

Structured Contexts For Large Language Models

Kevin Lin



Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2024-218

<http://www2.eecs.berkeley.edu/Pubs/TechRpts/2024/EECS-2024-218.html>

December 18, 2024

Copyright © 2024, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

Structured Contexts For Large Language Models

By

Kevin Lin

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Dan Klein, Co-chair
Associate Professor Joseph E. Gonzalez, Co-chair
Assistant Professor Alane Suhr
Associate Professor Mohit Iyyer

Fall 2024

Structured Contexts For Large Language Models

Copyright 2024
by
Kevin Lin

Abstract

Structured Contexts For Large Language Models

by

Kevin Lin

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Dan Klein, Co-chair

Associate Professor Joseph E. Gonzalez, Co-chair

Large language models (LLMs) are as capable as the context that they are given. This dissertation studies how to structure context, improving LLMs by orchestrating and engineering the contexts that they are given. First, we present context decomposition, a technique for breaking complex contexts into simpler contexts that specialized models are more capable of handling. Second, we show in a comparative study that, context rewriting, a method for re-representing conversational utterances into simpler contexts improves data labeling efficiency and modularity. Finally, we present context tuning, a technique to finetune LLMs to better handle input contexts.

To my family.

Contents

Contents	ii
List of Figures	iv
List of Tables	vi
1 Introduction	1
2 Context Decomposition	3
2.1 Introduction	3
2.2 Decomposing Queries	4
2.3 Datasets	7
2.4 Experiments	10
2.5 Error Analysis	13
2.6 Related Work	13
2.7 Limitations	14
2.8 Interactive Retrieval	15
2.9 Prompts	15
2.10 Acknowledgements	17
2.11 Conclusion	17
3 Contextual Rewriting	23
3.1 Introduction	23
3.2 Background: LLM-Based Parsing	24
3.3 Few-Shot Adaptation	25
3.4 Experiments	27
3.5 Limitations	30
3.6 CFG for Constrained Decoding	31
3.7 Dataset Construction and Examples	31
3.8 Fine-tuning Experiment Hyperparameters	32
3.9 Joint Results	33
3.10 Rewriter Implementation	33

3.11 Conclusion	34
4 Context Tuning	41
4.1 Introduction	41
4.2 Context Tuning	43
4.3 Experiments	45
4.4 Limitations	49
4.5 Results and Discussion	49
4.6 Related Work	52
4.7 Conclusion	53
5 Conclusion	54
5.1 Additional Projects	54
5.2 Future Work	55
Bibliography	57

List of Figures

2.1	To resolve tip-of-the-tongue queries about books, our approach decomposes long, complex queries into subqueries routed to specific retrieval experts, each trained to handle a particular aspect of a book.	4
2.2	Recall at k of the top performing baseline (Contriever) and our decomposition baseline for vary amounts of negatives documents added to the corpus.	12
3.1	Four canonical paradigms of conversational semantic parsing for contextual utterances.	24
4.1	Retriever recall and model performance as a function of the number of retrieved documents. Model performance saturates long before retriever recall, indicating that the models have difficulty making use of the extra retrieved documents. Figure originally published in [38].	42
4.2	Document sampling and construction process for generating examples from Context Tuning.	43
4.3	Overview of the unsupervised phase of Context Tuning. First, positive documents, Document 1 (green) is sampled, along with negative documents, Documents 2, and Document 3, (red). The LM is then trained to first recite the parts of Document 1 that are in the context before generating the completion of the document.	45
4.4	Overview of the supervised phase of Context Tuning. In standard supervised finetuning (left), given document question and answer triples, the model is trained to generate the answer given the document and question (left). IN the supervised phase of Context Tuning we introduce negative documents (red) in the input context, and train the model to first generate the supporting document before the answer.	46
4.5	Example of running Context Tuning to generate data examples on the Wikipedia. Text before the cyan text corresponds to the segments of the positive document that are part of the document that is reconstructed. In the output, the blue corresponds to text that is present in the input context. The green corresponds to the tokens that complete the document. (part 1 of 2)	47

4.6	Example of running Context Tuning to generate data examples on the Wikipedia. Text before the cyan text corresponds to the segments of the positive document that are part of the document that is reconstructed. In the output, the blue corresponds to text that is present in the input context. The green corresponds to the tokens that complete the document. (part 2 of 2)	48
4.7	Results comparing Llama 2 Chat fine tuned directly on supervised data with first using Context Tuning for 10 total documents on the NaturalQuestions task with random Wikipedia distractors. Context Tuning improves results over directly fine-tuning on supervised data.	49
4.8	Results comparing Llama 2 Chat fine tuned directly on supervised data with first using Context Tuning for 10 total documents on the NaturalQuestions task with contriever Wikipedia distractors. Context Tuning improves results over directly fine-tuning on supervised data.	50
4.9	Results on the development set of HotPotQA comparing directly finetuning LLama 2-Chat on examples of HotPotQA (red) with first applying Context Tuning on the subset of data where two contexts are relevant according to the original HotpotQA annotations. Index 1 and index 2 correspond to the position of supporting contexts.	51

List of Tables

2.1	Tip of the tongue (TOT) queries are significantly longer while also having less lexical overlap with the gold document, compared with queries in popular retrieval datasets. Query length is number of BPE [63] pieces, averaged across examples. Lexical overlap is fraction of whole words in query that occur in gold passage(s), averaged across examples.	5
2.2	Query-document pairs, their generated sub-queries or <i>clues</i> , and corresponding gold document fields. Clues can be <i>extracted</i> directly from the query or <i>predicted</i> as a best-guess attempt to match the actual document field. Clues can be about the book’s title, author, date, cover, genre, or general plot elements. (part 1 of 3)	8
2.3	Query-document pairs, their generated sub-queries or <i>clues</i> , and corresponding gold document fields. Clues can be <i>extracted</i> directly from the query or <i>predicted</i> as a best-guess attempt to match the actual document field. Clues can be about the book’s title, author, date, cover, genre, or general plot elements. (part 2 of 3)	9
2.4	Query-document pairs, their generated sub-queries or <i>clues</i> , and corresponding gold document fields. Clues can be <i>extracted</i> directly from the query or <i>predicted</i> as a best-guess attempt to match the actual document field. Clues can be about the book’s title, author, date, cover, genre, or general plot elements. (part 3 of 3)	10
2.5	Number of queries organized by results from running query decomposition with extractive prompting. Rows correspond to number of clues Extracted (between one and six). For instance, top row is queries with only a single extracted clue and bottom row is queries with all clue types found. Num Posts counts number of queries in each bucket. Cell counts correspond to number of times a given clue type was extracted in a post, and percentages normalize against Num Posts . For example, 61.9% of queries with two extracted clues have one “Date” subquery. . .	18
2.6	Results on test set of . (Top) Baselines operate directly over queries and descriptions of the books. (Middle) We use the top performing model, Contriever, on the concatenated representations of subqueries (E-extractive, P-predictive, Q-rewritten queries) or the hypothetical document (H). (Bottom) Subqueries routed to individual retriever experts.	19
2.7	Results of individual retrieval experts on the test set of <i>WhatsThatBook</i> and <i>Reddit-TOMT (Books)</i> . E and P indicate extractive and predictive prompts. Extractive prompts scores are calculated only over the “N/A” queries.	19

2.8	Results of individual retrieval experts on the test set of <i>Reddit-TOMT (Books)</i> . E and P indicate extractive and predictive prompts. Extractive prompts scores are calculated only over the “N/A” queries.	20
2.9	Error types from the top performing model. Book representation are simplified here without all metadata for space constraints. (part 1 of 2)	21
2.10	Error types from the top performing model. Book representation are simplified here without all metadata for space constraints. (part 2 of 2)	22
3.1	Exact match accuracy on <i>SMCalFlow-EQ</i> test set. For both ICL and FT, we test each paradigm against the corresponding Parse-with-Utterance-History predictions using McNemar’s test and show statistically significant ($p < 0.05$) results with *.	28
3.2	Error examples with gold parses. (part 1 of 2)	29
3.3	Error examples with gold parses. (part 2 of 2)	30
3.4	Exact match accuracy on <i>SMCalFlow-EQ</i> test set combined with non-contextual utterances. For both ICL and FT, we test each paradigm against the corresponding Parse-with-Utterance-History predictions using McNemar’s test and show statistically significant ($p < 0.05$) results indicated with *.	31
3.5	Exact match accuracy on <i>SMCalFlow-EQ</i> test set combined with non-contextual utterances. For both ICL and FT, we test each paradigm against the corresponding Parse-with-Utterance-History predictions using McNemar’s test and show statistically significant ($p < 0.05$) results indicated with *.	33
3.6	Examples of temporal query revisions and their corresponding programs.	35
3.7	Example of subject-based query revision and its corresponding programs.	36
3.8	List of sub-tree transformations for simplifying <i>SMCalFlow</i> programs (part 1).	37
3.9	List of sub-tree transformations for simplifying <i>SMCalFlow</i> programs (part 2).	38
3.10	List of sub-tree transformations for simplifying <i>SMCalFlow</i> programs (part 3).	39
3.11	List of sub-tree transformations for simplifying <i>SMCalFlow</i> programs (part 4).	40
4.1	Model performance comparing only tuning on supervised data on total 10 documents in the NaturalQuestions task. Context Tuning corresponds to our full approach of first training on unsupervised examples followed by supervised examples.	50
4.2	Model performance comparing only tuning on supervised data on total 10 documents in the NaturalQuestions task. Context Tuning corresponds to our full approach of first training on unsupervised examples followed by supervised examples.	51
4.3	Model performance comparing only tuning on supervised data on total 10 documents in the HotPotQA task. Context Tuning (CT) corresponds to our full approach of first training on unsupervised examples followed by supervised examples.	52
4.4	Model performance comparing only tuning on supervised data on total 20 documents in the HotPotQA task. Context Tuning (CT) corresponds to our full approach of first training on unsupervised examples followed by supervised examples.	52

Acknowledgments

I am extremely fortunate to have two amazing advisors, Joey Gonzalez and Dan Klein. Among the many lessons I have learned under their guidance, I am particularly grateful for Joey's unlimited range of research interests and Dan's uncanny ability to identify the highest order bits. I would also like to thank Alane Suhr and Mohit Iyyer who have graciously guided me through the critical milestones of my PhD.

Being part of the research community at Berkeley has given me many opportunities to work with amazing researchers. I was fortunate to collaborate with Daniel Fried, Mitchell Stern, Nikita Kitaev, Kevin Yang, Eric Wallace, Eve Fleisig, Jessy Lin, Zhuohan Li, Sheng Shen, Sarah Wooders, Simon Mo, Charles Packer, Shishir Patil, and Tianjun Zhang. I'm grateful to Yizhou Chi and Yike Wang's patience as I grew as a research mentor. Thank you to all the other members of the Berkeley NLP and RISE / Sky research groups for the discussions that have shaped my perspectives on research and amazing staff who have tirelessly supported the research community here.

Outside of Berkeley, I'm thankful for many people who have been generous with their time and energy in helping me navigate research. Kyle Lo at AI2 has been a amazing collaborator and Hao Fang was a fantastic host during my time at Microsoft. I am also immensely grateful to Eugene Wu for introducing me research and Matt Gardner for introducing me to NLP.

My PhD would not be possible without the support of several exceptional individuals. To Cathy Chen, thank you for teaching me more about the human brain and Berkeley trails than I thought I would ever learn. To Nelson Liu, thank you for literally being there on day one of my research career and for every discussion we've had since. To my sister Eunice Lin, thank you for checking in on me and reminding me of what matters. To my parents, thank you for all your love and sacrifices you have made for me along the way. Last but not least, Shiori, I am unbelievably lucky to have a partner who is always there for me through all the ups and downs. Thank you for your kindness, insight, and humor.

Chapter 1

Introduction

Large language models (LLMs) are as capable as the context that they are given. Given a task, how well LLMs perform on it is determined by how the task is represented in the input context of the LLM and how well the LLM is trained to handle the chosen representation. In recent years, the predominant paradigm to of prompting pretrained LLMs as text-in and text-out machines has been shown to be widely successful across a wide range of tasks [49]. Yet, for domains where there is a need to leverage specialized models or overcome scarcity of data, LLMs alone remain not immediately suitable.

Taking this view, of LLMs as critical compute substrates of larger intelligent systems, this thesis contributes to the abstractions and computational patterns for structuring contexts for using LLMs as components of such systems. We show that techniques for structuring context of language models leads to systems capable of more complex tasks and modularity that enables better use of specialized tools and models. First, we highlight two different computational patterns for structuring context: context decomposition and context rewriting. We then present context tuning, a technique for finetuning LLMs to better handle structured contexts. Taken together, structuring contexts improves results across a range of tasks, information retrieval, semantic parsing and question answering.

