তিগত পরিচালক *(Technical Director)*	লেস রিড *(Les Reed)*
ফুটসাল সমন্বয়কারী *(Futsal Coordinator)*	মাইকেল স্কুবালা *(Michael Skubala)*
জাতীয় দলের কোচ (পুরুষ) *(National Team Coach (Male))*	গ্যারেথ সাউথগেট *(Gareth Southgate)*
জাতীয় দলের কোচ (নারী) *(National Team Coach (Female))*	ফিল নেভিল *(Phil Neville)*
রেফারি সমন্বয়কারী *(Referee Coordinator)*	নিল ব্যারি *(Neil Barry)*

Table 5.8: Example: BnTabQA Input Table. (English translation of each cell is italicized and in parenthesis)

ফুটসাল সমন্বয়কারী (*Futsal Coordinator)*
মাইকেল সকুবালা (*Michael Skubala)*
মাইকেল সকুবালা (*Michael Skubala)*

Example 2 `OdiaG` also overfits to the demonstrations with গণনা (count) operator to generate incorrect value and header:

গণনা('নাম') (*count(Name)*)
১ *(1)*

Example 3 `GPT-3.5` with 2-shot in-context learning (ICL) extracts মাইকেল স্কুবালা (*Michael Skubala)* correctly but generates an incorrect table header over-fitting to the demonstrations:

গণনা('নাম') (*count(Name))*
মাইকেল সকুবালা (*Michael Skubala)*

Example 4 `GPT-4` with 2-shot in-context learning (ICL) correctly generates the answer table:

নাম (*Name)*
মাইকেল সকুবালা (*Michael Skubala)*
লেস রিড (*Les Reed)*

Example 5 Both encoder-decoder models, `BnTQA-mBart` and `BnTQA-M2M`, fine-tuned on BanglaTabQA dataset, correctly generate both answer table headers and values:

নাম (*Name)*
মাইকেল স্কুবালা (*Michael Skubala)*
লেস রিড (*Les Reed)*

Example 6 `BnTQA-Llama`, fine-tuned on BanglaTabQA dataset, is partially correct in its predictions by generating ফুটসাল সমন্বয়কারী (Fulsal Coordinator) in the first row, but incorrectly repeats the same entity instead of লেস রিড (*Les Reed)* in the second row:

নাম (*Name*)
ফুটসাল সমন্বয়কারী (*Fulsal Coordinator*)
ফুটসাল সমন্বয়কারী (*Fulsal Coordinator*)

We observe from the examples that all baselines except `GPT-4` generate incorrect table headers and overfits and mimics the demonstrations, showing a lack of understanding of table structure and reasoning. The BanglaTabQA models perform table reasoning, reflecting the utility and quality of the large-scale BanglaTabQA dataset.

5.8 Zero-Shot Cross-Lingual Transfer Examples

वर्ष (*year*)	शीर्षक (*Title*)	किरदार (*Character*)
2005	फ्लाइटप्लान (*Flight Plan*)	एरिक (*Eric*)
...	...	...
2011	इन टाइम (*In Time*)	हेनरी हैमिल्टन (*Henry Hamilton*)
2011	इन टाइम (*In Time*)	हेनरी हैमिल्टन (*Henry Hamilton*)
2011	इन टाइम (*In Time*)	हेनरी हैमिल्टन (*Henry Hamilton*)
2011	इन टाइम (*In Time*)	हेनरी हैमिल्टन (*Henry Hamilton*)
...	...	...
2014	स्पेस स्टेशन 76 (*Space Station 76*)	टैड (*Ted*)
...	...	...
2014	विंटर्स टेल (*Winter's Tale*)	पीटर लेक के पिता (*Peter Lake's Father*)

Table 5.9: Example: HiTabQA Input Table (English translation of each cell is italicized and in parenthesis)

Example 7 The Hindi question, वर्ष 2011 में कितने शीर्षक हैं? (*How many titles are there in year 2011?*), with Hindi input table, Table 5.9 (English translation is italicized and in parentheses) and target table:

गणना (शीर्षक) *(count(Title))*
४ *(4)*

`BnTQA-mBart` correctly performs table reasoning but generates the answer in Bengali script instead of Devnagari (Hindi) script:

গনণা(শিরোনাম) *(count(Title))*
8 *(4)*

Example 8 However, for Hindi extractive questions like कौनसे प्राप्तकर्ता अधिकतम बार आये हैं? (Which recipient occurs the maximum number of times?), with Hindi input table:

साल (*year*)	प्राप्तकर्ता (*Recipient*)
2016	विनोद भट्ट (*Vinod Bhatt*)
2016	विनोद भट्ट (*Vinod Bhatt*)
2017	तारक महेता [1] (*Tarak Mehta[1]*)

and target table:

প্রাপ্তকর্তা *(Recipient)*
বিনোদ ভট্ট (*Vinod Bhatt)*

`BnTQA-mBart` correctly generates the answer in Hindi:

प्राप्तकर्ता *(Recipient)*
विनोद भट्ट (*Vinod Bhatt)*

5.9 Conclusion

Our work in this chapter introduces tableQA for the low-resource languages. We propose a methodology for large-scale dataset development on limited budget and automatic quality control which can be applied over any low-resource language with a web-presence. We discuss in detail the application of the methodology with an Indic Language, Bengali, for which we release a large-scale dataset, BanglaTabQA. We further demonstrate generalizability of the process with another language, Hindi. We assess the datasets' quality by effectively training different Bengali and Hindi tableQA models and conducting various experiments on model efficacy.

Our studies on different operator classes and zero-shot cross-lingual transfer demonstrate that models trained with our dataset generalize well to unseen tables. Our proposed methodology can promote further research in low-resource tableQA, while our released dataset and models can be used to further explore tableQA for Bengali and Hindi.

Our answer to RQ4, *How to adapt tableQA for low-resourced languages?*, the research question at the heart of this chapter, is that large-scale synthetic dataset generation and models trained on the generated datasets are effective in addressing the low-resource problem and outperform existing multilingual baseline models. Further, cross-lingual transfer in a zero-shot setup demonstrates that by fine-tuning models in one low-resource language transfers reasoning capabilities and knowledge to another, reducing the need to generate large-scale datasets on languages of the same language family.

Chapter Appendices

5.A Limitations

We design a scalable automatic tableQA data generation method and apply it on with two low-resourced languages: Bengali and Hindi. We release two tableQA datasets: BanglaTabQA and HindiTabQA and several models as outcome. Our main results in Table 5.3 demonstrate successful adaptation of neural models to low-resourced tableQA task. Our extensive experimentation on generalizability in Section 5.6.1 and 5.6.2 shows that models trained on the BanglaTabQA dataset performs well across all operator classes and generalize to unseen languages and tables, proving generalizability of the datasets and methodology.

Our dataset methodology is generalizable, but it is limited to languages for which unlabelled tables are available online. For very-low resource languages with low web presence, our method has only limited impact. Also, we used SQUALL templates for query generation, which do not support multi-table operations or complex queries. We leave addressing these challenges to future work.

5.B Ethical Considerations

The task and models proposed in the paper is aimed at closing the gap of resource scarcity in low-resource languages. To do so, we have used existing open-source resources publicly available in the web under MIT, CC-BY-SA-3.0 and MIT, CC-BY-SA-4.0 licenses. Our dataset is generated synthetically data and will be released under MIT, CC-BY-SA-4.0 license. Our synthetic samples use templates from the SQUALL dataset also released under MIT, CC-BY-SA-4.0 license. Our test data splits are manually annotated. We pay each annotator €13.27/hour for their efforts. Further, we have utilized Wikipedia tables from the Huggingface Wikipedia dataset. Wikipedia tables contain information about named-entities, facts and events in the public domain. We do not use any user-specific data or sensitive information. Our models are built over open-source encoder-decoder models and closed-source GPT-3.5. Our work did not explicitly handle any bias which exists in the aforementioned pre-trained models or Wikipedia.

5.C Bengali SQL2NQSim (LaBse fine-tuning) Results

We evaluate semantic similarity of the LaBse model trained on the translated semantic parsing datasets comprising of Bengali SQL and it corresponding Bengali question (Section 5.4.4) and report the validation set results in Table 5.C.1. Both datasets show high semantic similarity among query-question pairs. However, BanglaTabQA have a higher semantic similarity on various distance metrics indicating higher similarity of the query-question pairs compared to HindiTabQA. HindiTabQA lower semantic scores can be attributed to the lower recall scores among query-question pairs leading to lower F1 similarity scores.

Scores	Bengali	Hindi
Accuracy with Cosine-Similarity	91.99	98.67
F1 with Cosine-Similarity	92.30	72.16
Precision with Cosine-Similarity	94.55	77.68
Recall with Cosine-Similarity	90.15	67.36
Avg Precision with Cosine-Similarity	97.79	75.32
Accuracy with Manhattan-Distance	91.97	98.62
F1 with Manhattan-Distance	92.31	70.96
Precision with Manhattan-Distance	93.73	77.15
Recall with Manhattan-Distance	90.94	65.69
Avg Precision with Manhattan-Distance	97.80	74.41
Accuracy with Euclidean-Distance	91.99	98.67
F1 with Euclidean-Distance	92.30	72.16
Precision with Euclidean-Distance	94.55	77.68
Recall with Euclidean-Distance	90.15	67.36
Avg Precision with Euclidean-Distance	97.79	75.32
Accuracy with Dot-Product	91.99	98.67
F1 with Dot-Product	92.30	72.16
Precision with Dot-Product	94.55	77.68
Recall with Dot-Product	90.15	67.36
Avg Precision with Dot-Product	97.79	75.32

Table 5.C.1: Bengali SQL2NQSim validation scores (%).

5.D GPT Prompts

The 2-shot in-context learning prompt with demonstrations to GPT is shown in Prompt 5.D.1 on the next page:

Prompt 5.D.1: 2-Shot ICL Prompt for GPT-3.5/4

আপনি একজন সহায়ক সহকারী যিনি বাংলা প্রশ্নের উত্তর দেন বাংলা টেবিল থেকে বাংলায় উত্তর টেবিল তৈরি করে। m সারি এবং n কলামগুলির একটি টেবিল নিম্নলিখিত প্যাটার্নে লেখা হয়ে: <কলাম> টেবিল হেডার <রো ১> মান ১,১ । মান ১,২ । ... মান ১,n <রো ২> মান ২,১ । ... <রো m> মান m,১ । মান m,২ । ... । মান m,n

উদাহরণ:

১) প্রশ্ন: কটা শিরোনাম কাউন্টডাউন? <কলাম> বছর । শিরোনাম । ভুমিকা <রো ১> 2006 । সি নো ইভল । জেকব গুড নাইট ...<রো ১৩> 2016 । কাউন্টডাউন । লেঃ ট্রোনিন <রো ১8> 2016 । কাউন্টডাউন । লেঃ ট্রোনিন <রো ১৫> 2016 । কাউন্টডাউন । লেঃ ট্রোনিন

উত্তর: <কলাম> গণনা('শিরেনিনাম') <রো ১> ৩

২) প্রশ্ন: কটা বছরে শিরেনিনাম সী নো ইভল? <কলাম> বছর । শিরোনাম । ভুমিকা <রো ১> 2006 । সি নো ইভল । জেকব গুড নাইট <রো ২> 2006 । সি নো ইভল । জেকব গুড নাইট <রো ৩> 2006 । সি নো ইভল । জেকব গুড নাইট ...

উত্তর: <কলাম> গণনা('বছর') <রো ১> ৩

The English translation of the 2-shot prompt for in-context learning (ICL) of GPT-3.5/4 is shown in Prompt 5.D.2:

Prompt 5.D.2: 2-Shot ICL Prompt for GPT-3.5/4 (English translation)

You are a helpful assistant who answers Bengali questions from Bengali tables by generating an answer table. A table of m rows and n columns is written in the following pattern: <column> table header <row 1> value 1,1 | value 1,2 | ... value 1,n <row 2> value 2,1 | ... <row m> value m,1 | value m,2 | ... | value m,n
Examples:
1) **Question:** How many titles are Countdown? <column> year | Title | Role <row 1> 2006 | See No Evil | Jacob Go ... <row 13> 2016 | Countdown | Le Trunin <row 14> 2016 | Countdown | Le Trunin <row 15> 2016 | Countdown | Le Trunin
Answer: <column> count('Title') <row 1> 3
2) **Question:** How many years have See no Evil as titles? <column> year | Title | Role <row 1> 2006 | See No Evil | Jacob Good Night <row 2> 2006 | See No Evil | Jacob Good Night | <row 3> 2006 | See No Evil | Jacob Good Night ...
Answer: <column> count('year') <row 1> 3

5.E Llama-based Model Prompt

The 2-shot in-context learning prompt with demonstrations to the Llama-7B based model, OdiaG, is shown in Prompt 5.E.1 on the next page:

Prompt 5.E.1: 2-Shot ICL Prompt for odiagenAI-bn

Instruction:

আপনি একজন সহায়ক সহকারী যিনি বাংলা টেবিল তৈরি করে বাংলা প্রশ্নের উত্তর দেন। উদাহরণ:

###Input:

কটা শিরোনাম কাউন্টডাউন? <কলাম> বছর । শিরোনাম । ভূমিকা <রো ১> 2014 । সী নো এভল ২ । জেকব গুড নাইট <রো ২> 2016 । কাউন্টডাউন । লেঃ ত্রেনিন <রো ৩> 2016 । কাউন্টডাউন । লেঃ ত্রেনিন

Response:

<কলাম> গণনা(শিরোনাম) <রো ১> ২

###End

###Input:

কটা বছর শিরোনাম সী নো এভল ২? <কলাম> বছর । শিরোনাম । ভূমিকা <রো ১> 2014 । সী নো এভল ২ । জেকব গুড নাইট <রো ২> 2016 । কাউন্টডাউন । লেঃ ত্রেনিন <রো ৩> 2016 । কাউন্টডাউন । লেঃ ত্রেনিন

Response:

<কলাম> গণনা(শিরোনাম) <রো ১> ১

###End

###Input:
{input}

Response:

The English translation of the 2-shot in-context learning prompt with demonstrations to the Llama-7B based model, OdiaG, is shown in Prompt 5.E.2 on the next page:

Prompt 5.E.2: 2-Shot ICL Prompt for odiagenAI-bn (English translation)

Instruction:

You are a helpful assistant who generates answers Bengali table to answer Bengali questions. Examples:

###Input:

How many titles are Countdown? <column> year | Title | Role <row 1> 2014 | See No Evil 2 | Jacob Goodnight <row 2> 2016 | Countdown | Le Trunin <row 3> 2016 | Countdown | Le Trunin

###Response:

<column> count(Title) <row 1> 2

End

###Input:

How many years have See no Evil as titles? <column> year | Title | Role <row 1> 2014 | See No Evil 2 | Jacob Goodnight <row 2> 2016 | Countdown | Le Trunin <row 3> 2016 | Countdown | Le Trunin

Response:

<column> count(year) <row 1> 1

###Input:
{input}

###Response:

6

Parameter-Efficient Sparse Retrievers and Re-rankers using Adapters

While the previous chapters focused on the *reader* module of information seeking systems, this chapter focuses on the *retriever* module. This chapter addresses RQ5 by studying how to balance the efficiency-accuracy trade-off to leverage efficient sparse neural models as first-stage rankers. As prior work focused only on dense bi-encoder [71] models as second stage rankers, this chapter explores the usage of adapters for sparse neural first-stage rankers and re-rankers. Next, domain-adaptation with adapters is explored on the BEIR benchmark to study the effectiveness of adapter-tuning compared to fine-tuning to domain-shift in datasets. Finally, knowledge sharing between rankers and re-rankers is explored to study whether adapters can be utilized effectively to transform a sparse first stage ranker to a re-ranker.

6.1 Introduction

Information Retrieval (IR) systems often aim to return a ranked list of documents ordered with respect to their relevance to a user query. In modern web search engines, there is, in fact, not a single retrieval model, but several ones specialized in diverse information needs such as different search verticals. To add to this complexity, multi-stage retrieval considers effectiveness-efficiency trade-off where first stage retrievers are essential for fast retrieval of potentially relevant candidate documents from a large corpus. Further down the pipeline, re-rankers are added focusing on effectiveness.

With the advent of large Pretrained Language Models (PLM), recent neural retrieval models have millions of parameters. Training, updating, and adapting such models implies significant computing and storage cost calling for efficient methods. Moreover, generalizability across out-of-domain datasets is critical and even when effectively adapted to new domains, full fine-tuning often comes at the expense of large storage and catastrophic forgetting. Fortunately, such research questions have already been studied in the NLP literature [9, 11, 68, 69] with parameter-efficient tuning. In spite

This chapter was published as V. Pal, C. Lassance, H. Déjean, and S. Clinchant. Parameter-efficient sparse retrievers and rerankers using adapters. In J. Kamps, L. Goeuriot, F. Crestani, M. Maistro, H. Joho, B. Davis, C. Gurrin, U. Kruschwitz, and A. Caputo, editors, *Advances in Information Retrieval*, pages 16–31, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-28238-6.

of very recent work exploring parameter-efficient techniques for neural retrieval, the use of adapters in IR has been overlooked. Previous work on dense retriever had mixed results [86] and successful adaptation was achieved for cross-lingual retrieval [117]. Our study aims to complete the examination of adapters for neural IR and investigates it with neural sparse retrievers. We study ablation of adapter layers to analyze whether all layers contribute equally. We examine how adapter-tuned neural sparse retriever SPLADE [46] fares on benchmark IR datasets MS MARCO [142], TREC DL 2019 and 2020 [27] and out-of-domain BEIR datasets [191]. We explore whether the generalizability of SPLADE can be further improved with adapter-tuning on BEIR and out-of-domain dataset, such as TripClick [173]. In addition, we examine knowledge transfer between first stage retrievers and re-rankers with full fine-tuning and adapter-tuning. To the best of our knowledge, this is the first work which studies adapters on sparse retrievers, focuses on sparse models' generalizability and explores knowledge transfer between retrievers in different stages of the retrieval pipeline. In summary, we address the following research questions:

(1) RQ1: What is the efficiency-accuracy trade-off of parameter-efficient fine-tuning with adapters on the sparse retriever model SPLADE?