Context Decomposition

In chapter 2, we present an information retrieval system and show how we can use context decomposition to tackle complex queries. In particular, we ground the problem in a complex information retrieval setting known as tip-of-the-tongue. In this setting, the user is unable to articulate a query that identifies the item that they are seeking. In such settings users generate long form queries with a variety of information facets that current models handle poorly out of the box. Using the LLMs to decompose complex queries and generate contexts for models specialized for different information facets, we up to 6% gain over vanilla state-of-the-art models for Recall@5.

Context Rewriting

In chapter 3, we show how contextual rewriting, using LLMs to rewrite complex contexts improves both data labeling efficiency and modularity in conversational semantic parsing. In interactive settings, utterances are often only understood with previous context. We explore several paradigms proposed in the literature for providing context to LLMs for this task. Along the way, we construct a dataset from SMCaFlow [62] with the annotations that allow fair comparisons between these major paradigms.

Context Tuning

In chapter 4, we propose a simple technique for fine-tuning language models to better use their contexts. As the context windows of LLMs are scaled up, it is appealing to structure context less to LLMs and rely more on the ability of language models to process many tokens. We observe the objectives and datasets that LLMs are pretrained on often fail to yield LLMs that effectively use the entire context. We design data-oriented approach that places relevant context across the context window and trains LLMs to first construct the context before generating the completion of the document. Across two document question answering tasks, we show up to 12% gain in accuracy when fine-tuning LLMs to better context.

The rest of this thesis is outlined as follows. In chapter 2, we explore how contexts can be decomposed. We take a challenging problem and use a language model to break down the problem into small contexts. In chapter 3, we explore how contexts can be rewritten. This is another important pattern, with contexts becoming longer, it at some point needs to be compressed down. We show that representation with language models themselves of the information can improve things. In chapter 4, we explore how to improve how the language models itself uses the context.

Chapter 2

Context Decomposition

In this chapter we present context decomposition, a technique for taking complex queries and decomposing them into simpler queries by an LLM that specialized retrievers better handle. We study the information retrieval setting known as tip-of-the-tongue, where users resort to long, queries that a variety of information facets because they are unable to articulate precise queries with identifiers. We construct a dataset, *WhatsThatBook*, a dataset of tip-of-the-tongue queries and show that on this dataset, decomposing context yields up to 6% gain in Recall@5 compared to state-of-the-art retrievers without using context decompositions.

2.1 Introduction

Tip of the tongue (TOT) refers to the retrieval setting in which a user is unable to formulate a precise query that identifies a sought item, even if the user knows they’ve encountered this item before. For example, users searching for movies they watched or books they read long ago often resort to complex and creative queries that employ a diverse set of strategies to express information relevant to the sought item—high-level categories (e.g., topic, genre), content details from the movie or book (e.g., events, characters), references to personal context (e.g., when they last read the book), descriptions of extratextual elements (e.g., movie promotional posters, book covers), and more. In fact, in an annotation study of TOT queries for movies, [2] found over 30 types of informational facets that users may include when crafting queries. Figure 2.1 shows a TOT query and its corresponding gold book.

A key challenge in TOT retrieval is that queries are not just longer and more complex than those in popular retrieval datasets, but resolving them requires an enriched document collection since query-document relevance can’t necessarily be established from document content alone (see Table 2.1). For example, in Figure 2.1, the query’s description of the book cover—*The cover of this book was yellow with a silhouette of a cat*—can be highly useful for identifying the book, but necessitates the book’s representation to contain that information.

In this work, we present a simple yet effective technique for improving TOT retrieval: First, we decompose queries into individual subqueries or *clues* that each capture a single

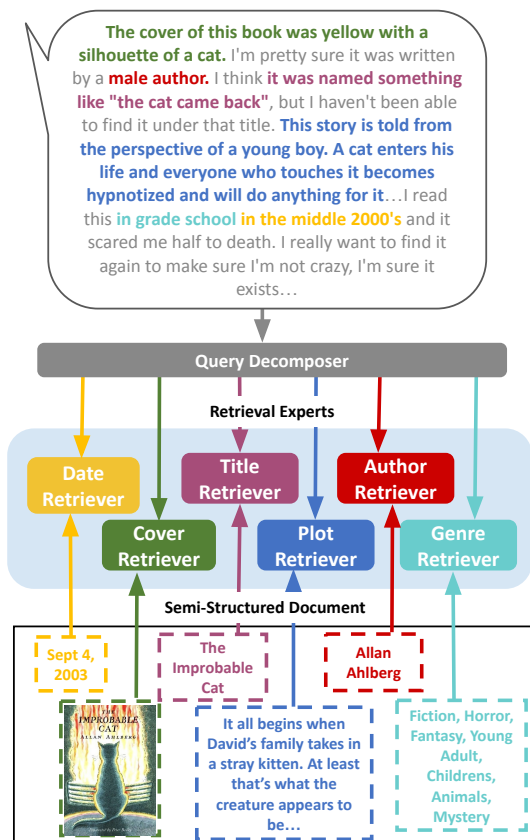


Figure 2.1: To resolve tip-of-the-tongue queries about books, our approach decomposes long, complex queries into subqueries routed to specific retrieval experts, each trained to handle a particular aspect of a book.

aspect of the target document. Then, we route these subqueries to expert retrievers that can be trained individually. Finally, we combine their results with those from a base retriever that receives the original query. Experiments show improvement in gold book recall over description-only retrieval baselines on a set of 14,441 real-world query-book pairs collected from an online forum for resolving TOT inquiries, complete with cover images and metadata.

¹

2.2 Decomposing Queries

In this section, we describe our a simple but effective method for tackling long, complex TOMT queries. Given a collection of documents d_1, \dots, d_n and a textual query q , the TOT retrieval

¹An earlier version of this chapter appeared in [37]

Dataset	Query Length	Lexical Overlap
MSMarco [6]	7.68	0.55
Natural Questions [31]	10.35	0.52
BioASQ [73]	14.82	0.58
TREC-COVID [54]	15.94	0.41
SciFact [75]	19.52	0.50
HotPotQA [82]	22.78	0.45
TOMT [4]	136.50	0.25
<i>WhatsThatBook</i>	156.20	0.19

Table 2.1: Tip of the tongue (TOT) queries are significantly longer while also having less lexical overlap with the gold document, compared with queries in popular retrieval datasets. Query length is number of BPE [63] pieces, averaged across examples. Lexical overlap is fraction of whole words in query that occur in gold passage(s), averaged across examples.

task aims to identify the sought document d^* . The input (raw) documents are semi-structured; each document d contains metadata fields $d^{(1)}, \dots, d^{(k)}$. In our the *WhatsThatBook* dataset, the documents are books that have the fields that correspond to plot, its publication year, an image of its book cover, title, genre, and author etc. Missing elements take on a default value (e.g., blank image, earliest publish date in overall book collection).

Method

First, the query decomposer takes a query q and outputs a set of subqueries $q^{(1)}, \dots, q^{(k)}$, for k metadata fields. To do this, we use in-context learning with a large language model (LLM) to generate a subquery q that is relevant to that field or optionally output the string "N/A" if the q does not contain any relevant information to the field; this is repeated for each field and can be run in independently in parallel for each field.

We experiment with two prompting strategies, *extractive* and *predictive*. Extractive prompting aims to generate subqueries that are purely extractions from the original query. The LLM is instructed to output "N/A" if there is not information relevant to the metadata field.

Predictive prompting aims to generate subqueries that are similar to the *content* of the metadata field. Note that the subqueries need not be extractions from the original query and can be inferred metadata or even hallucinations from the query decomposer. Using LLMs to generate subqueries affords us the ability to set the few-shot prompt generation targets to be *predictions*. This is important as the information in queries are rarely presented in a form amenable for matching with the corresponding document field. For example, books have publish dates, but queries will rarely mention these dates; instead, users may articulate personal context (e.g., "*I read this book in highschool around 2002-2005*"). Then to simplify

the learning task for a date-focused retrieval expert, we might ask the LLM to predict a “latest possible publish date” (e.g., 2005).

See Table 2.4 for examples of extractive and predictive subqueries and 2.9 for examples of their prompts. In practice, we use GPT 3.5 (`gpt-3.5-turbo`) few-shot prompting with up to 8 in-context examples. Each field has its own prompt template and set of examples. Subqueries for different fields can be generated in parallel, as they are independent of each other.

Retrieval Experts

We have retriever models, or experts, that specialize to specific field types. Let R_1, \dots, R_k represent these retrievers. Retrievers can be implemented as dense, sparse, or symbolic logic.

If a retriever requires training, we run the query decomposer over all query-document pairs (q, d) in the training set. This produces effectively k training datasets, where each dataset is comprised of a subquery and document-field pair. For example, field j would have training dataset of examples $(q^{(j)}, d^{(j)})$.

At indexing time, each document’s field is indexed according to the specifications of its retriever expert. For example, if the retriever is implemented as an embedding model, then that specific field is converted into an embedding. On the other hand, if the retriever is a sparse model, then a sparse index would be built using just that specific field’s text.

At inference time, each retriever takes a subquery $q^{(j)}$ and retrieves documents from its associated index of fields.

Implementation details

In practice, for titles, authors, plot, and genre, we use Contriever [22], a state-of-the-art dense retriever.² For both models, we train for a total of 10,000 steps with a batch size of 16, learning rate of 1e-4. For titles, we finetune with 3,327 extracted subqueries. For our base retriever, we use the full training set of original book descriptions. For embedding search during inference, we use the `faiss` library and project all embeddings to 768 [25].

For cover images, we use CLIP [50], a state-of-the-art image-text model that can be used for image retrieval by scoring matches between embedded images and their textual descriptions. Specifically, we finetune ViT-B/32³ on 2,220 extracted subqueries using cross-entropy loss with batch size of 4, learning rate of 5e-5 and weight decay of 0.2 for 10 epochs with the Adam optimizer [30]. We select the model with the best top 1 retrieval accuracy on a validation set.

For publish dates using the predictive prompting, we use a symbolic function that heuristically scores 0 if a book was published after the subquery date (i.e. predicted latest publish date) and 1 otherwise. If necessary, we heuristically resolve the subquery to a year.

²<https://huggingface.co/facebook/contriever>

³<https://huggingface.co/sentence-transformers/clip-ViT-B-32>

Combining retrieved results

In this work, we restrict to a simple strategy of using a weighted sum of all k retrieval scores across the $(q^{(j)}, d^{(j)})$. That is, the final score is:

$$s(q, d) = \sum_{j=1}^n w^{(j)} R_j(q^{(j)}, d^{(j)})$$

All documents are scored in this manner, which induces a document ranking for a given query q . We tune the weights w_j on the validation set and select the weights that have the best Recall@5.

2.3 Datasets

WhatsThatBook We introduce the *WhatsThatBook* dataset consisting of query-book pairs collected from a public online forum on GoodReads for resolving TOT inquiries about books.⁴ On this forum, users post inquiries describing their sought book and community members reply with links to books on GoodReads as proposed answers.⁵ If the searcher accepts a book as the correct answer, the post is manually tagged as SOLVED and a link to the found book is pinned to the thread. For these solved threads, we take the original inquiry as our query q and the linked book as gold d^* . At the end, *WhatsThatBook* contains 14,441 query-book pairs. Each query corresponds to a unique book. Finally, these books are associated with pages on GoodReads, which we used to obtain publication year metadata and images of book covers. We further collect 172,422 negative books (books that do not correspond to a gold query in our dataset) to make the setting more realistic, for a total of 186,863 books. To collect negative books, crawl books authored by crawling books by the authors of the positive books.

Understanding *WhatsThatBook* queries. To examine the information contained within queries, we analyze the results of query decomposition using extractive prompting (see Table 2.5). First, we find only 645 (4.5%) posts have no clue extracted (and thus aren't captured in the table total). Second, most posts have between one and three clues ($\mu=2.33$). Third, nearly every query contains some description of the book's plot elements. Beyond that, over half of the queries provide some description of temporal context surrounding the book. Queries containing descriptions of the author are rare, which is expected since knowing the author name would likely have allowed the user to find the book without seeking assistance. Given that, it's somewhat surprising that descriptions of titles occur almost a third of the time. Manually examining these, we find title clues are often uncertain guesses ("*I think the*

⁴<https://www.goodreads.com/group/show/185-what-s-the-name-of-that-book>. We scraped data from February 2022.

⁵This is a simplification of community interactions. Threads also may include dialogue between original poster and members but this is beyond the scope of our work.