(2) RQ2: How does each adapter layer ablation affect retrieval effectiveness?

(3) RQ3: Are adapters effective for adapting neural sparse neural retrieval in a new domain?

(4) RQ4: Could adapters be used to share knowledge between re-rankers and first stage rankers?

6.2 Background and Related Work

Parameter efficient transfer learning techniques aim to adapt large pretrained models to downstream tasks using a fraction of training parameters, achieving comparable effectiveness to full fine-tuning. Such methods [68, 69, 110, 161, 175] are memory efficient and scale well to numerous downstream tasks due to the massive reduction in task-specific trainable parameters. This makes them an attractive solution for efficient storage and deployment compared to fully fine-tuned instances. Such methods have been successfully applied to language translation [161], natural language generation [116], Tabular Question Answering [149], and on the GLUE benchmark [57, 175], In spite of all its advantages and a large research footprint in NLP, parameter-efficient methods remain under-explored in IR.

A recent comprehensive study [39] categorizes parameter-efficient transfer learning into 3 categories: 1) Addition based 2) Specification based 3) Reparameterization based. Addition based methods insert intermediate modules into the pretrained model. The newly added modules are adapted to the downstream task while keeping the rest of the pretrained model frozen. The modules can be added vertically by increasing the model depth as observed in Houlsby Adapters [68] and Pfeiffer Adapters [161]. Houlsby Adapters insert small bottle-neck layers after both the multi-head attention and feed-forward layer of each transformer layer which are optimized for NLP tasks

on GLUE benchmark. Pfeiffer Adapter inserts the bottle-neck layer after only the feed-forward layer and has shown comparable effectiveness to fine-tuning on various NLP tasks. Prompt-based adapter methods such as Prefix-tuning [110] prepend continuous task-specific vectors to the input sequence which are optimized as free-parameters. Compacter [89] hypothesizes that the model can be optimized by learning transformations of the bottle-neck layer in a low-rank subspace, leading to less parameters.

Specification based methods fine-tune only a subset of pretrained model parameters to the task-at-hand while keeping the rest of the model frozen. The fine-tuned model parameters can be only the bias terms as observed in BitFit [11], or only cross-attention weights as in the case of Seq2Seq models with X-Attention [51]. Re-parameterization methods transform the pretrained weights into parameter efficient form during training. This is observed in LoRA [69] which optimises rank decomposition matrices of pretrained layer while keeping the original layer frozen.

Recent studies exploring parameter-efficient transfer learning for Information Retrieval show promising results of such techniques for dense retrieval models [86, 117, 132, 190]. Jung et al. [86] study parameter efficient prefix-tuning [110] and LoRA [69] on bi-encoder and cross-encoder dense models. Additionally, they combine the two methods by sequentially optimizing one method for m epochs, freezing it and optimizing the other for n epochs. Their studies show that while cross-encoders with LoRA and LoRA+(50% more parameters compared to LoRA) outperform fine-tuning with TwinBERT [129] and ColBERT [94], parameter-efficient methods *do not outperform fine-tuning* for bi-encoders across all datasets. Litschko et al. [117] use parameter-efficient techniques such as Sparse Fine-Tuning Masks and Adapters for multilingual and cross-lingual retrieval tasks with re-rankers. They train language adapters with Masked Language Modeling (MLM hereafter) task and then task-specific retrieval adapters. This enables the fusion of reranking adapter trained with source language data together with the language adapter of the target language. Concurrent to our work, Tam et al. [190] study parameter-efficient prompt tuning techniques such as Prefix tuning and P-tuning v2, specification-based methods such as BitFit and adapter-tuning with Pfeiffer Adapters on late interaction bi-encoder models such as Dense Passage Retrieval [90] and ColBERT. They are motivated by cross-domain generalization of dense retrievals and achieve better results with P-tuning compared to fine-tuning on the BEIR benchmark. Ma et al. [132] study various parameter-efficient tuning procedures at both retrieval and re-ranking stages. They conduct a comprehensive study of parameter-efficient techniques such as BitFit, Prefix-tuning, Adapters, LoRA, MAM adapters with dense bi-encoders and cross-encoders with BERT-base as the backbone model. Their parameter-efficient techniques achieve comparable effectiveness to fine-tuning on top-20 retrieval accuracy and marginal gains on top-100 retrieval accuracy.

Compared to prior works, our experiments first study the use of adapters for state-of-the-art sparse models such as SPLADE, contrary to previous work that studied dense bi-encoder models.[1] Furthermore, our results show improvements compared to the previous studies. We also studied the case of using distinct adapters for query and document encoders in a "bi-adapter" setting where the same pretrained backbone

[1]To the best of our knowledge the only work involving SPLADE and adapters/freezing layers is [202], which found that freezing the embeddings improves effectiveness.

model is used by both the query and the document encoder but different adapters are trained for the queries and documents. Secondly, we address another research questions ignored by previous work, which is efficient domain adaptation[2] for neural first-stage rankers. We start from a trained neural ranker and study adaptation with adapters on a different domain, such as the ones present in the BEIR benchmark. Finally, we also study parameters sharing between re-rankers and first-stage rankers using adapters, which to our knowledge has not been studied yet.

6.3 Parameter-Efficient Retrieval with Adapters

In this section, we first present the self-attention used in transformers and how the adapters we use for our experiments interact with them. We then introduce the models used for first stage ranking and reranking.

6.3.1 Self-Attention Transformer Layers

Large pretrained language models are based on the transformer architecture composed of N stacked transformer layers . Each transformer layer comprises of a fully connected feed-forward module and a multi-headed self attention module. Each attention layer has a function of query matrix ($Q \in R^{nXd_k}$), a key matrix and a value matrix. The attention can be formally written as:

$$A(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{6.1}$$

where the query Q, key K and value V are parameterized by weight matrices $W_q \in R^{nXd_k}$, $W_k \in R^{nXd_k}$, and $W_v \in R^{nXd_v}$, as $Q = XW_q$, $K = XW_k$ and $V = XW_v$. Each of the N heads has its respective Q_i, V_i and K_i weights and its corresponding attention A_i. The feed-forward layer takes as input a transformation of the concatenation of the N attentions as:

$$FFN(x) = \sigma(XW_1 + b_1)W_2 + b_2 \tag{6.2}$$

where $\sigma(.)$ is the activation function. A residual connection is further added after each attention layer and feed-forward layer.

6.3.2 Adapters

In this paper, we focus on the Houlsby adapter [68], which as described in Section 6.3 can be considered an additive adapter and is depicted in Figure 6.1. An additive adapter inserts trainable parameters in addition to the aforementioned transformer layers. The added modules form a bottle-neck architecture with a down-projection, an up-projection and a non-linear transformation. The size of the bottle-neck controls the number of training parameters in an adapter layer. Additionally, a residual connection is applied

[2]Here we use adaptation as further fine-tuning on the target domain.

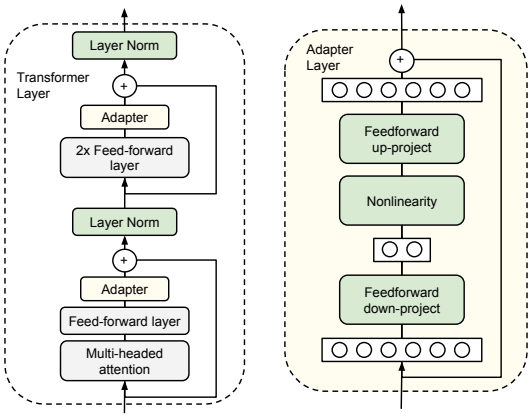


Figure 6.1: Houlsby Adapter, image from the original paper [68].

across each adapter layers. Finally, a layer normalization is added after each transformer sublayer. Formally, this is defined as:

$$x = f(hW_{down})W_{up} + x \tag{6.3}$$

where $x \in R^d$ is the input to the adapter layer, $W_{down} \in R^{dXr}$ is the down projection matrix transforming input x into bottle-neck dimension d, $W_{up} \in R^{rXd}$ is the up projection matrix transforming the bottle-neck representation back to the d-dimensional space. Each adapter layer is initialized with a near-identity weights to enable stable training.

6.3.3 Neural Sparse First-stage Retrievers

Neural sparse first stage retrievers learn contextualized representations of documents and queries in a sparse high-dimensional latent space. In this work, we focus on SPLADE sparse retriever [46, 103], which uses both L_1 and *FLOPS* regularizations to force sparsity. We freeze the pretrained language model while training the adapter layers. SPLADE predicts term weights of each vocabulary token j with respect to an input token i as:

$$w_{ij} = \text{transform}(h_i)^T E_j + b_j \qquad j \in 1, ..., |V| \tag{6.4}$$

where E_j is the j^{th} vocabulary token embedding, b_j is it's bias, h_i is i^{th} input token embedding, $transform(.)$ is a linear transformation followed by GeLU activation and LayerNorm. The final term importance for each vocabulary term j is obtained by taking the maximum predicted weights over the entire input sequence of length n, after applying a log-saturation effect:

$$w_j = \max_n \log(1 + ReLU(w_{ij})) \tag{6.5}$$

Given a query q_i, the ranking score s of a document d is defined by the degree to which it is relevant to q obtained as a dot product $s(q, d) = w(q).w(d)$. The learning objective is to discriminate representations obtained from Equation 6.5 of a relevant document d^+

and non-relevant hard-negatives d^- obtained from BM25 and in-batch negatives $d^-_{i,j}$ by minimizing the contrastive loss:

$$L = -\log \frac{e^{s(q_i,d^+)}}{e^{s(q_i,d_i^+)} + e^{s(q_i,d_i^-)} + \sum_j e^{s(q_i,d_{i,j}^-)}}. \tag{6.6}$$

SPLADE can be further improved with distillation. The learning objective here is to minimize the MarginMSE [46] loss: mean-squared-error between the positive negative margins of a cross-encoder teacher and the student:

$$L = MSE(M_s(q_i, d^+) - M_s(q_i, d^-), M_t(q_i, d^+) - M_t(q_i, d^-)), \tag{6.7}$$

where MSE is mean-squared error, M_t is the teacher's margin and M_s is the student's margin. The final objective optimizes either of the objective in Equation 6.6 or 6.7 with regularization losses:

The Flops regularizer is a smooth relaxation of the average number of floating-point operations necessary to compute the score of a document, and hence directly related to the retrieval time. It is defined using as a continuous relaxation of the activation (i.e. the term has a non zero weight) probability a_j for token j, and estimated for documents d in a batch of size N by $\widehat{a_j^2}$.

Retrieval flops SPLADE also reports the retrieval flops (noted R-FLOPS), i.e., the number of floating point operations on the inverted index to return the list of documents for a given query. The R-FLOPS metric is defined by an estimation of the average number of floating-point operations between a query and a document which is defined as the expectation $\mathbb{E}_{q,d}\left[\sum_{j \in V} p_j^{(q)} p_j^{(d)}\right]$ where p_j is the activation probability for token j in a document d or a query q. It is empirically estimated from a set of approximately 100k development queries, on the MS MARCO collection. It is thus an indication of the inverted index sparsity and of the computational cost for a sparse model (which is different from the inference, i.e. forward cost of the model)

6.3.4 Cross-Encoding Re-rankers

Another way to use PLMs for neural retrieval is to use what is called "cross-encoding" [204]. In this case, both query and document are concatenated before being provided to the network and the score is directly computed by the network. The cross-encoding procedure allows for networks that are much more effective, but this effectiveness comes with a cost on efficiency as the retrieval procedure now has to go through the entire network for each query document pair, instead of being able to precompute document representations and only go through the network for the query representation. The models are trained with a contrastive loss as seen in Equation (6.6) that aims to maximize the score of the true query/document pair compared to a BM25 negative query/document pair, without using in-batch negatives.

6.4 Experimental Setting and Results

We use the SPLADE github repository[3] to implement our modifications and followed the standard procedure to train SPLADE models. We implement our SPLADE models using an L_1 regularization for the query, and $FLOPS$ regularization for the document following [103]. Unless otherwise stated, the document regularization weight λ_d is set to $9\mathrm{e}{-5}$ and the query regularization weight λ_q to $5\mathrm{e}{-4}$ to train all variants of Adapters-SPLADE. In order to mitigate the contribution of the regularizer at the early stages of training, we follow [155] and use a scheduler for λ, quadratically increasing λ at each training iteration, until the $50k$ step. We use a learning rate of $8\mathrm{e}{-5}$, a batch size of 128, a linear scheduler and warmup step of 6000. We set the maximum sequence length to 256. We train for $300k$ iterations and keep the best checkpoint using MRR@10 on the validation set. We use a bottle-neck reduction factor of 16 (i.e. 16 times smaller) for all adapter layers. We use PyTorch [158], Hugging Face Transformers [196] and AdapterHub [9] to train all models on 4 Tesla V100 GPUs with 32GB memory. We compute statistical significance with $p \leq 0.05$ using the Student's t-test and use superscripts to identify statistical significance for almost all measures safe for metrics related to BEIR.[4]

6.4.1 RQ1: Adapters-SPLADE

We study two different settings of encoding with adapters. The first called `adapter`, is a mono-encoder setup where the query and document share a single encoder. The adapter layers are optimized with both the input sequences keeping the PLM frozen. The second setting inspired by the work on [103], is a bi-encoder setup which separates query and document encoders by training distinct query and document adapters on a shared frozen PLM. We call this setting `bi-adapter`. This setting not only benefits from optimizing exclusive adapters for input sequence type (different lengths of query/document, etc.), it is also possible to use smaller PLMs for the queries instead of sharing PLM weights. We explore different backbone PLMs: `DistilBERT` and `CC+MLM Flops`, a pretrained PLM of cocondenser trained on the masked language model (MLM) task using the FLOPS regularization in order to make it easier to work with SPLADE, introduced in [103]. We trained and evaluated Adapter-SPLADE models on the MS MARCO passage ranking dataset [142] in full ranking setting. The results for fine-tuning with BM25 triplets are available in Table 6.1, whereas in Table 6.2 we make available the results of training models with distillation. For distillation, we use hard-negatives and scores generated by a cross-encoder reranker[5] and the MarginMSE loss as described in [46] and set λ_d to $1\mathrm{e}{-2}$ and λ_q to $9\mathrm{e}{-2}$.

To study efficiency-effectiveness trade-off of Adapters-SPLADE, we compare effectiveness, R-FLOPS size and number of training parameters of adapter-tuned models with their baseline fine-tuned counterparts having the same backbone PLM. [155] first showed that R-FLOPs reduction is a reasonable measure of retrieval speed. R-FLOPS measure the average number of floating-point operations needed to compute a document

[3] `https://github.com/naver/splade`

[4] Due to a lack of standard procedure.

[5] `https://huggingface.co/cross-encoder/ms-marco-MiniLM-L-6-v2`

Model	#	Method	MS MARCO dev		TREC DL 2019	TREC DL 2020	R-Flops	Training params
			MRR@10	R@1000	N@10	N@10		
Distil-BERT	a	fine-tuning	0.346	0.963	0.692	0.677	1.43	100%
	b	adapter	0.351	0.968^{a}	0.711	0.676	1.44	2.23%
	c	bi-adapter	0.352	0.967^{a}	0.690	0.666	0.74	2.23%
CC + MLM FLOPS	d	fine-tuning	0.366^{abc}	0.977^{abc}	**0.712**	0.684	1.09	100%
	e	adapter	$\mathbf{0.376}^{abcd}$	$\mathbf{0.980}^{abcdf}$	0.712	0.688	0.8	2.23%
	f	bi-adapter	0.372^{abc}	0.976^{abc}	0.701	**0.700**	0.37	2.23%

Table 6.1: Fine-tuning and adapter-tuning comparison using BM25 triplets for training. N@10 is short for NDCG@10.

Model	#	Method	MS MARCO dev		TREC DL 2019	TREC DL 2020	R-Flops	Training Params
			MRR@10	R@1000	N@10	N@10		
DistilBERT	a	fine-tuning	0.371	0.979^{b}	0.727	0.711	3.93	100%
	b	adapter	0.373	0.975	0.728	0.716	1.86	2.16%
CC + MLM FLOPS	c	fine-tuning	0.388^{ab}	0.982^{ab}	0.734	**0.732**	4.38	100%
	d	adapter	$\mathbf{0.390}^{ab}$	$\mathbf{0.983}^{ab}$	**0.740**	0.729	2.34	2.16%

Table 6.2: Fine-tuning and adapter-tuning comparison using distillation training. N@10 is short for NDCG@10.

score during retrieval. A sparse embedding and subsequently lower FLOP achieves a retrieval speedup of the order of $1/p^2$ over an inverted index where p is the probability of each document embedding dimension being non-zero.