Query-Document	Clue-Field	Clue-Field	Clue-Field
<p><i>Query:</i> I think I saw this in a used store once and I remember saying to my new husband “my daughter use to read that book to her little brother” and it’s funny because on the outside cover is a little girl reading a book to her little brother. It’s called.....my book, or my story, or something simple like that. It would be about 15 or more years old. The girl was blond and the boy brunet....I think!!!! Inside was the cutest little sentences and my kids use to do what each page said...thank you!!</p>	<p><i>Extracted Title:</i> It’s called.....my book, or my story, or something simple like that.</p>	<p><i>Extracted Clue:</i> 15 years old (2006 or earlier)</p>	<p><i>Extracted Cover:</i> The outside cover is a little girl reading a book to her little brother.</p>
<p><i>Description:</i> Glossy pictorial hardcover no dust jacket. 2001 7.75x9.13x25. <i>GUIDE FOR PARENTS WITH PICTURES, HOW TO TEACHING CHILDREN READING.</i></p>	<p><i>Actual Title:</i> My First Book</p>	<p><i>Actual Date First published:</i> September 1, 1984</p>	<p><i>Actual Cover:</i></p> 

Table 2.2: Query-document pairs, their generated sub-queries or *clues*, and corresponding gold document fields. Clues can be *extracted* directly from the query or *predicted* as a best-guess attempt to match the actual document field. Clues can be about the book’s title, author, date, cover, genre, or general plot elements. (part 1 of 3)

title might start with Nurse but I’m not sure) or details that models are likely to struggle with (“*The title is made up of either two or four short words (maybe one syllable)*”).

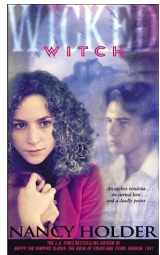
Query-Document	Clue-Field	Clue-Field	Clue-Field
<p><i>Query:</i> I read a book in 2008 or 2009 that was part of a series for young adults. It was fantasy, and about several families of witches and warlockes. The main character was named Holly.</p>	<p><i>Extracted Genre:</i> for young adults, fantasy</p>	<p><i>Predicted Plot:</i> Holly is a young witch who comes from a long line of magical families...</p>	<p><i>Predicted Cover:</i> Young woman with long, curly hair holding a wand and surrounded by swirling colors and symbols</p> <p><i>Actual Cover:</i></p> 
<p><i>Description:</i> Holly Cathers’s world shatters when her parents are killed in a rafting accident. She is wrenched from her home in San Francisco and sent to Seattle to live with her aunt, Marie-Claire, and her twin cousins, Amanda and Nicole...</p>	<p><i>Actual Genre:</i> Fantasy, Young Adult, Paranormal, Romance, Fiction</p>	<p><i>Actual Plot:</i> (see description)</p>	

Table 2.3: Query-document pairs, their generated sub-queries or *clues*, and corresponding gold document fields. Clues can be *extracted* directly from the query or *predicted* as a best-guess attempt to match the actual document field. Clues can be about the book’s title, author, date, cover, genre, or general plot elements. (part 2 of 3)

Reddit-TOMT (Books) We additionally use the books subset⁶ of the *Reddit-TOMT* [4] dataset, which includes 2,272 query-book pairs.⁷ Queries can refer to the same book, leaving 1,877 unique books in this dataset.

Experimental setup. For the experiments in the rest of this paper, we split *WhatsThatBook* into train ($n=11,552$), validation ($n=1,444$) and test ($n=1,445$) sets. By the nature of our dataset construction, the number of queries and books is equal. We use all 14,441 books,

⁶We note that the full *Reddit-TOMT* dataset also contains 13K queries matched with movies. To fully explore the capabilities of our approach, we restricted to the books subset specifically due to feasibility of obtaining images of book covers, while doing the same with movie posters was more difficult. We leave this to future investigations.

⁷We removed 47 query-book pairs for which the gold book did not have a GoodReads link, which was necessary for obtaining cover images.

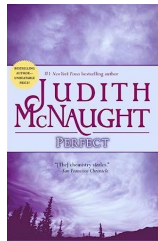
Query-Document	Clue-Field	Clue-Field	Clue-Field
<p><i>Query:</i> Written in the 1990s, by [probably] one of those romance novelists (not Nora Roberts though). The cover was just a snowed-in cabin on a mountain side at night. In a nut shell - An escaped prisoner takes a woman captive and steals her car with her in it!...Any ideas what this might be called? I haven't forgotten it after all these years.</p>	<p><i>Extracted Author:</i> by [probably] one of those romance novelists (not Nora Roberts though)</p>	<p><i>Predicted Cover:</i> Dark, mysterious cabin obscured by falling snow with the silhouette...</p>	<p><i>Extracted Cover:</i> The cover was just a snowed-in cabin on a mountain side at night.</p>
<p><i>Description:</i> A rootless foster child, Julie Mathison had blossomed under the love showered upon her by her adoptive family. Now a lovely and vivacious young woman...</p>	<p><i>Actual Author:</i> Judith McNaught</p>	<p><i>Actual Cover:</i> (see right)</p>	<p><i>Actual Cover:</i></p> 

Table 2.4: Query-document pairs, their generated sub-queries or *clues*, and corresponding gold document fields. Clues can be *extracted* directly from the query or *predicted* as a best-guess attempt to match the actual document field. Clues can be about the book’s title, author, date, cover, genre, or general plot elements. (part 3 of 3)

which are gold targets with respect to some query, as our full document collection for indexing. As for *Reddit-TOMT (Books)*, given its size we use the entire dataset as an additional test set.

2.4 Experiments

Baseline models

Models that use original queries. We evaluate our approach against several popular retrieval models that have been used as baselines for a range of other retrieval datasets (see Table 2.1). For text-only models—BM25 [55, 57], Dense Passage Retrieval (DPR) [26], Contriever [22], and ColBERT [27]—the document representation is simply the concatenation

of all available document fields into a single text field. For our image-only baseline—CLIP [50]—the document representation is only the embedded book cover. All baselines receive the same input (full) query, and are finetuned with the same hyperparameters (§2.2) except using the full training set (as opposed to just examples with a successful subquery extraction).

Models that use generated queries. To evaluate the effect of generating subqueries independently without training specialized retrievers, we also train the top performing text-only model from the set of models with queries enriched with LLMs. We experiment with using the concatenation of all subqueries generated by the query decomposer as the query representation for both the extractive and predicted subqueries. To isolate the effect of generating separate subqueries, we also use LLM to generate a *single* rewritten query that is used as input to Contriever, by simply prompting LLMs to generate a single new query that better retrieves the desired item (prompt in 2.9). In addition, we generate *hypothetical documents* and use these as the query, similar to [15] though we differ in that we train on these generations as queries while [15] restricted their use to the zero-shot setting.

Results

Table 3.1 shows the test results on *WhatsThatBook*. We use Recall@ K metric as our primary metric since each query has exactly one correct item.

Contriever is the best-performing baseline model. In this setting with low lexical overlap, we see that dense retrievers like DPR and Contriever outperform sparse retrievers like BM25. Without extracting clues about the book cover, using CLIP on its own is not effective, likely due to its limited context window.⁸

Query decomposition improves performance. When using the rewritten queries without training individual expert retrievers, we observe that concatenating the predicted clues improve the model. However, both generating rewritten queries and entire hypothetical documents perform worse than just using the original document as input. Our approach to decompose queries and route clues to specialized retrievers improves performance (ranging from +6 to +7 Recall@ K across all cutoffs) over the next best baseline retriever. Table 2.8 shows the results for each individual expert retriever on *WhatsThatBook* and the books subset of TOMT.

Performance degrades predictably with corpus size To see how well the performance scales as more documents are added, we also report the performance of the base performing baseline and our model as we add more documents to the corpus. We evaluate the performance

⁸We pass the full query into CLIP and allow for truncation to happen naturally. This is a big issue with CLIP, which supports a narrow query length; hence, motivating our approach to extract *clues* about book covers from the full query.

for different corpus sizes by varying the number of negative books added to the corpus. Figure 2.2 shows that for both our decomposition and the baseline, the performance drops significantly as more negative books are added. There is a sharp decline of performance when negative books are first introduced (between corpus size 1 and 2). Since the negative books are collected from the authors of the positive books, they may be more difficult to distinguish than a random book. As more documents are added, there is a marginal decrease in performance smoothly. Furthermore, our decomposition method performance better than the baseline for different corpus sizes and has slightly less performance degradation as more documents are added to the corpus.

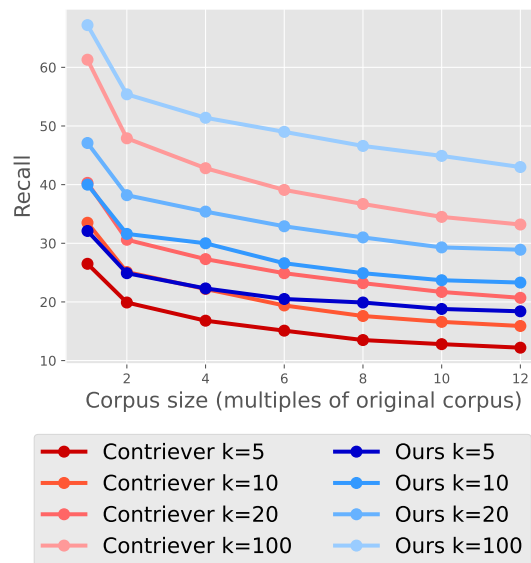


Figure 2.2: Recall at k of the top performing baseline (Contriever) and our decomposition baseline for vary amounts of negatives documents added to the corpus.

Metadata fields exhibit long-tailed behavior. The query decomposer generates a plot subquery in at least 90% of the queries for both *WhatsThatBook* and TOMT. Dates occur in a large proportion of the queries, but are not specific enough to be effective identifying the book. While subqueries for the author appear very infrequently, when they do appear, they are much more effective than more generic subqueries such as dates or genres. Images are much more effective when using decomposition compared to the image only CLIP baseline, as the image retriever model is able to access the visual part of the text

Predictive prompting performs better than extractive prompting. Overall, we find that predicting subqueries is more effective than extracting. Subqueries are generated

for every query, thus, there is more data for the specialized retriever experts to train on, compared to extractive clues where some retrievers only have a small fraction of the dataset for finetuning. Moreover, the predictive clues allows the LLM to make inferences based on information that is not explicitly present in the query. For example, the LLM is able to make inferences about the genre effectively based on the full query even when the user does not explicitly discuss the genre. Another benefit of the extractive clue is that the subqueries are more grounded in the original query.

Trade-off exists between generating subquery and individual retriever expert performance. Between the prompting strategies, we find that there is often a trade-off between how selective the query decomposer is with generating a subquery for a metadata field, and the effectiveness when generated. For most of the metadata retrieval queries, the extractive prompting approach is slightly more effective than predictive prompting on the examples that it does not predict "N/A" on.

2.5 Error Analysis

We sample 50 predictions where the top performing model where the model fails to get the book correct in the top 5 and categorize the types of errors made by the model in Table 2.10. The most common kind of error is failure to model the relationship between the document and query, which happens in instances where there may be events that dense models fail to capture indicating that there is still headroom on the task. Moreover, documents are sometimes brief or written in a style that is not similar to other descriptions. Lastly, because users are recalling memories, some of these can be *false memories* that lead to errors in retrieval.

2.6 Related Work

Dense methods for document retrieval. Document retrieval has a long history of study in fields like machine learning, information retrieval, natural language processing, library and information sciences, and others. Recent years has seen the rise in adoption of dense, neural network-based methods, such as DPR [26] and Contriever [23], which have been shown can outperform sparse methods like BM25 [55, 57] in retrieval settings in which query-document relevance cannot solely be determined by lexical overlap. Researchers have studied these models using large datasets of query-document pairs in web, Wikipedia, and scientific literature-scale retrieval settings [6, 61, 54]. Many retrieval datasets have adopted particular task formats such as question answering [31, 82, 73] or claim verification [72, 75]. We direct readers to [88] for a comprehensive, up-to-date survey of methods, tasks, and datasets.

Known-item and TOT retrieval. Tip of the tongue (TOT) is a form of known-item retrieval [5, 32], a long-studied area in the library and information sciences. Yet, lack of large-scale public datasets has made development of retrieval methods for this task difficult. Prior work on known-item retrieval focused on constructing synthetic datasets [3, 29, 14]. For example, [16] released a dataset of 2,755 query-item pairs from *Yahoo!* answers and injected query inaccuracies via hired annotators to simulate the phenomenon of *false memories* [17, 18], a common property of TOT settings.

The emergence of large, online communities for resolving TOT queries has enabled the curation of realistic datasets. [2] categorized the types of information referred to in TOT queries from the website *I Remember This Movie*.⁹ Most recently, [4] collected queries from the *Tip Of The Tongue* community on Reddit¹⁰ and evaluated BM25 and DPR baselines. Our work expands on their work in a key way: We introduce a new method for retrieval inspired by long, complex TOT queries. In order to test our method on a large dataset of TOT queries, we collected a new dataset of resolved TOT queries such that we also had access to metadata and book cover images, which were not part of [4]’s dataset.