Overall, we observe, from Table 6.1 and 6.2, all variants of adapter-tuned SPLADE outperform all baseline fine-tuned counterparts on MS MARCO and TREC DL 2019. The distilled cocondenser with MLM mono-encoder model is the highest performing with an MRR@10 score of 0.390 and R@100 of 0.983. The difference in effectiveness between the mono-encoder and bi-encoder adapter-tuning is marginal and depends on the PLM. Most noteworthy, we also observe that the R-FLOPS are lower for adapter-tuned models indicating sparser representation than the fine-tuned counterparts. This is more pronounced in the adapter-tuned models with distillation. Finally, the **bi-adapter** models have even lower R-FLOPS than the mono-encoder settings, which shows that for the same effectiveness the bi-adapters models are more efficient and sparse. We also observe that the number of training parameters is only 2.23% of the total model parameters for *triplets* training (1.5M/67M for mono-adapter `DistilBERT`, 3M/135M for bi-adapter `DistilBERT`, 2M/111M for `CC + MLM FLOPS`) and 2.16% for the distillation process (1.5M/67M for mono-adapter `DistilBERT`, 2M/111M for `CC + MLM FLOPS`). This has a direct consequence in low-hardware setting where adapters

Datasets	Triplets training				Distillation training			
	DistilBERT		CC + MLM		DistilBERT		CC + MLM	
	FT	AD	FT	AD	FT	AD	FT	AD
arguana	0.298	0.364	0.427	0.388	0.513	0.443	0.463	0.433
climate-fever	0.167	0.172	0.180	0.187	0.202	0.197	0.229	0.202
dbpedia-entity	0.379	0.392	0.388	0.401	0.419	0.417	0.438	0.432
fever	0.730	0.734	0.724	0.722	0.773	0.757	0.792	0.773
fiqa	0.295	0.289	0.317	0.320	0.332	0.314	0.342	0.337
hotpotqa	0.626	0.647	0.650	0.603	0.687	0.670	0.687	0.629
nfcorpus	0.318	0.321	0.331	0.333	0.335	0.335	0.340	0.344
nq	0.481	0.482	0.506	0.523	0.522	0.508	0.539	0.544
quora	0.819	0.810	0.821	0.806	0.825	0.722	0.841	0.552
scidocs	0.143	0.150	0.151	0.153	0.154	0.147	0.152	0.153
scifact	0.614	0.611	0.658	0.669	0.687	0.658	0.690	0.673
trec-covid	0.694	0.684	0.668	0.689	0.703	0.728	0.700	0.713
webis-touche2020	0.270	0.255	0.277	0.274	0.260	0.258	0.294	0.290
mean	0.449	0.455	0.469	0.467	0.493	0.473	0.500	0.467

Table 6.3: NDCG@10 score comparison on the BEIR zero-shot evaluation. “FT” is short for “fine-tuning,” “AD is short for “adapter.” “CC+MLM” is short for “CC+MLM FLOPS.”

with lower number of number of training parameters and gradients can be trained on a smaller GPU(such as 24GB P40) but full fine-tuning is infeasible. Overall, there is a clear advantage in using Adapter-SPLADE over fine-tuning, which differs from the previous results on dense adapters [86].

We also evaluate with the full BEIR benchmark [41] comprising of 18 different datasets to measure generalizability of IR models with zero-shot effectiveness on out-of-domain data. The results are listed in Table 6.3. We observe from that in the mono-adapter Triplets training, `adapter` outperforms fine-tuning on mean nDCG@10 with the highest gap in arguana. With `CC+MLM Flops` as the backbone model, fine-tuning and adapter-tuning performs similarly. However, adapter scores drop on models trained with distillation. This can be attributed to the adapter representations being sparser compared to the fine-tuned models. As depicted by the R-FLOPS in Table 6.1, adapter-tuned `DistilBERT` has less than half the number of R-FLOPS than its fine-tuned counterpart whereas `CC+MLM Flops` finetuned model has approximately 1.87 times the number of R-FLOPS of the adapter-tuned model. This reflects in model representation capacity in 0-shot setting in Table 6.3. However, as discussed in Section 6.4.3, adapters are well suited for domain adaptation when trained on out-of-domain datasets keeping the backbone retriever intact and free from catastrophic forgetting.

#	Adapters removed	MS MARCO dev		TREC DL 2019	TREC DL 2020	BEIR	R-Flops	Training params	Training time(hrs)
		MRR@10	R@1000	N@10	N@10	N@10			
a	None	**0.351**^{cdefg}	**0.968**^{defg}	**0.711**^{fg}	0.676^{g}	0.455	1.44	2.23%	34.42
b	0	0.348^{defg}	0.967^{efg}	0.708^{fg}	0.674^{g}	0.458	1.27	2.01%	32.23
c	0-1	0.344^{efg}	0.968^{efg}	0.709^{fg}	**0.699**^{abdefg}	0.459	1.34	1.80%	28.55
d	0-2	0.341^{efg}	0.966^{efg}	0.703^{fg}	0.665^{g}	0.459	1.36	1.59%	26.70
e	0-3	0.325^{fg}	0.962^{fg}	0.689	0.660^{g}	0.455	1.50	1.37%	24.18
f	0-4	0.318^{g}	0.956	0.659	0.663^{g}	0.455	1.27	1.15%	22.51
g	0-5	0.312	0.955	0.660	0.617	0.449	2.78	0.90%	21.35

Table 6.4: Adapter layer ablation with `adapters` on `DistilBERT` PLM.

6.4.2 RQ2: Adapter Layer Ablation

Furthermore, we perform extensive adapter layer ablation by progressively removing adapter layers from the early layers of the encoder. Doing so results in n separate models for each layer ablation setting. The frozen pretrained model for our ablation studies is `DistilBERT` in a mono-encoder setting where the same instance of the encoder is used to encode both the document and the query, which is the same configuration as the adapter method in Table 6.1. This results in a total of 6 configurations for the ablation study corresponding to the 6 adapter layers after each pretrained transformer layer. The final experimental setting removes all 6 adapter layers (0–5) and fine-tunes only the language model head.

We note that such an experiment (dropping adapter layers from transformer models) has been studied in NLP [175] and was shown to improve both training and inference time while retaining comparable effectiveness. We report the effectiveness of each adapter ablation setting on MS MARCO, TREC DL 2019 and TREC DL 2020 in Table 6.4. We actually observe a gradual performance drop for MS MARCO and TREC DL datasets as the training parameters decrease with the progressive removal of adapter layers as shown in Table 6.4. The drop is significantly higher (a drop of 0.25 MRR score) when layers are removed from the second half of the model ($\geq$ 0–3). This phenomenon is consistent with studies in NLP [149, 175] that task-specific information is stored in the later layers of the adapters. For the BEIR datasets, this effectiveness drop is not as evident until all adapters but the language model head are removed (configuration 0–5). The last configuration also has less sparsity as observed from the R-FLOPS size of 2.78 compared to the other configurations. We also observe that the training time drops proportional to the drop in adapter layers. The training time for adapter-tune without any drop in adapter layers is 34.42 hours on 4 Tesla V100 GPUS for $150,000$ iterations, and it drops to 26.70 hours with only 1% drop in MRR with the first 0–2 adapter layers dropped. The lowest training time is 21.35 hours with a drop of 3.2% in MRR for the configuration with all adapters dropped but the language model head.

Dataset	Training	Zero Shot		1st Round		2nd Round	
		NDCG @10	Recall @100	NDCG @10	Recall @100	NDCG @10	Recall @100
Fever	fine-tuning	0.793	0.954	0.692	0.866	0.851	0.959
	adapter			0.841	0.960	**0.881**	**0.964**
FiQA	fine-tuning	0.348	0.632	0.371	0.678	0.356	0.694
	adapter			0.373	0.675	**0.393**	**0.711**
NFCorpus	fine-tuning	0.348	0.285	0.384	0.466	**0.403**	**0.484**
	adapter			0.362	0.435	0.371	0.428

Table 6.5: Domain adaptation comparison on the BEIR datasets.

6.4.3 RQ3: Out-of-Domain Dataset Adaptation

For the next research question, we want to check how adapters compare to full fine-tuning when adapting a model trained on MSMARCO on a smaller out-of-domain dataset. We evaluate this question under two scenarios: i) BEIR and ii) TripClick.

BEIR On the BEIR benchmark we use 3 datasets (FEVER, FiQA and NFCorpus) that have training, development and test sets and aim for very different domains and tasks (fact checking , financial QA and bio-medical IR). We start from a pre-fine-tuned SPLADE model called "splade-cocondenser-ensembledistil" made available in [46]. We verify the effectiveness of the models in zero shot and get a first set of hard negatives. These hard negatives are then used to train either via fine-tuning of all parameters or via the introduction of adapters. The networks are trained for either 10 (FEVER) or 100 epochs (FiQA and NFCorpus), and at the end of each epoch, we compute the development set effectiveness. We use the models with the best development set to compute the 1st round test set effectiveness and generate hard negatives that are used for another round of training that we call 2nd round (which repeats the 1st round, starting from the best network of the 1st round and using negatives from the 1st round).

Results are available in Table 6.5. While fine-tuning is not always able to improve the results over the zero-shot, mostly due to overfitting on the training/dev sets. For example, on fever fine-tuning first makes all representations as it can easily overfit to the training even without using many words and only on the second round of training started using more dimensions. On the other hand, adapter tuning is able to consistently improve the effectiveness over the zero shot and first rounds (even if it does not always perform the best, as is the case on NFCorpus). Overall, we conclude that adapters are more stable than fine-tuning when fine-tuning on these specific domains.

TripClick Given that in the BEIR benchmark the adapters underperformed fine-tuning on bio-medical data, we decided to further experiment on a larger bio-medical dataset called TripClick. The TripClick collection [173] contains approximately 1.5 millions MEDLINE documents (title and abstract), and 692,000 queries. The test set is divided into three categories of queries: Head, Torso and Tail (according to their decreasing

#	Training	HEAD (dctr)		HEAD		Torso		Tail	
		N@10	R@100	N@10	R@100	N@10	R@100	N@10	R@100
a	Fine-tuning	0.218	**0.579**	**0.302**	0.523	0.219	**0.679**	0.238	**0.722**
b	Adapter	**0.219**	0.578	0.299	**0.526**	**0.229**^{a}	0.679	**0.253**^{a}	0.720

Table 6.6: Performance of mono-encoder on the out-of-domain Tripclick dataset. "N@10" is short for "NDCG@10." "R@100" is short for "Recall@100."

frequency), which contain 1,175 queries each. For the Head queries, a DCTR click model was employed to created relevance signals, otherwise raw clicks were used. We use the triplets released by [67]. Similarly to the BEIR experiments, we start from the "splade-cocondenser-ensembledistil" SPLADE model and fine-tune or adapt-tune it over 100,000 iterations (batch size equal to 100). As shown in Table 6.6, adapter-tuning shows very competitive results, on par with fine-tuning for head categories (frequent queries), and achieving even better results for the less frequent queries (torso and tail).

6.4.4 RQ4: Knowledge Sharing between re-rankers and First-stage Rankers

The final research question explores sharing knowledge between re-rankers and first-stage rankers. We explore this with transforming first stage rankers into re-rankers. First, we tune the pretrained `DistilBERT` for reranking task as a baseline for both fine-tuning and adapter-tuning. We then test transforming both sparse (`splade-cocondenser`) and dense (`tct_colbert-v2-msmarco`) first stage rankers into re-rankers, using either fine-tuning or adapter-tuning. The cross-encoder is initialized with the weights of the aforementioned first stage models, but the reranker classification head on the `CLS` token is randomly initialized. Note that we rerank the top-1k returned from "splade-cocondenser-ensembledistil" (represented by "first stage" on table).

We compare adapter-tuning with fine-tuning and display the results in Table 6.7. We observe that fine-tuning the baseline model (`DistilBERT`) is better than adapter-tuning. When using first stage rankers, results are varied. Dense first stage re-rankers were able to learn similarly with both adapter and fine-tuning. However, this was not the case for sparse first stage rankers (`splade-cocondenser-ensembledistil`). We posit that this may come from two different reasons: i) The SPLADE model does not focus on the CLS representations, but on the MLM head representations of all tokens, thus needing more flexibility; ii) The model has been trained multiple times (initial BERT training, then condenser, then cocondenser and finally SPLADE), while not always using the same precision (fp16 or fp32), which under preliminary analysis seems to have made some parts of the model unusable for cross-encoding without full fine-tuning. Overall, there is slight gain in using the first-stage model for the reranker. However, there's no increase in effectiveness of using adapters, we actually see worse effectiveness on all settings.

Base Model	#	Training	MS MARCO dev	TREC DL 2019	TREC DL 2020
			MRR@10	NDCG@10	NDCG@10
First Stage	a	None	0.383^{e}	0.732	0.721
DistilBERT	b	fine-tune	0.396^{ace}	**0.764**^{e}	0.736
	c	adapter	0.388^{e}	0.737	0.727
SPLADE++	d	fine-tune	**0.408**^{abceg}	0.753	**0.743**
	e	adapter	0.358	0.723	0.707
TCT Colbert v2	f	fine-tune	0.404^{abce}	0.749	0.731
	g	adapter	0.400^{ace}	0.740	0.739

Table 6.7: Knowledge Sharing between first stage rankers and rerankers comparison between fine-tuning and adapter-tuning.

6.5 Conclusion

Retrieval models, based on pre-trained language models, require fine-tuning millions of parameters which makes them memory inefficient and non-scalable for out-of-domain adaptation. This motivates the need for efficient methods to adapt them to information retrieval tasks. In this chapter, we examine adapters for sparse retrieval models. We show that with approximately 2% of training parameters, adapters can be successfully employed for SPLADE models with comparable or even better effectiveness on benchmark IR datasets such as MS MARCO and TREC. We further analyze adapter layer ablation and see a further reduction in training parameters to 1.8% retains effectiveness of full fine-tuning. For domain adaptation, adapters are more stable and outperform fine-tuning, which is prone to over-fitting, by a large margin on most BEIR datasets.

On the Tripclick dataset, adapters outperform on precision metrics Torso and Tail queries and performs comparably on Head queries. We explore knowledge transfer between first stage rankers and re-rankers as a final study. Adapters underperform full fine-tuning when trying to reuse sparse model to re-rankers. Dense first-stage rankers perform similarly for adapters and fine-tuning while sparse first stage rankers is less effective compared to fine-tuning. We leave this to future work. As memory-efficient adapters are effective for SPLADE, we leave for future studying larger sparse models and their generalizability. Finally, an interesting scenario could also be to tackle unsupervised domain adaptation with adapters.

Returning to RQ5, *How can we balance the efficiency-accuracy trade-off to leverage efficient sparse neural models as first-stage rankers?*, which motivated this chapter, our answer is that adapters are effective in balancing the efficiency-accuracy trade-off for the sparse-retriever model, SPLADE. The experimental results demonstrate the efficacy of adapters compared to fine-tuning on benchmark IR datasets. Adapters also perform better on out-of-domain adaptation compared to fine-tuning. However, transforming the first-stage ranker, SPLADE, to re-ranker produced mixed results with fine-tuning outperforming adapter-tuning on both IR datasets, MS MARCO and TREC DL.

Chapter Appendices

6.A Limitations

Our experiments demonstrated that adapter-tuning is not only more stable and generalizable to out-of-domain datasets compared to fine-tuning, but also outperforms fine-tuning on in-domain data. However, parameter-efficient adapters could not be transformed to a re-ranker successfully while maintaining sparsity. Adapters thus perform poorly compared to fine-tuning for re-ranking.

6.B Ethical Considerations

We use publicly available datasets for all our experiments such as the datasets in the BEIR benchmark and Tripclick. BEIR benchmark contains datasets from various domains and were created in previous research. All these datasets are available under the MIT, CC-BY-SA-3.0 and MIT, CC- BY-SA-4.0 licenses. Our work did not explicitly handle any bias which might exist in the pre-trained models or existing datasets.

7 Conclusions

This chapter concludes the thesis by revisiting the research questions listed in Section 1.1, and summarizing the main findings and considerations in Section 7.1. Further, limitations and future directions are discussed in Section 7.2.

7.1 Main Findings

We summarize the main findings in response to the research questions introduced in Section 1.1 by theme. We start with the **the first theme**, the **reader**:

> How to design and develop a machine comprehension reader over semi-structured tables to aid the information needs of users?

Our first research question under this theme was:

(RQ1) How do language models compare on the generative question answering task when the context is unstructured text or semi-structured tables?

RQ1 was answered by studying the effectiveness of state-of-the-art fine-tuned models with adapter-tuned models. To assess the effect of the introduction of structured input, two table question answering (tableQA) datasets and one text question answering (textQA) dataset were used with both training schemes. Additionally, adapter-layer ablation was performed to investigate whether the adapter layers across the encoder and decoder contribute equally to performance. The main finding of the experimental studies was that for textQA, the adapter-tuned models perform comparably to fine-tuned models, with adapter-tuning achieving only marginally lower scores.

However, the adapter-tuned models outperformed fine-tuned models on one tableQA dataset and achieved very similar scores on the other dataset. The insignificant gains of fine-tuning over adapter-tuning in tableQA can be attributed to catastrophic forgetting induced by differences in the distribution of the downstream tabular data format from the original text data format of pre-training. Further analysis of the adapter-layers was performed by eliminating successive layers from both encoder and decoder modules.