Query Understanding and Decomposition. Our work on understanding complex information-seeking queries by decomposition is related to a line of work breaking down language tasks into modular subtasks [1]. More recently, LLMs have been used for decomposing complex tasks such as multi-hop questions into a sequence of simpler subtasks [28] or smaller language steps handled by simpler models [24].

Related to decomposition of long, complex queries for retrieval is literature on document similarity [46] or query-by-document (QBD) [81]. In these works, a common approach is decomposing documents into sub-passages (e.g. sentences) and performing retrieval on those textual units. The key differentiator between these works and ours is that document similarity or QBD are inherently symmetric retrieval operations, whereas our setting requires designing approaches to handle asymmetry in available information (and thus choice of modeling approach or representation) between queries and documents. In this vein, one can also draw parallels to [34], which demonstrates that retrieving over model-generated question-answering pairs instead of their originating documents can improve retrieval, likely due to improved query-document form alignment. In a way, this is similar to our use of LLMs to generate clues that better align with extratextual document fields, though our work is focused on query-side decomposition rather than document-side enrichment. More recently, [76] propose using LLMs for decomposing different facets of complex queries for scientific documents.

2.7 Limitations

Given our proposed method is designed to handle TOT queries, there is an implicit assumption that the document collection covers the sought item with high probability. That is, a system

⁹<https://irememberthismovie.com/>

¹⁰<https://www.reddit.com/r/tipofmytongue/>

tailored for TOT retrieval is unlikely to perform well if its index is missing too many books. While our dataset is large-scale, one limitation is that even a corpus of 187k documents does not cover the sought document sufficiently. Consider, for instance, the total number of books available on GoodReads (over 3.5B as of time of writing). Another limitation of our method is that the overhead for tackling another domain with this technique is non-trivial as the prompts and few-shot examples may not be directly transferable. We believe a promising avenue for future work is reducing the effort needed to bootstrap the design and training of retrieval experts and incorporating them into a query decomposer.

2.8 Interactive Retrieval

Due to the ambiguity of the queries, tip-of-the tongue retrieval is a natural setting for interactive retrieval. In [11], we studying the task of generating informative clarification queries. Where given an ambiguous search query, the system generates questions in a multi-turn dialogue to improve retrieval quality. The approach We present a system that asks informative clarification questions by choosing questions whose answers would maximize certainty in the correct candidate. We finetune LLMs to condition on a retrieval distribution, to generate the question that would hav emaximized the rank of the true candidate at each turn. On *WhatsThatBook*, our approach outperforms traditional heuristics such as maximizing information game by 17% and vanilla prompting LLMs by 39% relative.

2.9 Prompts

Extractive Prompt

```

You are a utility built to take in forum queries from users looking for books and output the aspect that is
  about the cover. If there is not enough information, then output N/A. Do not guess and only output
  text that was in the
query. Here are some examples:
Question: Hi there, I read this book in highschool around 2002-2005. From what I remember, the main
  character is nicknamed "Mouse" and she rides a big chestnut horse in jumper shows. I think this book
  may have been Australian. The cover just showed a chestnut horse and rider in mid jump. I think the
  title was one word--it may have been the name of the horse.I cannot remember the name or the author of
  this book. I have googled everything I can think of but I cannot to find this book and its driving me
  crazy! I'd be grateful for any help on this! Thank you! Cover Clue: The cover just showed a chestnut
  horse and rider in mid jump.
Question: So I read this book somewhere between 2008-2010 when I was in elementary school or middle school.
  It was about a girl who lived by the sea, and started out with her having dinner at this boarding
  house she lived in. And something happened (she received a fruit or something else that was banned in
  her town) and she had to hide it. Then, the landlady of her house got mad when she found out. The MC
  went down to the beach and saw this giant walrus thing (I can't remember clearly but I think there was
  also a guy who rode the walrus but maybe not) and the MC got on the walrus and they rode away to this
  magical land. I don't remember much else, except the story had something to do with fruits for some
  reason, and the characters mother was likely from this magical land. Also, in the end, the main
  character brings Color and happiness back to her seaside village and I think there's something else to
  do with strawberries. Please help! I've been searching for this book for years! Thanks so much <3
  Cover Clue: N/A
Question: I remember reading this around 2008/2009. It was about a girl being prettier than her mother, and
  when the mother gets jealous she sends her daughter to a type of boarding school for people who are
  well-known (Like nobles and princesses. She may be royalty). The cover was purple and
  captivating,(which was why I picked the book to read ^^) with a girl's face on it. However, I'm not

```

sure if the cover really had a face on it, or if it was completely purple.(view spoiler)[I believe that at the end, a minion of the girl's mother helped the girl get rid of her mother. Maybe it was because the mother tried to kill her daughter for being beautiful, especially since she was getting older. (hide spoiler)]Please help me find it. It was the first book that got me to love reading at that time. :) Cover Clue: The cover was purple and captivating,(which was why I picked the book to read ^^) with a girl's face on it. However, I'm not sure if the cover really had a face on it, or if it was completely purple.

Question: I probably read this book in the late 90s, early 2000s, and cannot find the title anywhere. The main character is an outcast in her high school who lives in a trailer park near a cemetery. From what I remember she was walking through the cemetery trying to clear her mind when she wondered upon a funeral that had very few people at the funeral. After that she starts attending funerals on the weekend. She even has an outfit she wears just to go to these funerals. Cover Clue: N/A

Question: hi im looking for a book there was two girls. the older one liked dumpster diving and had a duck shaped jar of memories that she could go into. the younger one was blonde and liked buying shirts from a thrift store and writing on them, and she couldnt remember anyones faces. the younger one tried to mail her mom a coconut and the mailman was the only face she could remember. there was also a woman in a hospital who only said red shoes and the cover was green or blue. was read a couple years ago so probably 2017-2019ish? Cover Clue: the cover was green or blue.

Question: This book was blue (if that helps). And I think it had a mailbox on the front of it. There were three girls and they hated their rival school. So to get the two schools to be friends they made everyone send letters to another person in the school. One girl got a guy who wanted to impress a girl at his school so him and the girl he was sending letters went on on "Practice Dates" And they ended up liking each other :) The other girl likes playing "Games" with her letter guy... i dont really remember what happened with them. And the third girl had a jerk as her letter guy.. I dont really remember what happened with him either.If anyone knows the name of the book I'm looking for would you please let me know? :) Thanks so much :) :) Cover Clue: This book was blue (if that helps). And I think it had a mailbox on the front of it.

Question: So I do not remember but the title, author, or cover, but I remember a bit of the plot. I believe it starts out with the main characters set up on a date. They dont know each others names, but their date consists of a sexy get to know you /photo shoot where theyre not allowed to have sex but they do anyway. She gets pregnant but cant find him after. Flash forward a bit, she works for her fathers large New York office and they make a deal with with her baby daddys office. He finds out shes pregnant and so the story goes. I hope this is enough to go on and someone recognizes it. Thank you! Cover Clue: N/A

Now here is the example, remember not to add in additional information that's not in the question.Question: I read this book back around 2008 (I think) can't remember author but read two of her works 1) is about this girl who falls in love with a guy who is actually a dragon, he takes her back to his homeland and I think he is injured, there are other dragons there and his relatives are also dragons author gives very vivid imagery I believe there were types of dragons like fire and ice ones... 2) the second book (warning:spoilers ahead) is about this girl who has a sister named rose (not too sure about name) who is already engaged to some local village boy, but she falls in love with this cold man who is like "ivy" and they escape together for a while but she returns and the girl also loves him but he eludes her grasp, I remember the end she talks about the imagery on the wall of roses and ivy intertwining together... Any help much appreciated!!! :) Cover Clue:

Predictive Prompts

You are a utility for guessing titles of books. Given the book described below, what is a possible title for the book? Only return one predicted title without any extra text. Even if you're unsure, try to come up with something.\n\nDescription: {}

You are a utility for guessing author names. Guess a possible author for the book described below. Don't worry if you're unsure. Only return the name and a short explanation of why.\n\nDescription: {}

You are a utility for categorizing books. Given the book description below, generate several possible genre tags for the book. Try to have a diversity of coarse and fine-grained genres. Don't generate more than five genres.\n\nDescription: {}

```
You are a useful tool for generating ideas for cover art. Write 1 or 2 sentences depicting what a the cover of a book might look like based on the description below. Stick to only one idea.\n\nDescription: {}
```

```
You are a writing tool for generating ideas. Write a possible plot synopsis for a book based on the description below. Don't include the title or author.\n\nDescription: {}
```

Query Rewrite Prompt

```
"You are a utility for helping users find books. Given the user's book description below, generate a query that a user can copy-paste into a book database to find the book. The query should focus on the important aspects of the book that will help the database locate it. These can be keywords about the book's title, author, genre, year, or distinguishing character or plot elements.\n\nUser's Query: {}\nDatabase Query: "
```

2.10 Acknowledgements

We thank the members of the Berkeley NLP group and the anonymous reviewers for their insightful feedback. This work was supported by a grant from DARPA SemaFor, and gifts from sponsors of Sky Computing and BAIR. The content does not necessarily reflect the position of the sponsors, and no official endorsement should be inferred.

2.11 Conclusion

We study tip of the tongue retrieval, a real-world information-seeking setting in which users issue long, complex queries for re-finding items despite being unable to articulate identifying details about those items. We introduce *WhatsThatBook*, a large challenging dataset of real-world TOT queries for books. We also introduce a simple but effective approach to handling these complex queries that decomposes them into subqueries that are routed to expert retrievers for specialized scoring. Our simple framework allows for modular composition of different retrievers and leveraging of pretrained models for specific modalities such as CLIP for document images. We experiment with different subquery generation strategies and find that generating predictions of document fields is more effective. We observe improvements up to 6% absolute gain over state-of-the-art dense retrievers for Recall@5 when incorporating query decomposition into existing retrievers.

Part 1: Extract, Plot, Dates, and Cover			
Extract	Plot	Dates	Cover
1	2657 (89.3%)	138 (4.6%)	43 (1.4%)
2	4630 (96.0%)	2985 (61.9%)	283 (5.9%)
3	3476 (97.6%)	2759 (77.4%)	856 (24.0%)
4	1821 (98.8%)	1627 (88.3%)	1092 (59.3%)
5	537 (99.4%)	511 (94.6%)	484 (89.6%)
6	50 (100%)	50 (100%)	50 (100%)
Total	13171 (95.5%)	8070 (58.5%)	2808 (20.4%)
Part 2: Title, Genre, Author, and Posts			
Extract	Title	Genre	Author
Num Posts			
1	80 (2.7%)	39 (1.3%)	18 (0.6%)
2975			
2	641 (13.3%)	994 (20.6%)	117 (2.4%)
4825			
3	1415 (39.7%)	1841 (51.7%)	342 (9.6%)
3563			
4	1458 (79.1%)	1064 (57.7%)	310 (16.8%)
1843			
5	502 (93.0%)	456 (84.4%)	210 (38.9%)
540			
6	50 (100%)	50 (100%)	50 (100%)
50			
Total	4146 (30.1%)	4444 (32.2%)	1047 (7.6%)
13796			

Table 2.5: Number of queries organized by results from running query decomposition with extractive prompting. Rows correspond to number of clues **Extracted** (between one and six). For instance, top row is queries with only a single extracted clue and bottom row is queries with all clue types found. **Num Posts** counts number of queries in each bucket. Cell counts correspond to number of times a given clue type was extracted in a post, and percentages normalize against **Num Posts**. For example, 61.9% of queries with two extracted clues have one “Date” subquery.

Model	<i>WhatsThatBook</i>			
	Top 5	Top 10	Top 20	Top 100
BM25	8.5	12.5	16.2	22.5
CLIP	1.9	2.8	3.5	5.7
DPR	23.1	31.2	38.1	57.6
ColBERT	17.8	18.3	25.4	34.1
Contriever	26.5	33.5	40.3	61.3
Contriever (E)	26.7	33.4	41.8	60.5
Contriever (P)	29.3	35.5	42.1	61.7
Contriever (H)	25.0	34.1	40.2	60.4
Contriever (Q)	18.5	24.2	31.8	52.9
Ours (E)	26.6	34.1	40.2	60.4
Ours (P)	32.1	40.0	47.1	67.2

Table 2.6: Results on test set of . (Top) Baselines operate directly over queries and descriptions of the books. (Middle) We use the top performing model, Contriever, on the concatenated representations of subqueries (E-extractive, P-predictive, Q-rewritten queries) or the hypothetical document (H). (Bottom) Subqueries routed to individual retriever experts.

Subquery	<i>WhatsThatBook</i>				Subquery
	Top 5	Top 10	Top 20	Top 100	
plot (E)	29.0	36.4	44.9	64.2	90.4
plot (P)	27.9	34.7	42.4	61.7	100
dates (E)	1.3	2.1	3.4	6.0	54.8
dates (P)	0	0	1.1	2.3	100
cover (E)	5.3	6.7	7.7	14.1	21
cover (P)	3.1	4.4	5.6	7.2	100
title (E)	11.7	12.9	14.6	19.8	29.0
title (P)	4.3	5.7	8.2	13.4	100
genre (E)	2.7	3.4	5.9	17.4	28.2
genre (P)	3.8	6.8	10.9	24.4	100
author (E)	6.0	8.0	8.0	9.1	6.9
author (P)	0	1	1.1	3.2	100

Table 2.7: Results of individual retrieval experts on the test set of *WhatsThatBook* and *Reddit-TOMT (Books)*. E and P indicate extractive and predictive prompts. Extractive prompts scores are calculated only over the “N/A” queries.

<i>Reddit-TOMT (Books)</i>					
Subquery	Top 5	Top 10	Top 20	Top 100	% not N/A
plot (E)	45.5	56.2	61.0	76.0	94.3
plot (P)	43.2	50.5	58.8	77.1	100
dates (E)	0.0	1.0	2.2	2.8	59.8
dates (P)	0.0	1.8	2.3	4.4	100
cover (E)	8.6	11.6	15.5	17.7	35.8
cover (P)	5.8	8.4	10.0	13.4	100
title (E)	13.4	17.5	22.2	37.4	43.9
title (P)	14.3	17.2	21.0	35.5	100
genre (E)	1.3	4.4	5.2	28.8	26.8
genre (P)	6.7	8.6	12.3	25.5	100
author (E)	7.1	8.2	9.6	9.6	9.9
author (P)	1.9	3.0	4.7	14.5	100

Table 2.8: Results of individual retrieval experts on the test set of *Reddit-TOMT (Books)*. E and P indicate extractive and predictive prompts. Extractive prompts scores are calculated only over the “N/A” queries.

%	Error type	Query	Document
54%	Fail to Query-Doc Relationship t	"I think this was a teen book. Don't remember Author or character names. All " I remember is that the girl loses her memory, it is not " Memoirs of a teenage amnesiac". It was fiction book and it wasn't comic or manga. I think the girl was involved in a car accident where someone hit her while she was walking? Apparently she was quite wild and broke instruments and was quite hated before she had the accident or something like that for things she done "to people, but she cant remember any of it. ...	<i>Title</i> Kat Got Your Tongue' <i>Plot</i> After a terrible car accident, Kat wakes up with no idea who she is, and no memory of anything before the crash. She "doesn't even recognize her mum, much less her friends from " school—Poppy, Jade, and the mysterious Tina. Only after she finds her old diary—written in a voice no longer her own—does Kat begin to discover the terrible secrets of her previous life. This incredib After a terrible car accident, ...

Table 2.9: Error types from the top performing model. Book representation are simplified here without all metadata for space constraints. (part 1 of 2)

%	Error type	Query	Document
28%	Document Representation Insufficient	I read this book many years ago and it was an older book so I do not know the publication year. What I remember is a mother with two daughters that were young ladies possibly and their late teens early twenties. The mother "favored the one daughter named Charlotte...	<i>Title</i> The Daughters of Ardmore Hall <i>Plot:</i> An unbalanced woman seeks to destroy her daughters.
18%	Query Contains Error	Fiction, read in the 70s by my mother who thinks it was probably written in that time period too. Book is about a dying woman remembering her childhood during a war in (probably South) Africa. Title might have something to do with a dragon or a mosquito coil :)	<i>Title:</i> Moon Tiger <i>Plot:</i> The elderly Claudia Hampton, a best-selling author of popular history; lies alone in a London hospital bed. Memories of her life still glow in her fading consciousness, but she imagines writing a history

Table 2.10: Error types from the top performing model. Book representation are simplified here without all metadata for space constraints. (part 2 of 2)

Chapter 3

Contextual Rewriting

In the previous chapter, presented context decomposition, a method for structuring contexts by breaking down complex contexts into simpler components that are better handled by specialized models. In this chapter, we present context rewriting, a different way of structuring contexts, in conversational semantic parsing settings. In conversational semantic parsing settings, contexts often are complex because they refer to and make edits to previous events, people, and places in the conversation history. How such previous contexts is presented the LLM for semantic parsing is crucial to how well the overall system performs and how well it can be maintained. Along the way, we construct a dataset from SMCaFlow [62] with the annotations that allow fair comparisons between these major paradigms. On this dataset, we build representative systems of the major paradigms in conversational semantic parsing and show that using LLMs to rewrite context maintains performance while significantly lowering the need for expert data annotations.

3.1 Introduction

A key challenge in conversational semantic parsing (CSP) is handling *contextual* utterances (*i.e.*, utterances that can only be understood with its context) by mapping them to *non-contextual* programs that can be fulfilled by an executor without relying on the dialogue state. Many approaches have been proposed, *e.g.*, directly mapping the contextual utterance with utterance history to a non-contextual program [70], or mapping to an intermediate contextual program which is then resolved (usually in a deterministic manner) to a non-contextual program [62, 10]. In these prior works, there is often an assumption of having a substantial corpus of annotated data encompassing both non-contextual utterances and contextual utterances for training a parser. However, in practice, it is more expensive to collect and annotate contextual utterances compared to non-contextual utterances, due to the dependency on the conversation history. Furthermore, annotating non-contextual utterances usually precedes annotating contextual utterances. To reflect such real-world settings, we study few-shot adaptation for parsing contextual utterances, where we first build a parser

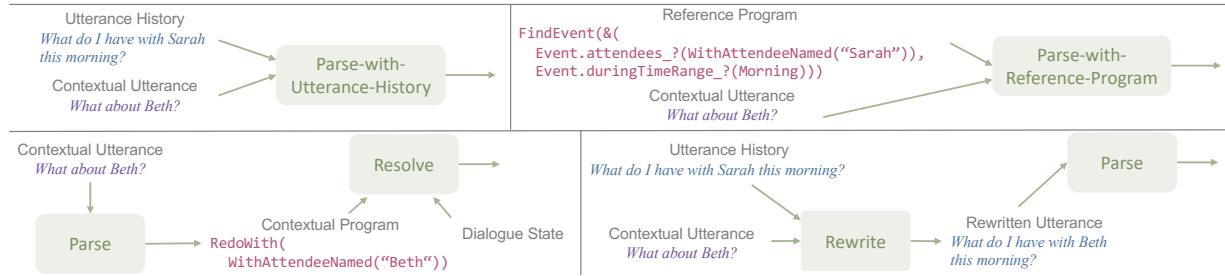


Figure 3.1: Four canonical paradigms of conversational semantic parsing for contextual utterances.

using a large number of annotated non-contextual utterances, and then adapt it for parsing contextual utterances using a few (or even zero) annotated contextual utterances.

Recent work has shown that large language models (LLMs) are capable of semantic parsing using a few examples [68, 67]. Hence, in this work, we conduct a focused study on few-shot adaptation using LLMs for CSP. Specifically, we consider four major paradigms: Parse-with-Utterance-History, Parse-with-Reference-Program, Parse-then-Resolve, and Rewrite-then-Parse. One challenge of carrying out a comparative study on these paradigms is the lack of annotated data, since existing CSP datasets such as SMCaFlow [62] and CoSQL [85] are often annotated based on a single paradigm. Therefore, we construct a new dataset, *SMCaFlow-EQ*, derived from a subset of SMCaFlow dialogues with annotations for all four paradigms.¹

Our experiments consider both in-context learning (ICL) using GPT-3.5 and fine-tuning (FT) using T5-base 220M [51] for building and adapting parsers. ICL typically has lower accuracy compared to FT, although the two are not strictly comparable as they use different models. The only exception is Parse-with-Reference-Program, suggesting that GPT-3.5 is effective at editing programs using natural language. Overall, we find Rewrite-then-Parse to be the most promising approach, as it achieves similar accuracy to other paradigms in both ICL and FT experiments, while requiring only a few annotated examples for to develop a query rewriter and no additional program annotations. We release code and data to facilitate future work on parsing contextual utterances.²

3.2 Background: LLM-Based Parsing

Following [68] and [59], we formulate parsing as a constrained decoding problem, where an LLM is used to predict the next token and a context-free grammar (CFG) is used to

¹A previous version of this chapter was published in [36]

²https://github.com/microsoft/few_shot_adaptation_for_parsing_contextual_utterances_with_llms

validate the predicted token. A program is represented as a sequence of S-expression tokens $y_1 y_2 \dots y_L$. The space of all valid S-expressions is governed by a CFG denoted by \mathcal{G} , which can be automatically derived from function definitions and types used in the domain (see Appendix 3.6).

To generate the program for a user utterance, we first feed the LLM with the user utterance and necessary context information as a sequence of tokens. Then the S-expression of the program is generated incrementally. At each decoding step l , we only keep the partial prefix sequence $y_1 y_2 \dots y_l$ if it is allowed by \mathcal{G} . This validation can be efficiently performed via Earley’s parsing algorithm [13] using the parsing state of the partial sequence $y_1 y_2 \dots y_{l-1}$.

In this paper, we consider both ICL and FT for constructing LLM-based parsers. For ICL, we prompt the pre-trained LLM with K_{ICL} demonstration examples retrieved via BM25 [58, 56], following [60] and [59]. For FT, we continue training the LLM on K_{FT} demonstration examples, producing a new model to be used during constrained decoding.

3.3 Few-Shot Adaptation

In this paper, we assume there are a large number (M) of annotated non-contextual utterances, $\mathcal{D} = \{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), \dots, (\mathbf{x}^{(M)}, \mathbf{y}^{(M)})\}$, where $\mathbf{x}^{(i)}$ denotes the i -th non-contextual utterance in the dataset, $\mathbf{y}^{(i)}$ is the corresponding non-contextual program, and M is the number of annotated examples. These examples are used to derive a grammar \mathcal{G}_1 and build the parser \mathcal{P}_1 for non-contextual utterances via either ICL or FT.

For a contextual utterance \mathbf{u}_t at the t -th turn of a dialogue, the goal is to obtain the non-contextual program \mathbf{y}_t using the *utterance history* $\mathbf{h}_t = [\mathbf{u}_{<t}]$, the corresponding programs $\mathbf{y}_{<t}$, and/or other information recorded in the dialogue state. Figure 3.1 illustrates four canonical paradigms for parsing contextual utterances. For each of these paradigms, we would like to obtain a new parser by adapting from the base parser \mathcal{P}_1 using N demonstration examples, where $N \ll M$.

Parsing Paradigms

Parse-with-Utterance-History: In this paradigm, the parser directly predicts \mathbf{y}_t by conditioning on the contextual utterance \mathbf{u}_t and its history \mathbf{h}_t . This paradigm has been used in contextual semantic parsing [86, 70] and belief state tracking [45].

Parse-with-Reference-Program: This paradigm assumes that the salient additional context to parse \mathbf{u}_t is captured by a *reference program*, which is a non-contextual program to be revised and typically that from the preceding turn, \mathbf{y}_{t-1} . The parsing process can be viewed as editing the reference program based on the contextual utterance which directly yields \mathbf{y}_t . [87] employs a similar strategy by using a copy operation during parsing to copy tokens from the reference program for text-to-SQL.

Parse-then-Resolve: This paradigm divides the task into two steps, leading to a modularized system with a parser followed by a resolver. \mathbf{u}_t is first mapped to an intermediate program

$\tilde{\mathbf{y}}_t$ which contains specialized contextual symbols. These contextual symbols (marking ellipsis or coreference) are resolved deterministically using the dialogue state determined from $\mathbf{y}_{<t}$, resulting in the final non-contextual prediction \mathbf{y}_t . Several recent datasets for CSP have adopted this paradigm [62, 10].

Rewrite-then-Parse: This paradigm modularizes the system using a rewriter followed by a parser. The history \mathbf{h}_t and contextual utterance \mathbf{u}_t are first rewritten into a single non-contextual utterance \mathbf{u}'_t . Then, \mathbf{u}'_t is parsed to \mathbf{y}_t by a single-turn semantic parser. This paradigm is closely related to incomplete utterance rewriting [39] and conversational query rewriting [52, 84, 9, 69, 21, 42] though the parsing step is usually unnecessary or overlooked in these related studies. Using this paradigm, the rewriter and the parser can be independently developed and maintained.