The main observation from this investigation was that as more adapter layers are eliminated, the performance drops across all datasets. However, the performance drop was minimal until the last adapter layers were also deleted. The inflection point varies across datasets but is limited to the last 2 layers of the encoder and decoder.

Next, we turn to:

(RQ2) How can we leverage multiple tabular context to perform complex tabular reasoning to address user needs?

RQ2 was answered by designing a scalable dataset creation methodology and an effective training scheme to leverage the large-scalable dataset created. This process led to the release of a multi-table pre-training dataset comprising $132,645$ samples of SQL queries and tabular answers. Additionally, three multi-table QA fine-tuning datasets were created and released from the existing semantic parsing dataset. This results in $6,715$ training and 985 validation samples from the Spider semantic parsing dataset; 530 training, 49 validation, and 253 test samples from the GeoQuery semantic parsing dataset; and 84 training, 45 validation, and 86 test samples from the Atis semantic parsing dataset.

Further, to assess the language models trained on the generated dataset, table generation evaluation metrics were designed to evaluate the different levels of granularity and strictness of a structured table.

The experimental results demonstrated that such metrics are insightful in understanding model behavior. Ablation experiments on the proposed curriculum learning training scheme showed the efficacy of the methodology, which outperforms fine-tuning by a large margin. Further, analysis on the impact of the number of input tables showed that as the number of multi-tables increases, question complexity increased and subsequently the models' performance degrades.

Next, we turn to:

(RQ3) How to generate summaries over multiple tables for conversational agents?

We answered RQ3 by introducing the query-focused multi-table summarization task. A two-stage methodology was designed to address the task, that comprised of a table serialization module and a summarization controller. Further, a dataset comprising of $4,909$ query-summary pairs was developed. A comprehensive experimental setup was created for supervised fine-tuning, adapter-tuning and prompt-based in-context learning to investigate the efficacy of various models on the task. Additionally, automatic evaluation metrics such as text-based evaluation metrics were analyzed and compared with table-based evaluation metrics.

Human evaluation of the generated summaries on faithfulness and fluency demonstrated a strong correlation of the automatic table metrics with human judgments.

The final research question addressed under the first theme of the thesis was:

(RQ4) How to adapt tableQA for low-resource languages?

RQ4 was answered by introducing the task of low-resourced tableQA and studying two Indo-Aryan languages: Bengali and Hindi. An automatic dataset generation and quality control methodology was designed for large-scale development of low-resource tableQA datasets in a budget-constrained environment. This led to Bengali table question answering (BanglaTabQA), a Bengali tableQA dataset comprising of 19K Wikipedia tables, 2M training, 2K validation and 165 test samples; and Hindi

table question answering (HindiTabQA), a Hindi tableQA dataset comprising of 2K Wikipedia tables, 643K training, 645 validation and 125 test samples.

Various models were trained on the generated data with supervised fine-tuning and few-shot in-context learning, and the models' performance was analyzed and compared.

The findings showed that the full fine-tuned small language models outperform on the low-resource languages compared to LLMs such as GPT-3.5/GPT-4 with in-context learning or Llama with parameter-efficient fine-tuning. Further analysis on zero-shot cross-lingual transfer and on various mathematical operations demonstrates the generalizability of the trained models.

After RQ4, we turned to the **second theme** of the thesis, the **retriever**:

> *How to design an efficient and effective retriever for question answering (QA) on textual data?*

Under this second theme, we addressed the following research question:

(RQ5) How can we balance efficiency-accuracy trade-off to leverage efficient sparse neural models as first-stage rankers?

RQ5 was addressed by investigating two setups of sparse encoding: a mono-encoder setup with shared weights and a bi-encoder setup with distinct weights between the query and document encoders. To investigate the effectiveness-efficiency trade-off, the number of training parameters and R-FLOPS [150] size was compared with the retrieval scores. The experiments showed that the effectiveness between the mono-encoder and bi-encoder was similar, with marginal difference between the two setups. However, the efficiency varies with fully fine-tuned models having larger R-FLOPS compared to adapter-tuned efficient models. Further experiments on distillation of the models demonstrated a higher difference in R-FLOPS. The bi-encoder setup also demonstrated lower R-FLOPS and subsequently was more efficient compared to the mono-encoder setup.

All the experimental results demonstrated the advantage of the proposed Adapter-SPLADE over fine-tuned models. Extensive analysis on adapter layer ablation showed that a significant loss in performance occurs when layers are removed from the later half of the retrieval models. Further, removing all layers, but the last one, also resulted in a loss of sparsity and efficiency.

Investigating the generalizability of the trained sparse retrievers on out-of-domain dataset adaptation showed that adapters are superior and more stable than full fine-tuning on specific domains. Finally, an investigation on knowledge sharing between re-rankers and first-stage rankers was performed by transforming the first-stage rankers into cross-encoder re-rankers. Experimental results showed that while dense first-stage rankers were successfully transformed into a re-ranker for both adapter-tuned and fine-tuned models, the same is not true for sparse first-stage rankers with fine-tuning outperforming adapter-tuning.

7.2 Future Work

Our ideas for potential future work are organized following the chapter and research question structure.

7.2.1 Parameter-efficient Generative QA over Structured Tables and Unstructured Text

Although the study and comparison of fine-tuned and adapter-tuned models across the two input modalities, unstructured text and structured tables, demonstrated the superiority and utility of parameter-efficient tuning for generalizability on structured data, explicit reasoning capabilities of the language models or training schemes were not explored. Further research on reasoning patterns with chain-of-thought needs to be explored to investigate the faithfulness or factuality of the generated answers in addition to the fluency. Additionally, only one table representation has been studied that transforms the structured table into a sequence to be compatible with the expected input format of language models. This disrupts the two-dimensional structure of tables. Further representation of tables is left for future work to investigate the implications of a linearized format.

7.2.2 Multi-table Question Answering

As the first work introducing the multi-table question answering task, the proposed dataset generation methodology from Chapter 3 was aimed for a fully automatic large-scale and low cost dataset generation. This led to the usage of SQL templates from existing semantic parsing datasets that may limit the diversity of the generated SQL queries. More work is needed to compare other dataset creation methodologies, such as query creation from context-free grammar [210]. Additionally, the released multi-table QA models are trained with supervised fine-tuning with no explicit supervision on reasoning patterns. Explicit reasoning supervision with chain-of-thought [194], agentic frameworks and tool [231] usage might help the complex reasoning.

7.2.3 Query Focused Multi-table Summarization

As the first work on multi-table summarization, the study reported in Chapter 4 successfully addressed the handling of complex user queries over multiple tabular contexts. However, human evaluation and analysis established the need for more exploration on better evaluation of summaries from structured inputs. Existing evaluation metrics such as ROUGE [111], BLEU [154, 163], BertScore [217], etc. are insufficient as faithfulness measures and focus on fluency instead. Further, the proposed and released dataset from Chapter 4 was generated using GPT-3.5 for scalability and cost-efficiency. This may limit lexical diversity of the generated answers. Alternative methods such as employing human annotators may improve lexical diversity with syntactic variations. Further, our proposed methodology in Chapter 4 uses in-context learning with fixed prompt using the same few-shot demonstrations. Advanced prompting techniques such as variable

demonstration with retrieval [108], prompt learning [38], tree of thought [127, 203], etc. can be used for better complex reasoning and faithful summary generation.

7.2.4 Low-resourced TableQA

The proposed dataset generation methodology in Chapter 5 to address low-resourced tableQA is language-agnostic and thus generalizable to any low-resourced language with a web presence and is fully automatic and scalable. However, the methodology in Chapter 5 has been applied to only two languages: Hindi and Bengali, and further analysis is required to study phrasal and lexical alignment to other low-resourced languages. The large-scale Bengali and Hindi tableQA datasets developed from the proposed dataset generation methodology are built by instantiating SQL templates from existing semantic parsing datasets. This limits the scope of the queries to the templates and does not handle multi-table operations and very complex queries. More complex datasets are necessary to handle real-world queries. Additionally, the trained models in Chapter 5 show zero-shot cross-lingual transfer between Bengali and Hindi that belong to the same language family. More detailed experiments are required to study this phenomenon in languages from different language families. Further, complex reasoning in tableQA is a challenge and sophisticated techniques such as tree of thought [203], agentic frameworks [231] are necessary to improve mathematical and logical reasoning.

7.2.5 Parameter-efficient Sparse Retrievers and Re-rankers

While the exploration of parameter-efficient first-stage rankers demonstrates the efficiency, effectiveness, and generalizability of Adapter-SPLADE over fine-tuned models, the experimental results of re-rankers showed that dense first-stage re-rankers perform similarly for adapters and fine-tuning while sparse first-stage re-rankers are less effective than fine-tuning. More research is needed to explore cross-encoder re-rankers for sparsity without sacrificing effectiveness. Further, all experimental setups were done on small language models. Larger language models with better generalization capability [15] may address some of the domain-specific adaptation challenges of the smaller models and need to be explored further. Lastly, chapter 6 does not study tables and focuses only on unstructured text. Table retrieval [153, 189, 214, 215] is an essential component of the *retriever-reader* framework of tableQA. Current work on neural table retrieval focuses on dense retrieval [65], while neural sparse retrieval over tables is an open challenge. The lessons learned in Chapter 6 on parameter-efficient sparse retrieval over text need to be investigated on structured tables and are left for future work.

Bibliography

[1] M. Aliannejadi, H. Zamani, F. Crestani, and W. B. Croft. Asking clarifying questions in open-domain information-seeking conversations. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, page 475–484, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450361729. doi: 10.1145/3331184.3331265. URL https://doi.org/10.1145/3331184.3331265. (Cited on page 1.)

[2] E. Andrejczuk, J. Eisenschlos, F. Piccinno, S. Krichene, and Y. Altun. Table-to-text generation and pre-training with TabT5. In Y. Goldberg, Z. Kozareva, and Y. Zhang, editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6758–6766, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.503. URL https://aclanthology.org/2022.findings-emnlp.503. (Cited on page 32.)

[3] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh. Vqa: Visual question answering. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2425–2433, 2015. doi: 10.1109/ICCV.2015.279. (Cited on page 11.)

[4] S. Badugu and R. Manivannan. A study on different closed domain question answering approaches. *Int. J. Speech Technol.*, 23(2):315–325, June 2020. ISSN 1381-2416. doi: 10.1007/s10772-020-09692-0. URL https://doi.org/10.1007/s10772-020-09692-0. (Cited on page 1.)

[5] T. Banerjee and P. Bhattacharyya. Meaningless yet meaningful: Morphology grounded subword-level NMT. In M. Faruqui, H. Schütze, I. Trancoso, Y. Tsvetkov, and Y. Yaghoobzadeh, editors, *Proceedings of the Second Workshop on Subword/Character LEvel Models*, pages 55–60, New Orleans, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-1207. URL https://aclanthology.org/W18-1207. (Cited on page 79.)

[6] R. Baradaran, R. Ghiasi, and H. Amirkhani. A survey on machine reading comprehension systems. *Natural Language Engineering*, 28(6):683–732, 2022. doi: 10.1017/S1351324921000395. (Cited on page 1.)

[7] L. Bauer, Y. Wang, and M. Bansal. Commonsense for generative multi-hop question answering tasks. In E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4220–4230, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1454. URL https://aclanthology.org/D18-1454. (Cited on page 13.)

[8] T. Baumgärtner, L. F. R. Ribeiro, N. Reimers, and I. Gurevych. Incorporating relevance feedback for information-seeking retrieval using few-shot document re-ranking. In *Conference on Empirical Methods in Natural Language Processing*, 2022. URL https://api.semanticscholar.org/CorpusID:252992633. (Cited on page 1.)

[9] T. Beck, B. Bohlender, C. Viehmann, V. Hane, Y. Adamson, J. Khuri, J. Brossmann, J. Pfeiffer, and I. Gurevych. AdapterHub playground: Simple and flexible few-shot learning with adapters. In V. Basile, Z. Kozareva, and S. Stajner, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 61–75, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-demo.6. URL https://aclanthology.org/2022.acl-demo.6. (Cited on pages 91 and 97.)

[10] P. Bellan, M. Dragoni, and C. Ghidini. Leveraging pre-trained language models for conversational information seeking from text, 2022. URL https://arxiv.org/abs/2204.03542. (Cited on page 1.)

[11] E. Ben Zaken, Y. Goldberg, and S. Ravfogel. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-short.1. URL https://aclanthology.org/2022.acl-short.1. (Cited on pages 91 and 93.)

[12] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, page 41–48. Association for Computing Machinery, 2009. ISBN 9781605585161. doi: 10.1145/1553374.1553380. URL https://doi.org/10.1145/1553374.1553380. (Cited on page 32.)

[13] A. Bhattacharjee, T. Hasan, W. Ahmad, K. S. Mubasshir, M. S. Islam, A. Iqbal, M. S. Rahman, and R. Shahriyar. BanglaBERT: Language model pretraining and benchmarks for low-resource language understanding evaluation in Bangla. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1318–1327, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-naacl.98. URL https://aclanthology.org/

2022.findings-naacl.98. (Cited on pages 69, 70, and 72.)

[14] A. Bordes, N. Usunier, S. Chopra, and J. Weston. Large-scale simple question answering with memory networks. *ArXiv*, abs/1506.02075, 2015. URL https://api.semanticscholar.org/CorpusID:9605730. (Cited on page 2.)

[15] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf. (Cited on pages 56, 76, and 111.)

[16] Z. Cai, X. Li, B. Hui, M. Yang, B. Li, B. Li, Z. Cao, W. Li, F. Huang, L. Si, and Y. Li. STAR: SQL guided pre-training for context-dependent text-to-SQL parsing. In Y. Goldberg, Z. Kozareva, and Y. Zhang, editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1235–1247, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.89. URL https://aclanthology.org/2022.findings-emnlp.89. (Cited on pages 3, 30, and 36.)

[17] A. O. Carnegie and A. H. Oh. Stochastic language generation for spoken dialogue systems. In *In Proc. of the ANLP/NAACL 2000 Wrkshp. on Conversational Systems*, pages 27–32, 2000. (Cited on page 11.)

[18] C. H. Chao, X. J. Hou, and Y. C. Chiu. Improve chit-chat and QA sentence classification in user messages of dialogue system using dialogue act embedding. In L.-H. Lee, C.-H. Chang, and K.-Y. Chen, editors, *Proceedings of the 33rd Conference on Computational Linguistics and Speech Processing (ROCLING 2021)*, pages 138–143, Taoyuan, Taiwan, Oct. 2021. The Association for Computational Linguistics and Chinese Language Processing (ACLCLP). URL https://aclanthology.org/2021.rocling-1.19/. (Cited on page 1.)

[19] D. Chen, A. Fisch, J. Weston, and A. Bordes. Reading Wikipedia to answer open-domain questions. In R. Barzilay and M.-Y. Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1171. URL https://aclanthology.org/P17-1171/. (Cited on page 1.)

[20] J. Chen, J.-Y. Pan, C. Faloutsos, and S. Papadimitriou. TSum: Fast, principled table summarization. In *Proceedings of the Seventh International Workshop on Data Mining for Online Advertising*. Association for Computing Machinery, 2013. ISBN 9781450323239. doi: 10.1145/2501040.2501981. URL https://doi.org/10.1145/2501040.2501981. (Cited on page 32.)

[21] S. Chen, Y. Hou, Y. Cui, W. Che, T. Liu, and X. Yu. Recall and learn: Fine-tuning deep pre-trained language models with less forgetting. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7870–7881, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.634. URL https://aclanthology.org/2020.emnlp-main.634. (Cited on page 18.)

[22] W. Chen, J. Chen, Y. Su, Z. Chen, and W. Y. Wang. Logical natural language generation from open-domain tables. In D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7929–7942, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.708. URL https://aclanthology.org/2020.acl-main.708. (Cited on page 45.)

[23] W. Chen, H. Wang, J. Chen, Y. Zhang, H. Wang, S. Li, X. Zhou, and W. Y. Wang. Tabfact: A large-scale dataset for table-based fact verification. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=rkeJRhNYDH. (Cited on pages 1 and 54.)

[24] Z. Cheng, H. Dong, R. Jia, P. Wu, S. Han, F. Cheng, and D. Zhang. FORTAP: Using formulas for numerical-reasoning-aware table pretraining. In S. Muresan, P. Nakov, and A. Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1150–1166, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.82. URL https://aclanthology.org/2022.acl-long.82. (Cited on pages 32 and 68.)

[25] Z. Cheng, H. Dong, Z. Wang, R. Jia, J. Guo, Y. Gao, S. Han, J.-G. Lou, and D. Zhang. HiTab: A hierarchical table dataset for question answering and natural language generation. In *Proceedings of*

the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1094–1110, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.78. URL https://aclanthology.org/2022.acl-long.78. (Cited on pages 13, 30, and 32.)

[26] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel. Palm: scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24(1), Mar. 2024. ISSN 1532-4435. (Cited on page 76.)

[27] N. Craswell, B. Mitra, E. Yilmaz, D. Campos, E. M. Voorhees, and I. Soboroff. Trec deep learning track: reusable test collections in the large data regime. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2369–2375, 2021. (Cited on page 92.)

[28] Y. Cui, Z. Yang, and X. Yao. Efficient and effective text encoding for chinese llama and alpaca. *arXiv preprint arXiv:2304.08177*, 2023. URL https://arxiv.org/abs/2304.08177. (Cited on page 79.)

[29] R. Dabre, H. Shrotriya, A. Kunchukuttan, R. Puduppully, M. Khapra, and P. Kumar. IndicBART: A pre-trained model for indic natural language generation. In S. Muresan, P. Nakov, and A. Villavicencio, editors, *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1849–1863, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.145. URL https://aclanthology.org/2022.findings-acl.145. (Cited on page 76.)

[30] D. A. Dahl, M. Bates, M. Brown, W. Fisher, K. Hunicke-Smith, D. Pallett, C. Pao, A. Rudnicky, and E. Shriberg. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994. URL https://aclanthology.org/H94-1010. (Cited on pages 5, 34, and 71.)

[31] J. Dalton, S. Fischer, P. Owoicho, F. Radlinski, F. Rossetto, J. R. Trippas, and H. Zamani. Conversational information seeking: Theory and application. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 3455–3458, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450387323. doi: 10.1145/3477495.3532678. URL https://doi.org/10.1145/3477495.3532678. (Cited on page 1.)

[32] H. T. Dang. DUC 2005: Evaluation of question-focused summarization systems. In *Proceedings of the Workshop on Task-Focused Summarization and Question Answering*, pages 48–55, 2006. (Cited on page 48.)

[33] A. Das and D. Saha. Deep learning based bengali question answering system using semantic textual similarity. *Multimedia Tools Appl.*, 81(1):589–613, jan 2022. ISSN 1380-7501. doi: 10.1007/s11042-021-11228-w. URL https://doi.org/10.1007/s11042-021-11228-w. (Cited on pages 69 and 70.)

[34] Y. Deldjoo, J. R. Trippas, and H. Zamani. Towards multi-modal conversational information seeking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 1577–1587, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380379. doi: 10.1145/3404835.3462806. URL https://doi.org/10.1145/3404835.3462806. (Cited on pages 1, 11, and 67.)

[35] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer. QLoRA: Efficient finetuning of quantized llms. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2024. Curran Associates Inc. (Cited on page 56.)

[36] B. Dhingra, M. Faruqui, A. Parikh, M.-W. Chang, D. Das, and W. Cohen. Handling divergent reference texts when evaluating table-to-text generation. In A. Korhonen, D. Traum, and L. Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4884–4895, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1483. URL https://aclanthology.org/P19-1483. (Cited on page 57.)

[37] B. Ding, C. Qin, L. Liu, Y. K. Chia, B. Li, S. Joty, and L. Bing. Is GPT-3 a good data annotator? In A. Rogers, J. Boyd-Graber, and N. Okazaki, editors, *Proceedings of the 61st Annual Meeting of the*

Association for Computational Linguistics (Volume 1: Long Papers), pages 11173–11195, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.626. URL https://aclanthology.org/2023.acl-long.626. (Cited on page 52.)

[38] N. Ding, S. Hu, W. Zhao, Y. Chen, Z. Liu, H. Zheng, and M. Sun. OpenPrompt: An open-source framework for prompt-learning. In V. Basile, Z. Kozareva, and S. Stajner, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 105–113, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-demo.10. URL https://aclanthology.org/2022.acl-demo.10/. (Cited on page 111.)

[39] N. Ding, Y. Qin, G. Yang, F. Wei, Z. Yang, Y. Su, S. Hu, Y. Chen, C.-M. Chan, W. Chen, J. Yi, W. Zhao, X. Wang, Z. Liu, H. Zheng, J. Chen, Y. Liu, J. Tang, J. Li, and M. Sun. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *ArXiv*, abs/2203.06904, 2022. (Cited on page 92.)

[40] S. Edunov, A. Baevski, and M. Auli. Pre-trained language model representations for language generation. In J. Burstein, C. Doran, and T. Solorio, editors, *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 4052–4059, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1409. URL https://aclanthology.org/N19-1409/. (Cited on page 1.)

[41] J. M. Eisenschlos, M. Gor, T. Müller, and W. W. Cohen. Mate: Multi-view attention for table transformer efficiency. In *Conference on Empirical Methods in Natural Language Processing*, page 7606–7619. Association for Computational Linguistics, 2021. (Cited on page 32.)

[42] A. Fan, Y. Jernite, E. Perez, D. Grangier, J. Weston, and M. Auli. ELI5: Long form question answering. In A. Korhonen, D. Traum, and L. Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3558–3567, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1346. URL https://aclanthology.org/P19-1346/. (Cited on pages 2, 3, and 6.)

[43] A. Fan, S. Bhosale, H. Schwenk, Z. Ma, A. El-Kishky, S. Goyal, M. Baines, O. Celebi, G. Wenzek, V. Chaudhary, N. Goyal, T. Birch, V. Liptchinsky, S. Edunov, E. Grave, M. Auli, and A. Joulin. Beyond english-centric multilingual machine translation. *J. Mach. Learn. Res.*, 22(1), jan 2021. ISSN 1532-4435. (Cited on page 76.)

[44] F. Feng, Y. Yang, D. Cer, N. Arivazhagan, and W. Wang. Language-agnostic bert sentence embedding. *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2022. doi: 10.18653/v1/2022.acl-long.62. URL http://dx.doi.org/10.18653/v1/2022.acl-long.62. (Cited on page 72.)

[45] J. L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76:378–382, 1971. URL https://api.semanticscholar.org/CorpusID:143544759. (Cited on pages 54 and 75.)

[46] T. Formal, C. Lassance, B. Piwowarski, and S. Clinchant. From distillation to hard negative sampling: Making sparse neural ir models more effective. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 2353–2359, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450387323. doi: 10.1145/3477495.3531857. URL https://doi.org/10.1145/3477495.3531857. (Cited on pages 7, 92, 95, 96, 97, and 101.)

[47] T. Formal, C. Lassance, B. Piwowarski, and S. Clinchant. Towards effective and efficient sparse neural information retrieval. *ACM Trans. Inf. Syst.*, 42(5), Apr. 2024. ISSN 1046-8188. doi: 10.1145/3634912. URL https://doi.org/10.1145/3634912. (Cited on pages 2 and 4.)

[48] R. M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4): 128–135, 1999. ISSN 1364-6613. doi: https://doi.org/10.1016/S1364-6613(99)01294-2. URL https://www.sciencedirect.com/science/article/pii/S1364661399012942. (Cited on page 18.)

[49] J. Gala, P. A. Chitale, A. K. Raghavan, V. Gumma, S. Doddapaneni, A. K. M, J. A. Nawale, A. Sujatha, R. Puduppully, V. Raghavan, P. Kumar, M. M. Khapra, R. Dabre, and A. Kunchukuttan. Indictrans2: Towards high-quality and accessible machine translation models for all 22 scheduled indian languages. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=vfT4YuzAYA. (Cited on page 76.)

[50] J. Gardner, Z. Popović, and L. Schmidt. Benchmarking distribution shift in tabular data with tableshift. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS

'23, Red Hook, NY, USA, 2023. Curran Associates Inc. (Cited on page 1.)

[51] M. Gheini, X. Ren, and J. May. Cross-attention is all you need: Adapting pretrained Transformers for machine translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1754–1765, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.132. URL https://aclanthology.org/2021.emnlp-main.132. (Cited on page 93.)

[52] Z. Gu, J. Fan, N. Tang, P. Nakov, X. Zhao, and X. Du. PASTA: Table-operations aware fact verification via sentence-table cloze pre-training. In Y. Goldberg, Z. Kozareva, and Y. Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4971–4983, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.331. URL https://aclanthology.org/2022.emnlp-main.331. (Cited on page 32.)

[53] W. Guan, I. Smetannikov, and M. Tianxing. Survey on automatic text summarization and transformer models applicability. In *Proceedings of the 2020 1st International Conference on Control, Robotics and Intelligent System*, CCRIS '20, page 176–184, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450388054. doi: 10.1145/3437802.3437832. URL https://doi.org/10.1145/3437802.3437832. (Cited on page 2.)

[54] J. Guo, Z. Zhang, L. Xu, H.-R. Wei, B. Chen, and E. Chen. Incorporating bert into parallel sequence decoding with adapters. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 10843–10854. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/7a6a74cbe87bc60030a4bd041dd47b78-Paper.pdf. (Cited on page 12.)

[55] V. Gupta, M. Mehta, P. Nokhiz, and V. Srikumar. INFOTABS: Inference on tables as semi-structured data. In D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2309–2324, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.210. URL https://aclanthology.org/2020.acl-main.210/. (Cited on page 1.)

[56] V. Gupta, P. Kandoi, M. Vora, S. Zhang, Y. He, R. Reinanda, and V. Srikumar. TempTabQA: Temporal question answering for semi-structured tables. In H. Bouamor, J. Pino, and K. Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2431–2453, Singapore, Dec. 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.149. URL https://aclanthology.org/2023.emnlp-main.149. (Cited on page 67.)

[57] W. Han, B. Pang, and Y. N. Wu. Robust transfer learning with pretrained language models through adapters. In C. Zong, F. Xia, W. Li, and R. Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 854–861, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-short.108. URL https://aclanthology.org/2021.acl-short.108. (Cited on page 92.)

[58] T. Hasan, A. Bhattacharjee, K. Samin, M. Hasan, M. Basak, M. S. Rahman, and R. Shahriyar. Not low-resource anymore: Aligner ensembling, batch filtering, and new datasets for Bengali-English machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2612–2623, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.207. URL https://aclanthology.org/2020.emnlp-main.207. (Cited on page 69.)

[59] P. He, X. Liu, J. Gao, and W. Chen. Deberta: decoding-enhanced bert with disentangled attention. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=XPZIaotutsD. (Cited on page 56.)

[60] X. He, Z. Lin, Y. Gong, A.-L. Jin, H. Zhang, C. Lin, J. Jiao, S. M. Yiu, N. Duan, and W. Chen. AnnoLLM: Making large language models to be better crowdsourced annotators. In Y. Yang, A. Davani, A. Sil, and A. Kumar, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*, pages 165–190, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-industry.15. URL https://aclanthology.org/2024.naacl-industry.15. (Cited on page 52.)

[61] M. A. Hedderich, L. Lange, H. Adel, J. Strötgen, and D. Klakow. A survey on recent approaches for natural language processing in low-resource scenarios. In *Proceedings of the 2021 Conference*

of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 2545–2568, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.201. URL https://aclanthology.org/2021.naacl-main.201. (Cited on pages 4 and 68.)

[62] M. Henderson, B. Thomson, and J. D. Williams. The second dialog state tracking challenge. In K. Georgila, M. Stone, H. Hastie, and A. Nenkova, editors, *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 263–272, Philadelphia, PA, U.S.A., June 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-4337. URL https://aclanthology.org/W14-4337/. (Cited on page 1.)

[63] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. (Cited on page 76.)

[64] J. Herzig, P. K. Nowak, T. Müller, F. Piccinno, and J. Eisenschlos. 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, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.398. URL https://aclanthology.org/2020.acl-main.398. (Cited on pages 2, 3, 6, 11, 13, 29, 30, 32, 68, and 69.)

[65] J. Herzig, T. Müller, S. Krichene, and J. Eisenschlos. Open domain question answering over tables via dense retrieval. In K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 512–519, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.43. URL https://aclanthology.org/2021.naacl-main.43/. (Cited on pages 11 and 111.)

[66] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de Las Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. van den Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, J. W. Rae, O. Vinyals, and L. Sifre. Training compute-optimal large language models, 2022. (Cited on page 76.)

[67] S. Hofstätter, S. Althammer, M. Sertkan, and A. Hanbury. Establishing strong baselines for tripclick health retrieval, 2022. (Cited on page 102.)

[68] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. Parameter-efficient transfer learning for nlp. In K. Chaudhuri and R. Salakhutdinov, editors, *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR, 2019. (Cited on pages 3, 12, 13, 16, 91, 92, 94, and 95.)

[69] E. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models, 2021. (Cited on pages 76, 91, 92, and 93.)

[70] D. A. Hudson and C. D. Manning. GQA: A new dataset for real-world visual reasoning and compositional question answering. *arXiv preprint arXiv:1902.09506*, 2019. (Cited on page 11.)

[71] S. Humeau, K. Shuster, M.-A. Lachaux, and J. Weston. Poly-encoders: Architectures and pre-training strategies for fast and accurate multi-sentence scoring. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SkxgnnNFvH. (Cited on pages 4 and 91.)

[72] C.-C. Hung, A. Lauscher, S. P. Ponzetto, and G. Glavaš. DS-TOD: Efficient domain specialization for task oriented dialog. *arXiv preprint arXiv:2110.08395*, 2021. (Cited on page 12.)

[73] S. Iyer, I. Konstas, A. Cheung, J. Krishnamurthy, and L. Zettlemoyer. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, page 963–973. Association for Computational Linguistics, 2017. doi: 10.18653/v1/p17-1089. URL http://dx.doi.org/10.18653/v1/P17-1089. (Cited on pages 34 and 35.)

[74] M. Iyyer, W.-t. Yih, and M.-W. Chang. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1821–1831, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1167. URL https://aclanthology.org/P17-1167. (Cited on page 67.)

[75] G. Izacard and E. Grave. Leveraging passage retrieval with generative models for open domain question answering. In P. Merlo, J. Tiedemann, and R. Tsarfaty, editors, *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, Online, Apr. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.74.

URL https://aclanthology.org/2021.eacl-main.74. (Cited on page 51.)
[76] N. Jain, V. Gupta, A. Rai, and G. Kumar. TabPert : An effective platform for tabular perturbation. In H. Adel and S. Shi, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 350–360, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-demo.39. URL https://aclanthology.org/2021.emnlp-demo.39/. (Cited on page 1.)
[77] P. Jain, A. Laha, K. Sankaranarayanan, P. Nema, M. M. Khapra, and S. Shetty. A mixed hierarchical attention based encoder-decoder approach for standard table summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 622–627. Association for Computational Linguistics, June 2018. doi: 10.18653/v1/N18-2098. URL https://aclanthology.org/N18-2098. (Cited on pages 32, 45, and 48.)
[78] D. Jannach, T. Schmitz, B. Hofer, and F. Wotawa. Avoiding, finding and fixing spreadsheet errors - a survey of automated approaches for spreadsheet qa. *J. Syst. Softw.*, 94:129–150, 2014. URL https://api.semanticscholar.org/CorpusID:18318715. (Cited on page 1.)
[79] S. K. Jauhar, P. D. Turney, and E. H. Hovy. Tables as semi-structured knowledge for question answering. In *Annual Meeting of the Association for Computational Linguistics*, 2016. (Cited on page 67.)
[80] A. Jena, V. Gupta, M. Shrivastava, and J. Eisenschlos. Leveraging data recasting to enhance tabular reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4512 – 4525. Association for Computational Linguistics, Dec. 2022. URL https://arxiv.org/abs/2211.12641. (Cited on page 32.)
[81] K. Jiang, D. Wu, and H. Jiang. FreebaseQA: A new factoid QA data set matching trivia-style question-answer pairs with Freebase. In J. Burstein, C. Doran, and T. Solorio, editors, *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 318–323, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1028. URL https://aclanthology.org/N19-1028/. (Cited on page 2.)
[82] Z. Jiang, Y. Mao, P. He, G. Neubig, and W. Chen. OmniTab: Pretraining with natural and synthetic data for few-shot table-based question answering. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 932–942. Association for Computational Linguistics, 2022. (Cited on pages 55 and 68.)
[83] N. Jin, J. Siebert, D. Li, and Q. Chen. A survey on table question answering: Recent advances. *arXiv preprint arXiv:2207.05270*, 2022. doi: 10.48550/ARXIV.2207.05270. URL https://arxiv.org/abs/2207.05270. (Cited on pages 29, 30, 32, and 67.)
[84] P. Joshi, S. Santy, A. Budhiraja, K. Bali, and M. Choudhury. The state and fate of linguistic diversity and inclusion in the NLP world. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6282–6293, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.560. URL https://aclanthology.org/2020.acl-main.560. (Cited on pages 4 and 68.)
[85] C. Jun, J. Choi, M. Sim, H. Kim, H. Jang, and K. Min. Korean-specific dataset for table question answering. In N. Calzolari, F. Béchet, P. Blache, K. Choukri, C. Cieri, T. Declerck, S. Goggi, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, J. Odijk, and S. Piperidis, editors, *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 6114–6120, Marseille, France, June 2022. European Language Resources Association. URL https://aclanthology.org/2022.lrec-1.657/. (Cited on page 4.)
[86] E. Jung, J. Choi, and W. Rhee. Semi-siamese bi-encoder neural ranking model using lightweight fine-tuning. In *Proceedings of the ACM Web Conference 2022*, WWW '22, page 502–511, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450390965. doi: 10.1145/3485447.3511978. URL https://doi.org/10.1145/3485447.3511978. (Cited on pages 92, 93, and 99.)
[87] E. Kanoulas, P. Eustratiadis, Y. Li, Y. Lyu, V. Pal, G. Poerwawinata, J. Qiao, and Z. Wang. Agent-centric information access. *arXiv preprint arXiv:2502.19298*, 2025. URL https://arxiv.org/abs/2502.19298.
[88] M. R. Karim, S. Kanti Dey, T. Islam, S. Sarker, M. Hasan Menon, K. Hossain, M. A. Hossain, and S. Decker. DeepHateExplainer: Explainable hate speech detection in under-resourced Bengali language. In *Proceeding of IEEE International Conference on Data Science and Advanced Analytics) (DSAA'2021)*, Porto, Portugal, Oct. 2021. (Cited on page 68.)
[89] R. Karimi Mahabadi, S. Ruder, M. Dehghani, and J. Henderson. Parameter-efficient multi-task fine-

tuning for transformers via shared hypernetworks. In C. Zong, F. Xia, W. Li, and R. Navigli, editors, *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 565–576, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.47. URL https://aclanthology.org/2021.acl-long.47. (Cited on page 93.)