Adaptation via ICL

For ICL, we use GPT-3.5 and the following prompt template provided by [68] and [59], where placeholders $\{X1\}$, $\{X2\}$, \dots are demonstrations input, $\{Y1\}$, $\{Y2\}$, \dots are demonstrations output, and $\{X'\}$ is the test input.

```
Let's translate what a human user says into what
a computer might say.
```

```
Human: {X1}
Computer: {Y1}
```

```
Human: {X2}
Computer: {Y2}
```

```
...
```

```
Human: {X'}
Computer:
```

For Parse-with-Utterance-History, Parse-with-Reference-Program, and Parse-then-Resolve, the input placeholders are respectively instantiated as $\mathbf{h} \mid \mathbf{u}, \mathbf{r} \mid \mathbf{u}$, and \mathbf{u} , where the character \mid is used as the separator. The output placeholders are all instantiated by non-contextual programs \mathbf{y} , except for Parse-then-Resolve which uses $\tilde{\mathbf{y}}$ instead. The test input placeholder follows the same form as demonstration input placeholders. New CFG rules are derived from the program annotations of contextual utterances, *i.e.*, $\tilde{\mathbf{y}}$ and \mathbf{y} , yielding two new grammars \mathcal{G}_α and \mathcal{G}_β , respectively. During constrained decoding, the joint grammar $\mathcal{G}_1 \cup \mathcal{G}_\alpha$ is used for Parse-then-Resolve, whereas $\mathcal{G}_1 \cup \mathcal{G}_\beta$ is used for the other three paradigms. In other words, the adaptation only changes the set of demonstration examples used during prompt instantiation and augments the CFG used during constrained decoding.

For Rewrite-then-Parse, we can re-use the same grammar \mathcal{G}_1 and parser \mathcal{P}_1 used for non-contextual utterances, without any annotated programs for contextual utterances.

Adaptation via FT

For FT, the parser \mathcal{P}_1 for non-contextual utterances uses an LLM \mathcal{M}_1 fine-tuned from T5-base 220M [51]. To adapt this parser for contextual utterances, we continue fine-tuning \mathcal{M}_1 on annotated contextual utterances, except for Rewrite-then-Parse which uses \mathcal{P}_1 itself. Similar to ICL, different forms of token sequences are used for different paradigms, *i.e.*, $\mathbf{h} \mid \mathbf{u} \mid \mathbf{y}$ for Parse-with-Utterance-History, $\mathbf{r} \mid \mathbf{u} \mid \mathbf{y}$ for Parse-with-Utterance-History, and $\mathbf{u} \mid \tilde{\mathbf{y}}$ for Parse-then-Resolve. The new grammar is constructed identically to ICL as well.

Data Annotation Effort

An important axis when comparing different parsing paradigms is the data annotation effort. For Parse-with-Utterance-History, annotating the non-contextual program for a contextual utterance can be a cognitively demanding task, as it needs to account for the full utterance history. Data annotation for Parse-with-Reference paradigm is similar to the Parse-with-Utterance-History, though it may be less cognitively intensive because the human annotator only needs to make a few edits as opposed to performing a full parse. Compared with Parse-with-Utterance-History, annotations of intermediate programs in the Parse-then-Resolve paradigm are much less context-dependent and more concise, which potentially makes the parser more data efficient. However, this comes at a cost of placing a greater burden on the resolver, which uses custom-designed contextual symbols based on the domain; their expressiveness can greatly affect the quality of the annotations and the complexity of the resolver. Finally, collecting annotations for the utterance rewriting task is relatively easy and domain independent compared to collecting annotations for parsers which often requires learning a domain-specific language.

3.4 Experiments

Data

Existing CSP datasets are often annotated based on only one or two paradigms, making it difficult to compare across different paradigms comprehensively. To address this challenge, we construct a dataset *SMCalFlow-EventQueries* (*SMCalFlow-EQ*) derived from a subset of SMCalFlow [62]. It contains 31 training and 100 test instances in total. Each instance consists of a contextual user utterance \mathbf{u} during an event-related query (*e.g.*, “*what about Tuesday?*”), the corresponding contextual/intermediate program $\tilde{\mathbf{y}}$ and non-contextual program \mathbf{y} , the utterance history \mathbf{h} , the reference program \mathbf{r} , and the rewritten non-contextual utterance \mathbf{u}' . The programs (\mathbf{y} , $\tilde{\mathbf{y}}$, \mathbf{r}) are semi-automatically derived from the original SMCalFlow annotations. The rewritten non-contextual utterances \mathbf{u}' are manually annotated by domain experts. See Appendix 3.7 for details of the dataset construction and examples.

We additionally use 8892 training and 100 test instances of non-contextual utterances (*e.g.*, “*do I have any meetings scheduled after Thursday?*”), each paired with their corresponding non-

Paradigm	ICL	FT
Parse-with-Utterance-History	51.8	81.2
Parse-with-Reference-Program	86.1*	78.2
Parse-then-Resolve	70.5*	82.4
Rewrite-then-Parse	65.3*	75.2
Rewrite-then-Parse (oracle)	76.2*	94.0*

Table 3.1: Exact match accuracy on *SMCalFlow-EQ* test set. For both ICL and FT, we test each paradigm against the corresponding Parse-with-Utterance-History predictions using McNemar’s test and show statistically significant ($p < 0.05$) results with *.

contextual programs, semi-automatically derived from *SMCalFlow* as well. These instances are used to construct and evaluate the base parser \mathcal{P}_1 for non-contextual utterances.

Experimental Results

For Parse-with-Reference-Program, we use the oracle reference program, which is the non-contextual program of the preceding turn.³ For Parse-then-Resolve, we assume an oracle resolver is available, which in practice can be implemented as a rule-based system. The rewriter used for Rewrite-then-Parse is implemented via GPT-3.5, and details are provided in Appendix 3.10. We also consider using the oracle rewritten utterances annotated in the contextual subset of *SMCalFlow-EQ*.

We evaluate the program exact match accuracy on the *SMCalFlow-EQ* test set for all paradigms. Table 3.1 presents the experimental results. Across all paradigms, FT achieves higher exact match than ICL by 7.9% to 29.4% absolute gain. For FT, Rewrite-then-Parse with oracle rewritten utterances performs the best. There is no significant difference among other approaches, including Rewrite-then-Parse using the GPT-3.5 rewriter which does not require additional fine-tuning. For ICL, Parse-with-Reference-Program performs the best, suggesting it is easier for GPT-3.5 to softly edit a program than parsing directly from natural language. Rewrite-then-Parse using oracle rewritten utterances is still better than the remaining approaches. By comparing the results of Rewrite-then-Parse, it is clear that improving the rewriter can lead to a corresponding improvement in parsing accuracy.

We manually examine incorrect predictions made by parsers for contextual utterances and identify common error categories: incorrect top-level program types, alternative parses for the input, extra constraints, missing constraints, and constraints with incorrect arguments/-functions (see Table 3.2 for examples).

³It is possible that the reference program is from an earlier turn or does not appear in the history, though the contextual subset does not contain such examples.

Error Type	Gold	Predicted
Top-level Incorrect	<pre>(Execute (ReviseConstraint (DefaultRootLocation) (^ (Event) ConstraintTypeIntension) (& (Event.attendees_? (WithAttendeeNamed "kim")) (& (Event.onDate_? (Tomorrow)) (Event.subject_? (?~= "lunch meeting"))))))</pre>	<pre>(Execute (NewClobber (DefaultIntension) (^ (Recipient) ConstraintTypeIntension) (intension (RecipientWithNameLike (^ (Recipient) EmptyStructConstraint) (PersonName.apply "kim")))))</pre>
Alternate parse	<pre>(FindEventWrapperWithDefaults (& (Event.attendees_? (WithAttendeeNamed "Barry")) (Event.start_? (DateTime.date_? (?= (Tomorrow)))))</pre>	<pre>(FindEventWrapperWithDefaults (& (Event.attendees_? (WithAttendeeNamed "Barry")) (Event.onDate_? (Tomorrow)))</pre>
Extra Constraint	<pre>(QueryEventResponseIsNonEmpty (FindEventWrapperWithDefaults (Event.attendees_? (& (WithAttendeeNamed "Marco") (WithAttendeeNamed "Peyton")))))</pre>	<pre>(QueryEventResponseIsNonEmpty (FindEventWrapperWithDefaults (& (Event.attendees_? (WithAttendeeNamed "Peyton")) (& (Event.attendees_? (WithAttendeeNamed "Marco")) (Event.duringDateRangeConstraint_? (FullMonthofMonth (Date.month (Today)))))</pre>
Missing Constraint	<pre>(QueryEventResponseIsNonEmpty (FindEventWrapperWithDefaults (& (Event.attendees_? (WithAttendeeNamed "Bob")) (& (Event.duringTimeRangeConstraint_? (Afternoon)) (Event.onDate_? (Tomorrow)))))</pre>	<pre>(QueryEventResponseIsNonEmpty (FindEventWrapperWithDefaults (& (Event.attendees_? (WithAttendeeNamed "Bob")) (Event.duringTimeRangeConstraint_? (Afternoon)))))</pre>

Table 3.2: Error examples with gold parses. (part 1 of 2)

For ICL, the most common error type is incorrect function calls. 30% of the errors made by Parse-with-Reference-Program are due to incorrect function use. In particular, the model struggles with predicting rare functions such as negations, potentially because the only

Error Type	Gold	Predicted
Constraint With Incorrect Function	<pre>(Execute (NewClobber (DefaultIntension) (extensionConstraint (^ (LocationKeyphrase) AlwaysTrueConstraint)) (intension (LocationKeyphrase.apply "EVO"))))</pre>	<pre>(Execute (NewClobber (DefaultIntension) (^ (Recipient) ConstraintTypeIntension) (intension (RecipientWithNameLike (^ (Recipient) EmptyStructConstraint) (PersonName.apply "EVO"))))</pre>
Constraint With Incorrect Argument	<pre>(QueryEventResponseIsNonEmpty (FindEventWrapperWithDefaults (Event.onDate_? (adjustByPeriod (Tomorrow) (toDays 1))))</pre>	<pre>(QueryEventResponseIsNonEmpty (FindEventWrapperWithDefaults (Event.onDate_? (adjustByPeriod (Tomorrow) (toDays 2))))</pre>

Table 3.3: Error examples with gold parses. (part 2 of 2)

knowledge of the target language is from the contextual subset of *SMCalFlow-EQ*.

For FT, 33% of the errors in Parse-then-Resolve are from incorrect top-level program types. Introducing new symbols increases the program space, especially different intermediate programs that have similar functions, suggesting that the design of these specialized contextual symbols is crucial. For Parse-with-Utterance-History, we find that 40% of the errors come from missing constraints, indicating that jointly learn parsing and consolidating constraints from multiple turns is challenging for the parsing model. For Rewrite-then-Parse, 55% of the errors are due to incorrect arguments, and 45% are due to differences in capitalization (*e.g.*, the rewriter converts a lowercase name to uppercase) which is arguably less critical.

We also examine the overall parsing accuracy on the joint test set of contextual and non-contextual utterances. We use a binary classifier which takes the user utterance as input and determines whether to use the parser for non-contextual utterances or the parser for contextual utterances. The classifier is obtained by fine-tuning the RoBERTa-base [41] to on *SMCalFlow-EQ* utterances. The overall classification accuracy is 95.5%. The results are summarized in Table 3.5. We use exact match accuracy as the evaluation metric, where the prediction is treated as correct only when classification and parsing are both correct.

3.5 Limitations

Due to the cost of collecting program annotations for all paradigms, the size of the *SMCalFlow-EQ* test set is relatively small and we only study dialogues from *SMCalFlow*. While the

Paradigm	ICL	FT
Parse-with-Utterance-History	63.5	84.5
Parse-with-Reference-Program	79.5*	83.0
Parse-then-Resolve	73.5*	85.5
Rewrite-then-Parse	69.5*	82.0
Rewrite-then-Parse (oracle)	75.0*	90.5*

Table 3.4: Exact match accuracy on *SMCalFlow-EQ* test set combined with non-contextual utterances. For both ICL and FT, we test each paradigm against the corresponding Parse-with-Utterance-History predictions using McNemar’s test and show statistically significant ($p < 0.05$) results indicated with *.

experiments results are informative under significance test, it would be useful for future work to conduct a similar study on larger and diverse datasets.

The LLMs used in this work are pre-trained primarily on English, and the *SMCalFlow-EQ* also only contains English utterances. It would be interesting to study the few-shot adaptation problem on other languages.

3.6 CFG for Constrained Decoding

The CFG used for constrained decoding can be automatically derived from function definitions and types used in the domain. For example, given a function $\text{FN}(\text{arg}_1, \dots, \text{arg}_N)$ with corresponding argument types τ_1, \dots, τ_N and output type τ_O , we can automatically derive a CFG rule $\text{NT}_{\tau_O} \rightarrow (\text{FN} (\text{NT}_{\tau_1} \dots \text{NT}_{\tau_N}))$ where NT_{τ_i} denotes the non-terminal symbol for the type τ_i , and the function name FN and the parentheses are terminal symbols in \mathcal{G} . For each primitive type (e.g., “string”, “number”), we additionally define CFG rules to expand the non-terminal of the primitive type to terminals representing acceptable values of the type (sometimes using regular expressions).