[90] V. Karpukhin, B. Oğuz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550. URL https://aclanthology.org/2020.emnlp-main.550. (Cited on page 93.)

[91] Y. Katsis, S. Chemmengath, V. Kumar, S. Bharadwaj, M. Canim, M. Glass, A. Gliozzo, F. Pan, J. Sen, K. Sankaranarayanan, and S. Chakrabarti. Ait-qa: Question answering dataset over complex tables in the airline industry. *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Track*, 2022. doi: 10.18653/v1/2022.naacl-industry.34. URL http://dx.doi.org/10.18653/v1/2022.naacl-industry.34. (Cited on pages 11, 13, and 67.)

[92] E. Kavaz, A. Puig, and I. Rodríguez. Chatbot-based natural language interfaces for data visualisation: A scoping review. *Applied Sciences*, 13(12), 2023. ISSN 2076-3417. doi: 10.3390/app13127025. URL https://www.mdpi.com/2076-3417/13/12/7025. (Cited on page 3.)

[93] S. Kazi, S. Khoja, and A. Daud. A survey of deep learning techniques for machine reading comprehension. *Artif. Intell. Rev.*, 56(Suppl 2):2509–2569, Sept. 2023. ISSN 0269-2821. doi: 10.1007/s10462-023-10583-4. URL https://doi.org/10.1007/s10462-023-10583-4. (Cited on page 1.)

[94] O. Khattab and M. Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 39–48, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450380164. doi: 10.1145/3397271.3401075. URL https://doi.org/10.1145/3397271.3401075. (Cited on page 93.)

[95] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114 (13):3521–3526, 2017. doi: 10.1073/pnas.1611835114. URL https://www.pnas.org/doi/abs/10.1073/pnas.1611835114. (Cited on page 18.)

[96] H. Y. Koh, J. Ju, M. Liu, and S. Pan. An empirical survey on long document summarization: Datasets, models, and metrics. *ACM Comput. Surv.*, 55(8), Dec. 2022. ISSN 0360-0300. doi: 10.1145/3545176. URL https://doi.org/10.1145/3545176. (Cited on page 2.)

[97] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9781713871088. (Cited on page 50.)

[98] T. Kočiský, J. Schwarz, P. Blunsom, C. Dyer, K. M. Hermann, G. Melis, and E. Grefenstette. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, Dec 2018. ISSN 2307-387X. doi: 10.1162/tacl_a_00023. URL http://dx.doi.org/10.1162/tacl_a_00023. (Cited on pages 6, 14, and 17.)

[99] S. Kulkarni, S. Chammas, W. Zhu, F. Sha, and E. Ie. AQuaMuSe: Automatically generating datasets for query-based multi-document summarization. *CoRR*, abs/2010.12694, 2020. (Cited on page 48.)

[100] J. Kupiec, J. Pedersen, and F. Chen. A trainable document summarizer. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '95, page 68–73, New York, NY, USA, 1995. Association for Computing Machinery. ISBN 0897917146. doi: 10.1145/215206.215333. URL https://doi.org/10.1145/215206.215333. (Cited on page 2.)

[101] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, K. Toutanova, L. Jones, M. Kelcey, M.-W. Chang, A. M. Dai, J. Uszkoreit, Q. Le, and S. Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/. (Cited on page 6.)

[102] M. T. R. Laskar, M. Rahman, I. Jahan, E. Hoque, and J. X. Huang. CQSumDP: A ChatGPT-Annotated resource for query-focused abstractive summarization based on debatepedia. *CoRR*, abs/2305.06147,

2023. (Cited on page 48.)

[103] C. Lassance and S. Clinchant. An efficiency study for splade models. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2220–2226, 2022. (Cited on pages 95 and 97.)

[104] R. Lebret, D. Grangier, and M. Auli. Neural text generation from structured data with application to the biography domain. In J. Su, K. Duh, and X. Carreras, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213, Austin, Texas, Nov. 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1128. URL https://aclanthology.org/D16-1128. (Cited on pages 45 and 51.)

[105] C.-H. Lee, O. Polozov, and M. Richardson. Kaggledbqa: Realistic evaluation of text-to-sql parsers. *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)*, 2021. doi: 10.18653/v1/2021.acl-long.176. URL http://dx.doi.org/10.18653/v1/2021.acl-long.176. (Cited on page 30.)

[106] K. Lee, S. Salant, T. Kwiatkowski, A. Parikh, D. Das, and J. Berant. Learning recurrent span representations for extractive question answering. *arXiv preprint arXiv:1611.01436*, 2016. (Cited on page 13.)

[107] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*, pages 7871–7880, 2020. (Cited on pages 14, 16, and 55.)

[108] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546. (Cited on pages 51 and 111.)

[109] T. Li, Z. Wang, L. Shao, X. Zheng, X. Wang, and J. Su. A sequence-to-sequence&set model for text-to-table generation. In A. Rogers, J. Boyd-Graber, and N. Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5358–5370, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.330. URL https://aclanthology.org/2023.findings-acl.330/. (Cited on page 2.)

[110] X. L. Li and P. Liang. 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, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.353. URL https://aclanthology.org/2021.acl-long.353. (Cited on pages 92 and 93.)

[111] C.-Y. Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013. (Cited on pages 54 and 110.)

[112] C.-Y. Lin and E. Hovy. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, page 71–78, USA, 2003. Association for Computational Linguistics. doi: 10.3115/1073445.1073465. URL https://doi.org/10.3115/1073445.1073465. (Cited on page 56.)

[113] K. Lin, B. Bogin, M. Neumann, J. Berant, and M. Gardner. Grammar-based neural text-to-sql generation, 2019. URL https://arxiv.org/abs/1905.13326. (Cited on page 6.)

[114] W. Lin, R. Blloshmi, B. Byrne, A. de Gispert, and G. Iglesias. An inner table retriever for robust table question answering. In A. Rogers, J. Boyd-Graber, and N. Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9909–9926, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.551. URL https://aclanthology.org/2023.acl-long.551. (Cited on page 67.)

[115] Y. Lin, T. Ruan, J. Liu, and H. Wang. A survey on neural data-to-text generation. *IEEE Transactions on Knowledge and Data Engineering*, 2023. (Cited on page 54.)

[116] Z. Lin, A. Madotto, and P. Fung. Exploring versatile generative language model via parameter-efficient transfer learning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 441–459, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.41. URL https://aclanthology.org/2020.findings-emnlp.41. (Cited on pages 12, 13, and 92.)

[117] R. Litschko, I. Vulić, and G. Glavaš. Parameter-efficient neural reranking for cross-lingual and multilingual retrieval. In N. Calzolari, C.-R. Huang, H. Kim, J. Pustejovsky, L. Wanner, K.-S. Choi, P.-M. Ryu, H.-H. Chen, L. Donatelli, H. Ji, S. Kurohashi, P. Paggio, N. Xue, S. Kim, Y. Hahm, Z. He, T. K. Lee, E. Santus, F. Bond, and S.-H. Na, editors, *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1071–1082, Gyeongju, Republic of Korea, Oct. 2022. International Committee on Computational Linguistics. URL https://aclanthology.org/2022.coling-1.90. (Cited on pages 92 and 93.)

[118] A. Liu, H. Dong, N. Okazaki, S. Han, and D. Zhang. PLOG: Table-to-logic pretraining for logical table-to-text generation. In Y. Goldberg, Z. Kozareva, and Y. Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5531–5546, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.373. URL https://aclanthology.org/2022.emnlp-main.373. (Cited on pages 45 and 49.)

[119] J. Liu, D. Shen, Y. Zhang, B. Dolan, L. Carin, and W. Chen. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning inside out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, 2022. (Cited on page 63.)

[120] P. Liu, L. Zhang, and J. A. Gulla. Pre-train, prompt, and recommendation: A comprehensive survey of language modeling paradigm adaptations in recommender systems. *Transactions of the Association for Computational Linguistics*, 11:1553–1571, 12 2023. ISSN 2307-387X. doi: 10.1162/tacl_a_00619. URL https://doi.org/10.1162/tacl_a_00619. (Cited on page 1.)

[121] Q. Liu, B. Chen, J. Guo, M. Ziyadi, Z. Lin, W. Chen, and J. Lou. TAPEX: table pre-training via learning a neural SQL executor. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=O50443AsCP. (Cited on pages 3, 29, 30, 32, 33, 38, 49, 55, 67, 68, 69, and 74.)

[122] R. Liu, S. Yuan, A. Dai, L. Shen, T. Zhu, M. Chen, and X. He. Few-shot table understanding: A benchmark dataset and pre-training baseline. In N. Calzolari, C.-R. Huang, H. Kim, J. Pustejovsky, L. Wanner, K.-S. Choi, P.-M. Ryu, H.-H. Chen, L. Donatelli, H. Ji, S. Kurohashi, P. Paggio, N. Xue, S. Kim, Y. Hahm, Z. He, T. K. Lee, E. Santus, F. Bond, and S.-H. Na, editors, *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3741–3752, Gyeongju, Republic of Korea, Oct. 2022. International Committee on Computational Linguistics. URL https://aclanthology.org/2022.coling-1.329. (Cited on page 32.)

[123] S. Liu, J. Cao, R. Yang, and Z. Wen. Long text and multi-table summarization: Dataset and method. In Y. Goldberg, Z. Kozareva, and Y. Zhang, editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1995–2010, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.145. URL https://aclanthology.org/2022.findings-emnlp.145. (Cited on pages 45 and 48.)

[124] W. Liu, J. Qin, R. Tang, and B. Chen. Neural re-ranking for multi-stage recommender systems. In *Proceedings of the 16th ACM Conference on Recommender Systems*, RecSys '22, page 698–699, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392785. doi: 10.1145/3523227.3547369. URL https://doi.org/10.1145/3523227.3547369. (Cited on page 4.)

[125] Y. Liu, J. Gu, N. Goyal, X. Li, S. Edunov, M. Ghazvininejad, M. Lewis, and L. Zettlemoyer. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742, 2020. doi: 10.1162/tacl_a_00343. URL https://aclanthology.org/2020.tacl-1.47. (Cited on pages 71 and 76.)

[126] Y. Liu, Z. Wang, and R. Yuan. Querysum: A multi-document query-focused summarization dataset augmented with similar query clusters. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):18725–18732, Mar. 2024. doi: 10.1609/aaai.v38i17.29836. URL https://ojs.aaai.org/index.php/AAAI/article/view/29836. (Cited on page 48.)

[127] J. Long. Large language model guided tree-of-thought, 2023. URL https://arxiv.org/abs/2305.08291. (Cited on page 111.)

[128] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7. (Cited on page 56.)

[129] W. Lu, J. Jiao, and R. Zhang. Twinbert: Distilling knowledge to twin-structured compressed bert models for large-scale retrieval. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '20, page 2645–2652, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450368599. doi: 10.1145/3340531.3412747. URL

https://doi.org/10.1145/3340531.3412747. (Cited on page 93.)
[130] Y. Luo, F. Lu, V. Pal, and D. Graus. Enhancing resume content extraction in question answering systems through T5 model variants. In *RecSys in HR'23: The 3rd Workshop on Recommender Systems for Human Resources*, CEUR Workshop Proceedings, 2023.
[131] K. Ma, H. Cheng, X. Liu, E. Nyberg, and J. Gao. Open domain question answering with a unified knowledge interface. In S. Muresan, P. Nakov, and A. Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1605–1620, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.113. URL https://aclanthology.org/2022.acl-long.113. (Cited on pages 30 and 32.)
[132] X. Ma, J. Guo, R. Zhang, Y. Fan, and X. Cheng. Scattered or connected? An optimized parameter-efficient tuning approach for information retrieval. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, CIKM '22, page 1471–1480, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392365. doi: 10.1145/3511808.3557445. URL https://doi.org/10.1145/3511808.3557445. (Cited on page 93.)
[133] A. Magueresse, V. Carles, and E. Heetderks. Low-resource languages: A review of past work and future challenges. *CoRR*, abs/2006.07264, 2020. URL https://arxiv.org/abs/2006.07264. (Cited on page 2.)
[134] R. Marinelli. Proper names and polysemy: From a lexicographic experience. In M. T. Lino, M. F. Xavier, F. Ferreira, R. Costa, and R. Silva, editors, *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal, May 2004. European Language Resources Association (ELRA). URL https://aclanthology.org/L04-1359/. (Cited on page 2.)
[135] A. Masry, X. L. Do, J. Q. Tan, S. Joty, and E. Hoque. ChartQA: A benchmark for question answering about charts with visual and logical reasoning. In S. Muresan, P. Nakov, and A. Villavicencio, editors, *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2263–2279, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.177. URL https://aclanthology.org/2022.findings-acl.177/. (Cited on page 1.)
[136] B. Min, H. Ross, E. Sulem, A. P. B. Veyseh, T. H. Nguyen, O. Sainz, E. Agirre, I. Heintz, and D. Roth. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Comput. Surv.*, 56(2), Sept. 2023. ISSN 0360-0300. doi: 10.1145/3605943. URL https://doi.org/10.1145/3605943. (Cited on page 1.)
[137] B. Minhas, A. Shankhdhar, V. Gupta, D. Aggarwal, and S. Zhang. XInfoTabS: Evaluating multilingual tabular natural language inference. In *Proceedings of the Fifth Fact Extraction and VERification Workshop (FEVER)*, pages 59–77, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.fever-1.7. URL https://aclanthology.org/2022.fever-1.7. (Cited on pages 1, 68, 70, 79, and 80.)
[138] R. Mitra. A generative approach to question answering. *arXiv preprint arXiv:1711.06238*, 2017. (Cited on page 13.)
[139] N. S. Moosavi, A. Rücklé, D. Roth, and I. Gurevych. SciGen: A dataset for reasoning-aware text generation from scientific tables. In *NeurIPS*, 2021. (Cited on page 51.)
[140] W. Myers, T. Etchart, and N. Fulda. Conversational scaffolding: An analogy-based approach to response prioritization in open-domain dialogs. In *International Conference on Agents and Artificial Intelligence*, 2020. URL https://api.semanticscholar.org/CorpusID:215756688. (Cited on page 1.)
[141] L. Nan, C. Hsieh, Z. Mao, X. V. Lin, N. Verma, R. Zhang, W. Kryściński, H. Schoelkopf, R. Kong, X. Tang, M. Mutuma, B. Rosand, I. Trindade, R. Bandaru, J. Cunningham, C. Xiong, and D. Radev. FeTaQA: Free-form Table Question Answering. *Transactions of the Association for Computational Linguistics*, 10:35–49, 01 2022. ISSN 2307-387X. doi: 10.1162/tacl_a_00446. URL https://doi.org/10.1162/tacl_a_00446. (Cited on pages 3, 11, 13, 14, 17, 29, 30, 32, 49, 67, and 69.)
[142] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng. MS MARCO: A human generated machine reading comprehension dataset. In T. R. Besold, A. Bordes, A. S. d'Avila Garcez, and G. Wayne, editors, *CoCo@NIPS*, volume 1773 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016. (Cited on pages 92 and 97.)
[143] K. Nishida, I. Saito, K. Nishida, K. Shinoda, A. Otsuka, H. Asano, and J. Tomita. Multi-style generative reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association*

for Computational Linguistics, pages 2273–2284. Association for Computational Linguistics, July 2019. doi: 10.18653/v1/P19-1220. URL https://aclanthology.org/P19-1220. (Cited on page 17.)

[144] R. Nogueira, W. Yang, K. Cho, and J. J. Lin. Multi-stage document ranking with BERT. *arXiv*, abs/1910.14424, 2019. URL https://api.semanticscholar.org/CorpusID:207758365. (Cited on page 4.)

[145] R. Nogueira, Z. Jiang, and J. Lin. Investigating the limitations of transformers with simple arithmetic tasks. *arXiv preprint arXiv:2102.13019*, 2021. (Cited on page 37.)