3.7 Dataset Construction and Examples

The original *SMCalFlow* data only provide annotations of *contextual programs* for individual utterances. We develop a heuristic-based implementation of `NewClobber` and `ReviseConstraint` to propose candidates of the corresponding non-contextual programs. Specifically, given the non-contextual program

```
(NewClobber (
  (intension)
  (slotConstraint)
  (value)))
```

we modify the non-contextual program of the preceding turn by replacing its fragment satisfying the `slotConstraint` with the new fragment value. Similarly, given the non-contextual program

```
(ReviseConstraint (
  (rootLocation)
  (oldLocation)
  (newConstraint)))
```

we modify the non-contextual program of the preceding turn by replacing a fragment `oldConstraint` which satisfies the `oldLocation` and is governed by a bigger fragment satisfying the `rootLocation` with a new fragment

```
(& ((oldConstraint) (newConstraint)))
```

i.e., conjoining the two constraints regardless whether they conflicts with each other. For both cases, when there are multiple possible replacements, all resulting candidates are proposed. These candidates are manually reviewed and edited by the authors to finalize non-contextual program annotations. For example, if `newConstraint` contradicts with a part of `oldConstraint`, we drop the such conflicting parts in the `oldConstraint`.

Furthermore, as noted by [43], the original annotations of *SMCalFlow* can be complex and contain many boilerplate segments. Therefore, we use heuristics to simplify the original annotations to obtain programs that are shorter and potentially easier to read and predict. Similar to [43], the simplification was implemented via a set of tree transformation rules, which convert specific sub-trees of the original program into simplified sub-trees. The list of sub-tree transformations are provided in Table 3.8–Table 3.10.

Two data specialists are asked to produce the annotations for the rewritten non-contextual utterances in the contextual subset. They are provided with instructions and training materials, which explains how to rewrite a contextual user utterance with its preceding utterance into a single non-contextual utterance. Each example takes 10 to 30 seconds to annotate. Additionally, annotators were asked to provide a confidence from 0 (least confident) to 3 (most confident) in the rewritten utterance. The average confidence was 2.9. Then they are asked to review the each other’s annotations and answer whether they agree with each other. In our pilot data collection, the agreement rate between the two data specialists was 93.3%.

Table 2.4 provides some examples from in *SMCalFlow-EQ*.

3.8 Fine-tuning Experiment Hyperparameters

For fine-tuning, we employ the Adafactor optimizer [64] and set the batch size to 32. The slanted triangular learning rate scheduler [20] is used with a maximum learning rate of 10^{-5} and 1000 warmup steps. We fine-tune \mathcal{M}_0 for 10000 steps on the non-contextual subset to obtain \mathcal{M}_1 , and another 10000 steps on the corresponding data to obtain the models for

Paradigm	ICL	FT
Parse-with-Utterance-History	63.5	84.5
Parse-with-Reference-Program	79.5*	83.0
Parse-then-Resolve	73.5*	85.5
Rewrite-then-Parse	69.5*	82.0
Rewrite-then-Parse (oracle)	75.0*	90.5*

Table 3.5: Exact match accuracy on *SMCalFlow-EQ* test set combined with non-contextual utterances. For both ICL and FT, we test each paradigm against the corresponding Parse-with-Utterance-History predictions using McNemar’s test and show statistically significant ($p < 0.05$) results indicated with *.

individual paradigms. For constrained decoding, the maximum output sequence length is 1000.

3.9 Joint Results

To analyze how this affects parsing overall, we also evaluate on joint test set of *SMCalFlow-EQ* and the additional 100 non-contextual utterances. We use a binary classifier which takes the user utterance as input and determines whether to use the parser for non-contextual utterances or the parser for contextual utterances. The classifier is obtained by fine-tuning the RoBERTa-base [41] to on *SMCalFlow-EQ* utterances. The overall classification accuracy is 95.5%.

The results are summarized in Table 3.5. We use exact match accuracy as the evaluation metric, where the prediction is treated as correct only when classification and parsing are both correct.

3.10 Rewriter Implementation

The rewriter used for Rewrite-then-Parse is implemented via GPT-3.5 (`text-davinci-003`). The prompt template is shown below, where placeholders $\{H1\}$, $\{H2\}$, \dots are for the utterance history (*i.e.*, the preceding utterances), $\{X1\}$, $\{X2\}$, \dots are for contextual user utterances, $\{Z1\}$, $\{Z2\}$, \dots are for rewritten non-contextual utterances, and $\{H'\}$ and $\{X'\}$ are for test input.

```
Combine the utterances into a single utterance
with the meaning of the last utterance.
```

```
Last Utterance: {H1}
Current Utterance: {X1}
```

```
Rewritten Utterance: {Z1}
Last Utterance: {H2}
Current Utterance: {X2}
Rewritten Utterance: {Z2}
...
Last Utterance: {H' }
Current Utterance: {X' }
Rewritten Utterance:
```

We sample 8 demonstration examples are sampled uniformly from the contextual subset training instances. Greedy decoding is used with 50 maximum tokens and no frequency or presence penalty. The BLEU score using the oracle rewritten utterances as reference is 93.6.

3.11 Conclusion

We study a real-world CSP setting, *i.e.*, few-shot adaptation for parsing contextual utterances with LLMs, and compare four different paradigms using both ICL and FT. To facilitate the study, we construct a new dataset, *SMCalFlow-EQ* with annotations for all paradigms. Experiments show that ICL with GPT-3.5 usually underperforms FT with T5-base except for Parse-with-Reference-Program, suggesting GPT-3.5 is good at editing programs via natural language in these data conditions. Overall, Rewrite-then-Parse stands out as a promising approach for future development of LLM-based CSP, as it performs as well as other paradigms but require only a few annotated examples for the rewriter and no additional program annotation.

Example 1	
Utterance	What about later next week?
Last Utterance	Did I have any meetings early next week?
Oracle Rewritten Utterance	Did I have any meetings later next week?
Non-Contextual Program	<pre>(QueryEventResponseIsNonEmpty (FindEventWrapperWithDefaults (Event.duringDateRangeConstraint_? (LateDateRange (NextWeekList))))</pre>
Contextual Program	<pre>(Execute (ReviseConstraint (DefaultRootLocation) (^ (Event) ConstraintTypeIntension) (Event.duringDateRangeConstraint_? (LateDateRange (NextWeekList))))</pre>
Example 2	
Utterance	Actual I meant the day after tomorrow.
Last Utterance	Is there any appointments tomorrow?
Oracle Rewritten Utterance	Is there any appointments the day after tomorrow?
Non-Contextual Program	<pre>(QueryEventResponseIsNonEmpty (FindEventWrapperWithDefaults (Event.onDate_? (adjustByPeriod (Tomorrow) (toDays 1))))</pre>
Contextual Program	<pre>(Execute (ReviseConstraint (DefaultRootLocation) (^ (Event) ConstraintTypeIntension) (Event.onDate_? (adjustByPeriod (Tomorrow) (toDays 1))))</pre>

Table 3.6: Examples of temporal query revisions and their corresponding programs.

Example 3	
Utterance	What about training?
Last Utterance	Is there a vacation scheduled for me?
Oracle Rewritten Utterance	Is there a training scheduled for me?
Non-Contextual Program	<pre>(QueryEventResponseIsNonEmpty (FindEventWrapperWithDefaults (Event.subject_? (?~= "training"))))</pre>
Contextual Program	<pre>(Execute (ReviseConstraint (DefaultRootLocation) (^ (Event) ConstraintTypeIntension) (Event.subject_? (?~= "training"))))</pre>

Table 3.7: Example of subject-based query revision and its corresponding programs.

Original	Simplified
<pre>(& (^(\$type) EmptyStructConstraint) (\$c))</pre>	<pre>(\$c)</pre>
<pre>(& (\$c) (^(\$type) EmptyStructConstraint))</pre>	<pre>(\$c)</pre>
<pre>(> (size (QueryResponse.results (\$response))), 0L)</pre>	<pre>(QueryEventResponseIsNonEmpty (\$response))</pre>
<pre>(AttendeeListHasRecipientConstraint (RecipientWithNameLike (^ (Recipient) EmptyStructConstraint) (PersonName.apply \$name)))</pre>	<pre>(WithAttendeeNamed (\$name))</pre>
<pre>(AttendeeListHasRecipient (Execute (refer (extensionConstraint (RecipientWithNameLike (^ (Recipient) EmptyStructConstraint) (PersonName.apply \$name))))))</pre>	<pre>(WithAttendeeNamed (\$name))</pre>
<pre>(AttendeeListExcludeRecipient (Execute (refer (extensionConstraint (RecipientWithNameLike (^ (Recipient) EmptyStructConstraint) (PersonName.apply \$name))))))</pre>	<pre>(WithoutAttendeeNamed (\$name))</pre>

Table 3.8: List of sub-tree transformations for simplifying SMCaFlow programs (part 1).

Original	Simplified
<pre>(EventAtTime (\$event) (\$time))</pre>	<pre>(& (\$event) (Event.atTime_? (\$time)))</pre>
<pre>(EventDuringRangeTime (\$event) (\$timeRange)))</pre>	<pre>(& (\$event) (Event.duringTimeRangeConstraint_? (\$timeRange)))</pre>
<pre>(EventOnDate (\$date) (\$event))</pre>	<pre>(& (\$event) (Event.onDate_? (\$date)))</pre>
<pre>(EventDuringDateRange (\$event) (\$dateRange)))</pre>	<pre>(& (\$event) (Event.duringDateRangeConstraint_? (\$dateRange)))</pre>

Table 3.9: List of sub-tree transformations for simplifying SMCaFlow programs (part 2).

Original	Simplified
<pre>(EventOnDateTime (DateAtTimeWithDefaults ((\$date) (\$time)) (\$event)))</pre>	<pre>(& (\$event) (& (Event.onDate_? (\$date)) (Event.atTime_? (\$time))))</pre>
<pre>(EventOnDateAfterTime ((\$date) (\$event) (\$time)))</pre>	<pre>(& (\$event) (& (Event.onDate_? (\$date)) (Event.afterTime_? (\$time))))</pre>
<pre>(EventOnDateBeforeTime ((\$date) (\$event) (\$time)))</pre>	<pre>(& (\$event) (& (Event.onDate_? (\$date)) (Event.beforeTime_? (\$time))))</pre>
<pre>(EventOnDateFromTimeToTime ((\$date) (\$event) (\$time1) (\$time2)))</pre>	<pre>(& (\$event) (& (Event.onDate_? (\$date)) (Event.betweenTimeAndTime_? (\$time1) (\$time2))))</pre>

Table 3.10: List of sub-tree transformations for simplifying SMCaFlow programs (part 3).

Original	Simplified
<pre>(EventAfterDateTime ((\$event) (\$dateTime)))</pre>	<pre>(& (\$event) (Event.afterDateTime_? (\$dateTime)))</pre>
<pre>(EventBeforeDateTime ((\$event) (\$dateTime)))</pre>	<pre>(& (\$event) (Event.beforeDateTime_? (\$dateTime)))</pre>
<pre>(EventOnDateWithTimeRange (EventOnDate (\$date) (\$event)) (\$timeRange))</pre>	<pre>(& (\$event) (& (Event.onDate_? (\$date)) (Event.duringTimeRangeConstraint_? (\$timeRange))))</pre>
<pre>(EventOnDateWithTimeRange (EventDuringRange (\$event) (\$dateRange) (\$timeRange)))</pre>	<pre>(& (\$event) (& (Event.duringDateRangeConstraint_? (\$dateRange)) (Event.duringTimeRangeConstraint_? (\$timeRange))))</pre>
<pre>(EventDuringRangeDateTime (\$event) (\$dateTimeRange))</pre>	<pre>(& (\$event) (Event.duringDateTimeRangeConstraint_? (\$dateTimeRange)))</pre>

Table 3.11: List of sub-tree transformations for simplifying SMCaFlow programs (part 4).

Chapter 4

Context Tuning

The previous two chapters covered two core patterns of structuring contexts for LLMs, to improve systems by what input contexts they are given. In this chapter, we cover not what context LLMs are given, but how they handle such contexts. As LLMs are increasingly trained with longer context lengths, it is appealing to structure context less to LLMs and rely more on the LLMs to handle the entire contexts. However, we observe that LLMs are often not trained to fully handle the contexts that are given. We present an approach that improves how LLMs use context by structuring relevant context across the input context of the LLM more uniformly, and structuring the output such that the LLM first outputs relevant tokens from the input context before the final generation. Across two document question answering tasks, we show up to 12% gain in accuracy when fine-tuning LLMs to better context.

4.1 Introduction

Recent advances in language models have significantly improved the long-context abilities of language models i.e. the number of tokens that a language model that can take as input, with commercial language models offering up to 10M tokens [53]. Long-context models are appealing because they open up the potential for many applications such as summarization of entire books [79], code assistants over entire repositories of code [40] etc. Moreover, the generality of the approach is appealing as the all context could be given to the language model, and the model internally can process the task without any task-specific engineering.