[146] OpenAI, J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, R. Avila, I. Babuschkin, S. Balaji, V. Balcom, P. Baltescu, H. Bao, M. Bavarian, J. Belgum, I. Bello, J. Berdine, G. Bernadett-Shapiro, C. Berner, L. Bogdonoff, O. Boiko, M. Boyd, A.-L. Brakman, G. Brockman, T. Brooks, M. Brundage, K. Button, T. Cai, R. Campbell, A. Cann, B. Carey, C. Carlson, R. Carmichael, B. Chan, C. Chang, F. Chantzis, D. Chen, S. Chen, R. Chen, J. Chen, M. Chen, B. Chess, C. Cho, C. Chu, H. W. Chung, D. Cummings, J. Currier, Y. Dai, C. Decareaux, T. Degry, N. Deutsch, D. Deville, A. Dhar, D. Dohan, S. Dowling, S. Dunning, A. Ecoffet, A. Eleti, T. Eloundou, D. Farhi, L. Fedus, N. Felix, S. P. Fishman, J. Forte, I. Fulford, L. Gao, E. Georges, C. Gibson, V. Goel, T. Gogineni, G. Goh, R. Gontijo-Lopes, J. Gordon, M. Grafstein, S. Gray, R. Greene, J. Gross, S. S. Gu, Y. Guo, C. Hallacy, J. Han, J. Harris, Y. He, M. Heaton, J. Heidecke, C. Hesse, A. Hickey, W. Hickey, P. Hoeschele, B. Houghton, K. Hsu, S. Hu, X. Hu, J. Huizinga, S. Jain, S. Jain, J. Jang, A. Jiang, R. Jiang, H. Jin, D. Jin, S. Jomoto, B. Jonn, H. Jun, T. Kaftan, Łukasz Kaiser, A. Kamali, I. Kanitscheider, N. S. Keskar, T. Khan, L. Kilpatrick, J. W. Kim, C. Kim, Y. Kim, H. Kirchner, J. Kiros, M. Knight, D. Kokotajlo, Łukasz Kondraciuk, A. Kondrich, A. Konstantinidis, K. Kosic, G. Krueger, V. Kuo, M. Lampe, I. Lan, T. Lee, J. Leike, J. Leung, D. Levy, C. M. Li, R. Lim, M. Lin, S. Lin, M. Litwin, T. Lopez, R. Lowe, P. Lue, A. Makanju, K. Malfacini, S. Manning, T. Markov, Y. Markovski, B. Martin, K. Mayer, A. Mayne, B. McGrew, S. M. McKinney, C. McLeavey, P. McMillan, J. McNeil, D. Medina, A. Mehta, J. Menick, L. Metz, A. Mishchenko, P. Mishkin, V. Monaco, E. Morikawa, D. Mossing, T. Mu, M. Murati, O. Murk, D. Mély, A. Nair, R. Nakano, R. Nayak, A. Neelakantan, R. Ngo, H. Noh, L. Ouyang, C. O'Keefe, J. Pachocki, A. Paino, J. Palermo, A. Pantuliano, G. Parascandolo, J. Parish, E. Parparita, A. Passos, M. Pavlov, A. Peng, A. Perelman, F. de Avila Belbute Peres, M. Petrov, H. P. de Oliveira Pinto, Michael, Pokorny, M. Pokrass, V. Pong, T. Powell, A. Power, B. Power, E. Proehl, R. Puri, A. Radford, J. Rae, A. Ramesh, C. Raymond, F. Real, K. Rimbach, C. Ross, B. Rotsted, H. Roussez, N. Ryder, M. Saltarelli, T. Sanders, S. Santurkar, G. Sastry, H. Schmidt, D. Schnurr, J. Schulman, D. Selsam, K. Sheppard, T. Sherbakov, J. Shieh, S. Shoker, P. Shyam, S. Sidor, E. Sigler, M. Simens, J. Sitkin, K. Slama, I. Sohl, B. Sokolowsky, Y. Song, N. Staudacher, F. P. Such, N. Summers, I. Sutskever, J. Tang, N. Tezak, M. Thompson, P. Tillet, A. Tootoonchian, E. Tseng, P. Tuggle, N. Turley, J. Tworek, J. F. C. Uribe, A. Vallone, A. Vijayvergiya, C. Voss, C. Wainwright, J. J. Wang, A. Wang, B. Wang, J. Ward, J. Wei, C. Weinmann, A. Welihinda, P. Welinder, J. Weng, L. Weng, M. Wiethoff, D. Willner, C. Winter, S. Wolrich, H. Wong, L. Workman, S. Wu, J. Wu, M. Wu, K. Xiao, T. Xu, S. Yoo, K. Yu, Q. Yuan, W. Zaremba, R. Zellers, C. Zhang, M. Zhang, S. Zhao, T. Zheng, J. Zhuang, W. Zhuk, and B. Zoph. Gpt-4 technical report, 2023. (Cited on pages 56 and 76.)

[147] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9781713871088. (Cited on pages 50 and 56.)

[148] K. K. Pal and C. Baral. Investigating numeracy learning ability of a text-to-text transfer model. In M.-F. Moens, X. Huang, L. Specia, and S. W.-t. Yih, editors, *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3095–3101, Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.265. URL https://aclanthology.org/2021.findings-emnlp.265. (Cited on page 37.)

[149] V. Pal, E. Kanoulas, and M. de Rijke. Parameter-efficient abstractive question answering over tables or text. In *Proceedings of the Second DialDoc Workshop on Document-grounded Dialogue and Conversational Question Answering*, pages 41–53. Association for Computational Linguistics, May 2022. URL https://aclanthology.org/2022.dialdoc-1.5. (Cited on pages 30, 32, 49, 67, 69, 92, and 100.)

[150] V. Pal, C. Lassance, H. Déjean, and S. Clinchant. Parameter-efficient sparse retrievers and rerankers

using adapters. In J. Kamps, L. Goeuriot, F. Crestani, M. Maistro, H. Joho, B. Davis, C. Gurrin, U. Kruschwitz, and A. Caputo, editors, *Advances in Information Retrieval*, pages 16–31, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-28238-6. (Cited on page 109.)

[151] V. Pal, A. Yates, E. Kanoulas, and M. de Rijke. MultiTabQA: Generating tabular answers for multi-table question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6322–6334, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.348. URL https://aclanthology.org/2023.acl-long.348. (Cited on pages 5, 52, 55, 64, 68, 69, 72, 74, 76, and 78.)

[152] V. Pal, E. Kanoulas, A. Yates, and M. de Rijke. Table question answering for low-resourced Indic languages. In Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 75–92, Miami, Florida, USA, Nov. 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.5. URL https://aclanthology.org/2024.emnlp-main.5.

[153] F. Pan, M. Canim, M. Glass, A. Gliozzo, and P. Fox. CLTR: An end-to-end, transformer-based system for cell level table retrieval and table question answering, 2021. URL https://arxiv.org/abs/2106.04441. (Cited on page 111.)

[154] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In P. Isabelle, E. Charniak, and D. Lin, editors, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040. (Cited on pages 56 and 110.)

[155] B. Paria, C. Yeh, I. E. Yen, N. Xu, P. Ravikumar, and B. Póczos. Minimizing flops to learn efficient sparse representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=SygpC6Ntvr. (Cited on page 97.)

[156] S. Parida, S. Sekhar, G. S. Kohli, A. Sen, and S. Sahoo. Bengali instruction-tuning model. https://huggingface.co/OdiaGenAI, 2023. (Cited on page 76.)

[157] P. Pasupat and P. Liang. Compositional semantic parsing on semi-structured tables. In C. Zong and M. Strube, editors, *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, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1142. URL https://aclanthology.org/P15-1142. (Cited on pages 30, 36, 70, and 76.)

[158] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: an imperative style, high-performance deep learning library. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2019. Curran Associates Inc. (Cited on page 97.)

[159] K. Pearce, T. Zhan, A. Komanduri, and J. Zhan. A comparative study of transformer-based language models on extractive question answering. *arXiv preprint arXiv:2110.03142*, 2021. (Cited on page 13.)

[160] G. Penha, S. Vakulenko, O. Dusek, L. Clark, V. Pal, and V. Adlakha. The seventh workshop on search-oriented conversational artificial intelligence (SCAI 2022). In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 3466–3469, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450387323. doi: 10.1145/3477495.3531700. URL https://doi.org/10.1145/3477495.3531700.

[161] J. Pfeiffer, I. Vulić, I. Gurevych, and S. Ruder. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.617. URL https://aclanthology.org/2020.emnlp-main.617. (Cited on pages 12, 13, and 92.)

[162] J. Philip, A. Berard, M. Gallé, and L. Besacier. Monolingual adapters for zero-shot neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4465–4470, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.361. URL https://aclanthology.org/2020.emnlp-main.361. (Cited on page 12.)

[163] M. Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers, WMT*, pages 186–191, 2018. (Cited on page 110.)

[164] J. Prager. Open-domain question–answering. *Foundations and Trends® in Information Retrieval*, 1 (2):91–231, 2007. ISSN 1554-0669. doi: 10.1561/1500000001. URL http://dx.doi.org/10.1561/1500000001. (Cited on page 1.)

[165] P. J. Price. Evaluation of spoken language systems: the ATIS domain. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*, 1990. URL https://aclanthology.org/H90-1020. (Cited on pages 5, 34, and 71.)

[166] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019. (Cited on page 12.)

[167] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. SQuAD: 100,000+ questions for machine comprehension of text. In J. Su, K. Duh, and X. Carreras, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, Nov. 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://aclanthology.org/D16-1264. (Cited on page 13.)

[168] O. Rambow, S. Bangalore, and M. Walker. Natural language generation in dialog systems. In *Proceedings of the First International Conference on Human Language Technology Research*, 2001. URL https://aclanthology.org/H01-1055. (Cited on page 11.)

[169] G. Ramesh, S. Doddapaneni, A. Bheemaraj, M. Jobanputra, R. AK, A. Sharma, S. Sahoo, H. Diddee, M. J, D. Kakwani, N. Kumar, A. Pradeep, S. Nagaraj, K. Deepak, V. Raghavan, A. Kunchukuttan, P. Kumar, and M. S. Khapra. Samanantar: The largest publicly available parallel corpora collection for 11 Indic languages. *Transactions of the Association for Computational Linguistics*, 10:145–162, 2022. doi: 10.1162/tacl_a_00452. URL https://aclanthology.org/2022.tacl-1.9. (Cited on page 72.)

[170] A. Ratnaparkhi. Trainable approaches to surface natural language generation and their application to conversational dialog systems. *Computer Speech & Language*, 16(3):435–455, 2002. ISSN 0885-2308. doi: https://doi.org/10.1016/S0885-2308(02)00025-6. URL https://www.sciencedirect.com/science/article/pii/S0885230802000256. Spoken Language Generation. (Cited on page 11.)

[171] S. Reddy, D. Chen, and C. D. Manning. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266, 2019. doi: 10.1162/tacl_a_00266. URL https://aclanthology.org/Q19-1016. (Cited on page 13.)

[172] N. Reimers and I. Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In K. Inui, J. Jiang, V. Ng, and X. Wan, editors, *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 3982–3992, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410/. (Cited on page 4.)

[173] N. Rekabsaz, O. Lesota, M. Schedl, J. Brassey, and C. Eickhoff. TripClick: The log files of a large health web search engine. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2507–2513, 2021. doi: 10.1145/3404835.3463242. (Cited on pages 92 and 101.)

[174] O. Rubin, J. Herzig, and J. Berant. Learning to retrieve prompts for in-context learning. In M. Carpuat, M.-C. de Marneffe, and I. V. Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2655–2671, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.191. URL https://aclanthology.org/2022.naacl-main.191. (Cited on page 63.)

[175] A. Rücklé, G. Geigle, M. Glockner, T. Beck, J. Pfeiffer, N. Reimers, and I. Gurevych. AdapterDrop: On the efficiency of adapters in transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7930–7946, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.626. URL https://aclanthology.org/2021.emnlp-main.626. (Cited on pages 12, 17, 92, and 100.)

[176] S. Ruder. The 4 biggest open problems in NLP. https://www.ruder.io/4-biggest-open-problems-in-nlp, 2019. (Cited on pages 4 and 68.)

[177] I. A. Sag, T. Baldwin, F. Bond, A. A. Copestake, and D. Flickinger. Multiword expressions: A pain in the neck for nlp. In A. F. Gelbukh, editor, *CICLing*, volume 2276 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2002. ISBN 3-540-43219-1. URL http://dblp.uni-trier.de/db/conf/cicling/cicling2002.html#SagBBCF02. (Cited on page 2.)

[178] J. Seltzer, K. Cheng, S. Zong, and J. Lin. Flipping the script: Inverse information seeking dialogues for market research. *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2022. URL https://api.semanticscholar.org/CorpusID:250210994. (Cited on page 1.)

[179] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016. (Cited on page 13.)

[180] I. V. Serban, A. García-Durán, C. Gulcehre, S. Ahn, S. Chandar, A. Courville, and Y. Bengio. Generating factoid questions with recurrent neural networks: The 30M factoid question-answer corpus. In K. Erk and N. A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 588–598, Berlin, Germany, Aug. 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1056. URL https://aclanthology.org/P16-1056/. (Cited on page 2.)

[181] A. Shankarampeta, V. Gupta, and S. Zhang. Enhancing tabular reasoning with pattern exploiting training. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing*, page 706–726. Association for Computational Linguistics, 2022. (Cited on page 32.)

[182] W. Shen, J. Wang, and J. Han. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering*, 27(2):443–460, 2015. doi: 10.1109/TKDE.2014.2327028. (Cited on page 2.)

[183] W. Shen, Y. Li, Y. Liu, J. Han, J. Wang, and X. Yuan. Entity linking meets deep learning: Techniques and solutions. *IEEE Transactions on Knowledge and Data Engineering*, 35(3):2556–2578, 2023. doi: 10.1109/TKDE.2021.3117715. (Cited on page 2.)

[184] T. Shi, C. Zhao, J. Boyd-Graber, H. Daumé III, and L. Lee. On the potential of lexico-logical alignments for semantic parsing to SQL queries. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, page 1849–1864. Association for Computational Linguistic, 2020. (Cited on pages 34 and 70.)

[185] A. Shigarov. Table understanding: Problem overview. *WIREs Data Mining and Knowledge Discovery*, 13(1):e1482, 2023. doi: https://doi.org/10.1002/widm.1482. URL https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1482. (Cited on page 32.)

[186] I. Stelmakh, Y. Luan, B. Dhingra, and M.-W. Chang. ASQA: Factoid questions meet long-form answers. In Y. Goldberg, Z. Kozareva, and Y. Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8273–8288, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.566. URL https://aclanthology.org/2022.emnlp-main.566. (Cited on page 57.)

[187] D. Su, X. Li, J. Zhang, L. Shang, X. Jiang, Q. Liu, and P. Fung. Read before generate! faithful long form question answering with machine reading. In S. Muresan, P. Nakov, and A. Villavicencio, editors, *Findings of the Association for Computational Linguistics: ACL 2022*, pages 744–756, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.61. URL https://aclanthology.org/2022.findings-acl.61/. (Cited on page 2.)

[188] L. H. Suadaa, H. Kamigaito, K. Funakoshi, M. Okumura, and H. Takamura. Towards table-to-text generation with numerical reasoning. In C. Zong, F. Xia, W. Li, and R. Navigli, editors, *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 1451–1465, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.115. URL https://aclanthology.org/2021.acl-long.115. (Cited on pages 45 and 51.)

[189] Y. Sun, Z. Yan, D. Tang, N. Duan, and B. Qin. Content-based table retrieval for web queries. *Neurocomputing*, 349:183–189, 2019. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2018.10.033. URL https://www.sciencedirect.com/science/article/pii/S0925231218312219. (Cited on page 111.)

[190] W. Tam, X. Liu, K. Ji, L. Xue, J. Liu, T. Li, Y. Dong, and J. Tang. Parameter-efficient prompt tuning makes generalized and calibrated neural text retrievers. In H. Bouamor, J. Pino, and K. Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 13117–13130, Singapore, Dec. 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.874. URL https://aclanthology.org/2023.findings-emnlp.874. (Cited on page 93.)

[191] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, and I. Gurevych. Beir: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. (Cited on page 92.)

[192] H. Touvron, L. Martin, and et al. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023. doi: 10.48550/arXiv.2307.09288. (Cited on pages 56 and 76.)

[193] S. Vakulenko, K. Revoredo, C. Di Ciccio, and M. de Rijke. QRFA: A data-driven model of information-seeking dialogues. In *Advances in Information Retrieval: 41st European Conference on IR Research, ECIR 2019, Cologne, Germany, April 14–18, 2019, Proceedings, Part I*, page 541–557, Berlin, Heidelberg, 2019. Springer-Verlag. ISBN 978-3-030-15711-1. doi: 10.1007/978-3-030-15712-8_35. URL https://doi.org/10.1007/978-3-030-15712-8_35. (Cited on page 11.)

[194] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9781713871088. (Cited on pages 50 and 110.)

[195] T.-H. Wen, M. Gašić, N. Mrkšić, P.-H. Su, D. Vandyke, and S. Young. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In L. Màrquez, C. Callison-Burch, and J. Su, editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721, Lisbon, Portugal, Sept. 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1199. URL https://aclanthology.org/D15-1199. (Cited on page 11.)

[196] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al. 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, 2020. (Cited on page 97.)

[197] X. Wu, J. Zhang, and H. Li. Text-to-Table: A new way of information extraction. *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2518–2533, 2022. doi: 10.18653/v1/2022.acl-long.180. URL http://dx.doi.org/10.18653/v1/2022.acl-long.180. (Cited on pages 2 and 32.)

[198] X. Xu, T. Tohti, and A. Hamdulla. A survey of machine reading comprehension methods. In *2022 International Conference on Asian Language Processing (IALP)*, pages 312–317, 2022. doi: 10.1109/IALP57159.2022.9961260. (Cited on page 1.)

[199] Y. Xu and M. Lapata. Coarse-to-fine query focused multi-document summarization. In B. Webber, T. Cohn, Y. He, and Y. Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3632–3645, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.296. URL https://aclanthology.org/2020.emnlp-main.296. (Cited on page 48.)

[200] Y. Xu and M. Lapata. Generating query focused summaries from query-free resources. In C. Zong, F. Xia, W. Li, and R. Navigli, editors, *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 6096–6109, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.475. URL https://aclanthology.org/2021.acl-long.475. (Cited on page 48.)

[201] J. Yang, A. Gupta, S. Upadhyay, L. He, R. Goel, and S. Paul. TableFormer: Robust transformer modeling for table-text encoding. In S. Muresan, P. Nakov, and A. Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 528–537, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.40. URL https://aclanthology.org/2022.acl-long.40/. (Cited on page 1.)

[202] J.-H. Yang, X. Ma, and J. Lin. Sparsifying sparse representations for passage retrieval by top-k masking. *arXiv preprint arXiv:2112.09628*, 2021. (Cited on page 93.)

[203] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan. Tree of thoughts: deliberate problem solving with large language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc. (Cited on page 111.)

[204] A. Yates, R. Nogueira, and J. Lin. Pretrained transformers for text ranking: Bert and beyond. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 1154–1156, 2021. (Cited on page 96.)

[205] Y. Ye, B. Hui, M. Yang, B. Li, F. Huang, and Y. Li. Large language models are versatile decomposers: Decomposing evidence and questions for table-based reasoning. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, page 174–184, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450394086.

doi: 10.1145/3539618.3591708. URL https://doi.org/10.1145/3539618.3591708. (Cited on pages 68 and 69.)

[206] J. Yin, X. Jiang, Z. Lu, L. Shang, H. Li, and X. Li. Neural generative question answering. *Proceedings of the Workshop on Human-Computer Question Answering*, 2016. doi: 10.18653/v1/w16-0106. URL http://dx.doi.org/10.18653/v1/W16-0106. (Cited on page 13.)

[207] P. Yin, G. Neubig, W.-t. Yih, and S. Riedel. TaBERT: Pretraining for joint understanding of textual and tabular data. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020. doi: 10.18653/v1/2020.acl-main.745. URL http://dx.doi.org/10.18653/v1/2020.acl-main.745. (Cited on pages 2, 3, 11, 30, 32, and 69.)

[208] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, Z. Zhang, and D. Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921. Association for Computational Linguistics, Oct.-Nov. 2018. doi: 10.18653/v1/D18-1425. URL https://aclanthology.org/D18-1425. (Cited on pages 3, 5, 30, 31, 33, 51, 68, and 71.)

[209] T. Yu, R. Zhang, H. Y. Er, S. Li, E. Xue, B. Pang, X. V. Lin, Y. C. Tan, T. Shi, Z. Li, Y. Jiang, M. Yasunaga, S. Shim, T. Chen, A. Fabbri, Z. Li, L. Chen, Y. Zhang, S. Dixit, V. Zhang, C. Xiong, R. Socher, W. S. Lasecki, and D. Radev. Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases, 2019. URL https://arxiv.org/abs/1909.05378. (Cited on page 3.)

[210] T. Yu, C.-S. Wu, X. V. Lin, B. Wang, Y. C. Tan, X. Yang, D. Radev, R. Socher, and C. Xiong. GraPPa: Grammar-augmented pre-training for table semantic parsing. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2009.13845. (Cited on pages 34 and 110.)

[211] Y. Yu, Y. Zhuang, J. Zhang, Y. Meng, A. Ratner, R. Krishna, J. Shen, and C. Zhang. Large language model as attributed training data generator: a tale of diversity and bias. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2024. Curran Associates Inc. (Cited on page 52.)

[212] J. M. Zelle and R. J. Mooney. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, AAAI'96, page 1050–1055. AAAI Press, 1996. ISBN 026251091X. (Cited on pages 5, 34, and 71.)

[213] L. Zha, J. Zhou, L. Li, R. Wang, Q. Huang, S. Yang, J. Yuan, C. Su, X. Li, A. Su, T. Zhang, C. Zhou, K. Shou, M. Wang, W. Zhu, G. Lu, C. Ye, Y. Ye, W. Ye, Y. Zhang, X. Deng, J. Xu, H. Wang, G. Chen, and J. Zhao. TableGPT: Towards unifying tables, nature language and commands into one GPT, 2023. (Cited on page 76.)

[214] S. Zhang and K. Balog. Ad hoc table retrieval using semantic similarity. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, page 1553–1562, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee. ISBN 9781450356398. doi: 10.1145/3178876.3186067. URL https://doi.org/10.1145/3178876.3186067. (Cited on page 111.)

[215] S. Zhang and K. Balog. Web table extraction, retrieval, and augmentation: A survey. *ACM Trans. Intell. Syst. Technol.*, 11(2), Jan. 2020. ISSN 2157-6904. doi: 10.1145/3372117. URL https://doi.org/10.1145/3372117. (Cited on page 111.)

[216] S. Zhang, Z. Dai, K. Balog, and J. Callan. Summarizing and exploring tabular data in conversational search. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 1537–1540, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450380164. doi: 10.1145/3397271.3401205. URL https://doi.org/10.1145/3397271.3401205. (Cited on pages 11, 14, 17, 30, 32, 45, and 48.)

[217] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi. BERTScore: Evaluating Text Generation with BERT. In *ICLR*. OpenReview.net, 2020. (Cited on pages 56 and 110.)

[218] W. Zhang, V. Pal, J. Huang, E. Kanoulas, and M. de Rijke. QFMTS: generating query-focused summaries over multi-table inputs. In U. Endriss, F. S. Melo, K. Bach, A. J. B. Diz, J. M. Alonso-Moral, S. Barro, and F. Heintz, editors, *ECAI 2024 - 27th European Conference on Artificial Intelligence, 19-24 October 2024, Santiago de Compostela, Spain - Including 13th Conference on Prestigious Applications of Intelligent Systems (PAIS 2024)*, volume 392 of *Frontiers in Artificial Intelligence and Applications*, pages 3875–3882. IOS Press, 2024. doi: 10.3233/FAIA240951. URL https://doi.org/10.3233/FAIA240951. (Cited on pages 68 and 69.)

[219] X. Zhang, J. Zhang, Z. Ma, Y. Li, B. Zhang, G. Li, Z. Yao, K. Xu, J. Zhou, D. Zhang-Li, J. Yu, S. Zhao,

J. Li, and J. Tang. TableLLM: Enabling tabular data manipulation by llms in real office usage scenarios. *CoRR*, abs/2403.19318, 2024. URL http://dblp.uni-trier.de/db/journals/corr/corr2403.html#abs-2403-19318. (Cited on page 1.)

[220] Y. Zhao, Y. Li, C. Li, and R. Zhang. MultiHiertt: Numerical reasoning over multi hierarchical tabular and textual data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6588–6600, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.454. URL https://aclanthology.org/2022.acl-long.454. (Cited on pages 2 and 68.)

[221] Y. Zhao, Z. Qi, L. Nan, B. Mi, Y. Liu, W. Zou, S. Han, R. Chen, X. Tang, Y. Xu, D. Radev, and A. Cohan. QTSumm: Query-focused summarization over tabular data. In H. Bouamor, J. Pino, and K. Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1157–1172, Singapore, Dec. 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.74. URL https://aclanthology.org/2023.emnlp-main.74. (Cited on pages 46, 48, 51, 52, 54, 56, 58, 68, and 69.)

[222] Y. Zhao, H. Zhang, S. Si, L. Nan, X. Tang, and A. Cohan. Investigating table-to-text generation capabilities of large language models in real-world information seeking scenarios. In M. Wang and I. Zitouni, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 160–175, Singapore, Dec. 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-industry.17. URL https://aclanthology.org/2023.emnlp-industry.17. (Cited on page 52.)

[223] Y. Zhao, C. Zhao, L. Nan, Z. Qi, W. Zhang, X. Tang, B. Mi, and D. Radev. RobuT: A systematic study of table QA robustness against human-annotated adversarial perturbations. In A. Rogers, J. Boyd-Graber, and N. Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6064–6081, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.334. URL https://aclanthology.org/2023.acl-long.334/. (Cited on page 1.)

[224] V. Zhong, C. Xiong, and R. Socher. Seq2SQL: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103, 2017. (Cited on pages 30, 36, and 71.)

[225] F. Zhou, M. Hu, H. Dong, Z. Cheng, F. Cheng, S. Han, and D. Zhang. TaCube: Pre-computing data cubes for answering numerical-reasoning questions over tabular data. In Y. Goldberg, Z. Kozareva, and Y. Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2278–2291, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.145. URL https://aclanthology.org/2022.emnlp-main.145. (Cited on page 32.)

[226] Y. Zhou, X. Liu, K. Zhou, and J. Wu. Table-based fact verification with self-adaptive mixture of experts. In S. Muresan, P. Nakov, and A. Villavicencio, editors, *Findings of the Association for Computational Linguistics: ACL 2022*, pages 139–149, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.13. URL https://aclanthology.org/2022.findings-acl.13. (Cited on pages 1 and 32.)

[227] F. Zhu, W. Lei, Y. Huang, C. Wang, S. Zhang, J. Lv, F. Feng, and T.-S. Chua. TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance. In C. Zong, F. Xia, W. Li, and R. Navigli, editors, *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 3277–3287, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.254. URL https://aclanthology.org/2021.acl-long.254. (Cited on pages 2, 3, 11, 13, 29, 30, 32, 67, 68, and 74.)

[228] F. Zhu, W. Lei, C. Wang, J. Zheng, S. Poria, and T. Chua. Retrieving and reading: A comprehensive survey on open-domain question answering. *CoRR*, abs/2101.00774, 2021. URL https://arxiv.org/abs/2101.00774. (Cited on page 1.)

[229] G. Zhu and C. A. Iglesias. Exploiting semantic similarity for named entity disambiguation in knowledge graphs. *Expert Systems with Applications*, 101:8–24, 2018. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2018.02.011. URL https://www.sciencedirect.com/science/article/pii/S0957417418300897. (Cited on page 2.)

[230] Y. Zhu, H. Yuan, S. Wang, J. Liu, W. Liu, C. Deng, H. Chen, Z. Liu, Z. Dou, and J.-R. Wen. Large language models for information retrieval: A survey, 2024. URL https://arxiv.org/abs/2308.07107. (Cited on page 1.)

[231] Y. Zhuang, Y. Yu, K. Wang, H. Sun, and C. Zhang. ToolQA: A dataset for LLM question answering with external tools. In *Proceedings of the 37th International Conference on Neural Information*

Processing Systems, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc. (Cited on pages 110 and 111.)

Summary

Information seeking systems are usually designed with two main components: a *Retriever* which retrieves a small subset of relevant information sources that is utilized by a *Reader* which provides a precise answer to the user's query to satisfy the user requirement. Although the Web contains different modalities of information, focus has been primarily on unstructured text with little attention to various heterogeneous information sources. This motivates the investigation of information seeking systems from heterogeneous sources of information, such as structured tables in addition to unstructured text. This thesis inquires two broad research themes: *developing a machine comprehension reader on semi-structured tables to aid the information needs of users* and *designing an efficient and effective retriever to aid question answering on unstructured text?*. Most of this thesis focuses on the first aspect, with the last chapter focusing on the second aspect.

Chapter 2 studies generative question answering with either structured tables or unstructured text. Additionally, Chapter 2 investigates how supervised fine-tuning compares with adapter-tuning on text-based pre-trained models. Quantitative analysis such as adapter layer ablation and qualitative analysis studies the effectiveness-efficiency trade-off and the effect that the structured input modality and subsequent domain shift have on the model performance.

Chapter 3 introduces the task of QA over multiple table contexts. To effectively train language models for the task, a curriculum learning approach was designed with first training the language model on simple SQL queries over single tables, then training on complex SQL queries but over multiple tables, and finally training on natural language questions over multiple tables. Multiple datasets were developed to aid in all the training stages. Further, automatic evaluation metrics were designed to assess the performance of the trained models. Extensive experiments and ablation studies demonstrate the effectiveness of the proposed methodology, datasets, and trained models.

Chapter 4 introduces the query-focused multi-table summarization task. A two-stage prompting methodology was developed and datasets created to aid the task. Extensive experiments with various training strategies were explored, such as supervised fine-tuning, LoRA adapter-tuning, and in-context learning. Lastly, automatic evaluation metrics were analyzed and compared with human judgments.

Chapter 5 introduces the task of low-resourced tableQA and studies two Indo-Aryan languages: Bengali and Hindi. As the tableQA task over low-resource setting is severely resource-scarce, an automatic dataset generation methodology was developed to aid in large-scale dataset creation over any language with Wikipedia presence. This process led to the creation of two large-scale datasets, for Bengali and Hindi, respectively. The models trained on these datasets were superior in performance to existing multi-lingual baseline models and state-of-the-art large language models. Further, analysis on the performance of various mathematical operator classes and experiments on zero-shot learning showed the generalizability of the trained models.

Chapter 6 investigates the efficiency-accuracy trade-off by introducing a parameter-efficient sparse encoding mechanism with first-stage neural retriever SPLADE called Adapter-SPLADE. Adapter-SPLADE was explored in two setups, the mono-encoder and the bi-encoder setup. Experimental results demonstrate the superiority of Adaper-

SPLADE compared to full fine-tuning. Additionally, other training techniques such as distillation were conducted, showing higher variance in the flops which captures the sparsity of the encodings. Additionally, analysis on adapter layer ablation and out-of-domain generalizability was performed. Finally, investigation on whether knowledge can be shared between re-rankers and first-stage rankers was conducted.

Samenvatting

Informatiezoeksystemen worden meestal ontworpen met twee hoofdcomponenten: een *retriever* die een kleine subset van relevante informatiebronnen ophaalt, die wordt gebruikt door een *reader* die een nauwkeurig antwoord geeft op de vraag van de gebruiker om aan de gebruikersbehoefte te voldoen. Hoewel het web verschillende informatiemodaliteiten kent, ligt de focus voornamelijk op ongestructureerde tekst met weinig aandacht voor diverse heterogene informatiebronnen. Dit motiveert het onderzoek naar informatiezoeksystemen met heterogene informatiebronnen, zoals gestructureerde tabellen naast ongestructureerde tekst. Dit proefschrift onderzoekt twee brede onderzoeksthema's: *het ontwikkelen van een machine-comprehension reader met semi-gestructureerde tabellen om te voldoen aan de informatiebehoeften van gebruikers* en *het ontwerpen van een efficiënte en effectieve retriever om het beantwoorden van vragen in ongestructureerde tekst te ondersteunen*. Het grootste deel van dit proefschrift richt zich op het eerste aspect, terwijl het laatste hoofdstuk zich richt op het tweede aspect.

Hoofdstuk 2 bestudeert generatieve vraag-antwoord systemen met gestructureerde tabellen of ongestructureerde tekst. Daarnaast onderzoekt hoofdstuk 2 hoe gesuperviseerde fine-tuning zich verhoudt tot adapter-tuning op tekstgebaseerde, vooraf getrainde modellen. Kwantitatieve analyse, zoals ablatie van de adapterlaag en kwalitatieve analyse, bestudeert de afweging tussen effectiviteit en efficiëntie en het effect van de gestructureerde invoermodaliteit en de daaropvolgende domeinverschuiving op de modelprestaties.

Hoofdstuk 3 introduceert de vraag-antwoordtaak over meerdere tabelcontexten. Om taalmodellen effectief voor deze taak te trainen, werd een curriculumleeraanpak ontworpen, waarbij het taalmodel eerst werd getraind met eenvoudige SQL-query's over enkele tabellen, vervolgens met complexe SQL-query's maar over meerdere tabellen, en tot slot met vragen in natuurlijke taal over meerdere tabellen. Er werden meerdere datasets ontwikkeld ter ondersteuning van alle trainingsfasen. Verder werden automatische evaluatiemetrieken ontworpen om de prestaties van de getrainde modellen te beoordelen. Uitgebreide experimenten en ablatiestudies tonen de effectiviteit van de voorgestelde methodologie, datasets en getrainde modellen aan.

Hoofdstuk 4 introduceert de querygerichte multi-tabelsamenvattingstaak. Er werd een tweestaps prompting-methodologie ontwikkeld en datasets gecreëerd om de taak te ondersteunen. Er werden uitgebreide experimenten met verschillende trainingsstrategieën uitgevoerd, zoals supervised fine-tuning, LoRA-adapter-tuning en in-context leren. Ten slotte werden automatische evaluatiemetrieken geanalyseerd en vergeleken met menselijke oordelen.

Hoofdstuk 5 introduceert de taak van tableQA met beperkte middelen en bestudeert twee Indo-Arische talen: Bengaals en Hindi. Omdat de tableQA-taak in een omgeving met beperkte middelen zeer schaars is, werd een methodologie voor automatische datasetgeneratie ontwikkeld ter ondersteuning van grootschalige datasetcreatie voor elke taal met een aanwezigheid in Wikipedia. Dit proces leidde tot de creatie van twee grootschalige datasets, respectievelijk voor het Bengaals en Hindi. De modellen die op deze datasets werden getraind, presteerden beter dan bestaande meertalige basismodellen en state-of-the-art grote taalmodellen. Verder toonden analyses van de prestaties van verschillende wiskundige operatorklassen en experimenten met zero-shot

learning de generaliseerbaarheid van de getrainde modellen aan.

Hoofdstuk 6 onderzoekt de afweging tussen efficiëntie en nauwkeurigheid door de introductie van een parameterefficiënt sparse-encodingmechanisme met de neurale retriever SPLADE in de eerste fase, genaamd Adapter-SPLADE. Adapter-SPLADE werd onderzocht in twee opstellingen: de mono-encoder en de bi-encoder. Experimentele resultaten tonen de superioriteit van Adaper-SPLADE aan ten opzichte van volledige finetuning. Daarnaast werden andere trainingstechnieken, zoals distillatie, toegepast, wat een hogere variantie in de flops aantoonde, wat de schaarste van de coderingen weergeeft. Daarnaast werd analyse uitgevoerd op de ablatie van de adapterlaag en generaliseerbaarheid buiten het domein. Tot slot werd onderzocht of kennis gedeeld kan worden tussen re-rankers en first-stage rankers.