However, various empirical studies have shown the limitations of long-context models showing that language models have positional biases and can be easily distracted by irrelevant context [71, 38, 65]. In particular, for retrieval-augmented language model systems, increasing the amount of retrieved context does not necessarily improve downstream performance. Figure 4.1 shows that increasing the number of documents increases the recall of retrieval systems; however, when models are presented with documents, they are not able to use these contexts, and question answering performance saturates quickly.

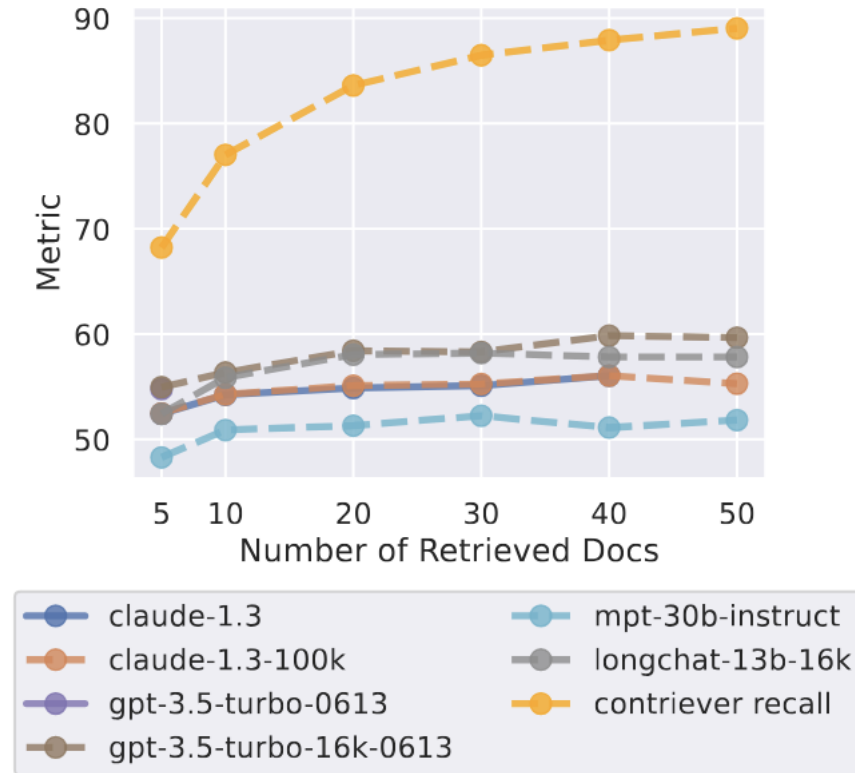


Figure 4.1: Retriever recall and model performance as a function of the number of retrieved documents. Model performance saturates long before retriever recall, indicating that the models have difficulty making use of the extra retrieved documents. Figure originally published in [38].

The size of language model context windows greatly outpace the natural distribution of document lengths in corpora that are typically used to train language models. As a reference point, commercial language offerings such as Gemini have context lengths of up to 10M tokens [53] and common open-source models such as Llama have up to hundreds of thousands of tokens [12], while corpora such as Wikipedia on average have articles with only a couple hundred words.¹ As a result, much of the the data used to train language models does not require the model to reason over the majority of the tokens. Beyond the length of the documents, most of the the tokens in language modeling does not require learning long-term dependencies and shorter inputs also yield competitive performance in language modeling

¹As of writing, the average size of Wikipedia is 686. For more details, see https://en.wikipedia.org/wiki/Wikipedia:Size_of_Wikipedia

Figure 4.2: Document sampling and construction process for generating examples from Context Tuning.

```

Input: Corpus  $\mathcal{C} = \{d_1, \dots, d_N\}$  of documents
Input: Number of negative documents  $n$ 
Input: Chunk size  $h$ 
// Sample positive document
Sample document  $d_p = \{d_p^1, \dots, d_p^k\}$  uniformly from  $\mathcal{C}$ 
// Select splitting point
Choose random split point  $s \in [1, k]$ 
prefix  $\leftarrow \{d_p^1, \dots, d_p^s\}$ 
completion  $\leftarrow \{d_p^{s+1}, \dots, d_p^k\}$ 
// Sample negative documents
 $\mathcal{N} \leftarrow$  Sample  $n$  documents from  $\mathcal{C} \setminus \{d_p\}$ 
// Chunk documents
prefix_chunks  $\leftarrow$  SplitIntoChunks(prefix,  $h$ )
for each  $d_{\text{neg}} \in \mathcal{N}$  do
    neg_chunks  $\leftarrow$  SplitIntoChunks( $d_{\text{neg}}$ ,  $h$ )
    Add neg_chunks to all_chunks
end for
// Interleave chunks
interleaved_prefix  $\leftarrow$  InterleaveChunks(prefix_chunks, all_chunks)
// Create training example
training_example  $\leftarrow$  (interleaved_prefix, prefix||[NOTES]||completion)
Output: Training example with mixed positive and negative content

```

[48]. Language modeling also uses the same amount of compute per token, and adaptive inference time procedures are typically prompting techniques such as chain-of-thought [77].

Based on these observations, we design context tuning, an approach for fine-tuning language models to better use their context by structuring the contexts of the input prefix, and the output context. We design our approach with the following desiderata. First, the method should be unsupervised, and require minimal effort to scale up to larger datasets to maintain the benefits of vanilla language modeling. Secondly, the method should provide training signal across the entire context window. Third, the method should give signal to the model to use more compute, by generating more tokens, for more difficult examples.

4.2 Context Tuning

where:

\mathcal{C}	is the corpus of documents
d_p	is the sampled positive document
d_p^i	is the i -th token of the positive document
s	is the splitting point
n	is the number of negative documents
h	is the chunk size
\oplus	denotes concatenation
[NOTES]	is a special delimiter token

At a high level, Context Tuning samples documents and interleaves it with negative irrelevant documents to construct noisy input contexts where LLMs first need to recite relevant parts of the positive document before generating the completion. We take a data oriented approach to improving language models' ability to use context and without making modifications to the language modeling loss itself. We first initialize models with LLMs that have been pretrained with the standard approach of unsupervised language modeling following by instruction tuning. We then apply Context-Tuning in two phases: unsupervised Context Tuning then supervised Context Tuning.

Unsupervised Context Tuning In the unsupervised phase, we start by sampling documents and mixing in negatives. Given a corpus of C of documents $\{d_1, \dots, d_N\}$. To generate data samples, we first sample a document d_p which is composed of tokens $\{d_p^1, \dots, d_p^k\}$ from the corpus uniformly at random. We then select a splitting point s between 1 to k to and divide document into two input context $\{d_p^1, \dots, d_p^s\}$ and completion context, $\{d_p^{s+1}, \dots, d_p^k\}$. We then select n negative documents as from the corpus. The negative documents and the positive document prefix are then split into size h chunks of tokens. To construct the prefix, we then interleave the chunks of the negative document and positive prefix to create the full prefix for the language model. The completion blocks $\{d_p^1, \dots, d_p^s\}$ [NOTES] $\{d_p^{s+1}, \dots, d_p^k\}$, where [NOTES] is a special delimiter token. We then do supervised fine-tuning on these examples, where the cross entropy loss is computed on the completion tokens. 4.2 details the data construction process. Figure 4.6 shows an example of the constructed data, where parts of the noisy input context needs to be first recited before the final completion. Figure 4.3 shows an overview of the data construction for the unsupervised phase in context tuning.

Supervised Context-Tuning Following Unsupervised Context Tuning, we do task specific fine-tuning for each of the tasks. Each of the tasks contain triples of question q , answer a and supporting set of document $D_s = \{d_s^1, \dots, d_s^k\}$. For each triple (q_i, i, D_s) we construct input set of documents $D_c = \{d_1, \dots, d_k\}q_i$ that fill the context window, such that $D_s \in D_c$, all the supporting documents are in the context window. The output is then constructed the supporting document $\{d_s^1, \dots, d_s^k\}a_1$, where the model is trained the first output each of the supporting documents before generating the final answer.

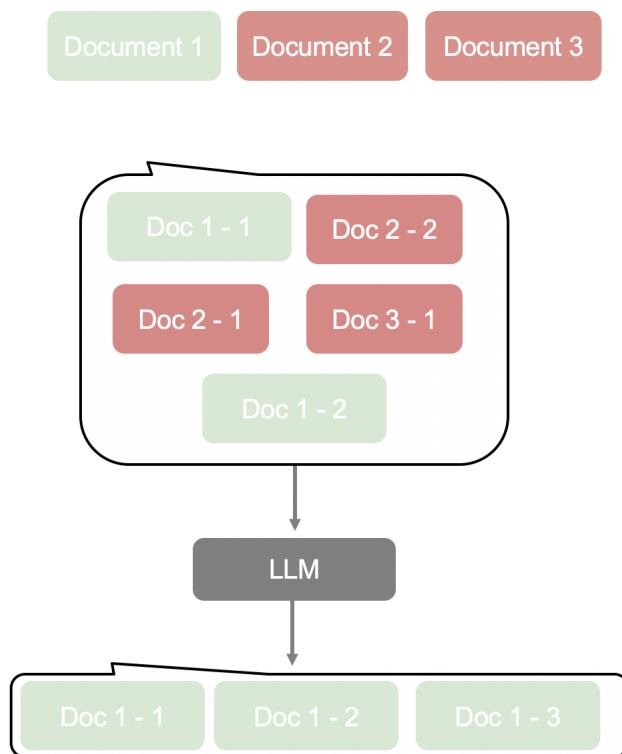


Figure 4.3: Overview of the unsupervised phase of Context Tuning. First, positive documents, Document 1 (green) is sampled, along with negative documents, Documents 2, and Document 3, (red). The LM is then trained to first recite the parts of Document 1 that are in the context before generating the completion of the document.

4.3 Experiments

Data and Tasks

We experiment with two realistic document question answering tasks that require context selection, i.e. part of the context is relevant for answering the question, and the rest of the context is filled with distractor documents.

Single Context Relevance: The first is based on the task introduced in [38] based on the data released in NaturalQuestions [31]. NaturalQuestions contains document, question, answer triples grounded in Wikipedia from anonymized real-world queries from Google. In this task, designed to probe the positional sensitivity of the model, the model is given a list of k documents, k , 1 of which answers the question and $k - 1$ documents which do not answer the question. This then allows the supporting document’s position to be modulated, and ideally models are able to answer correctly regardless of the position of the supporting document.

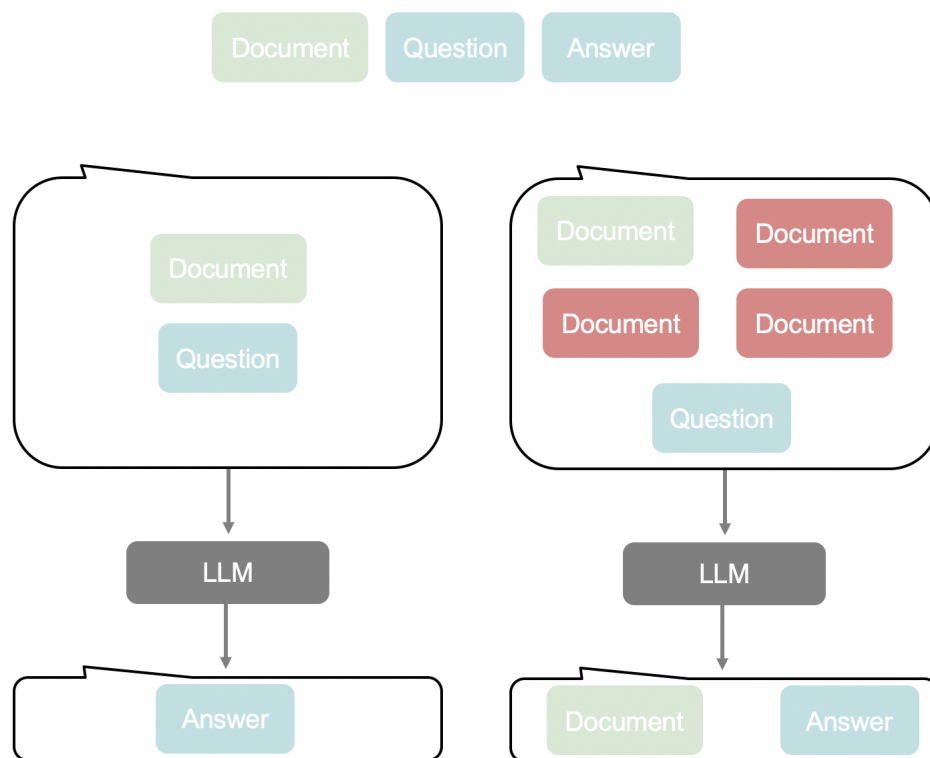


Figure 4.4: Overview of the supervised phase of Context Tuning. In standard supervised finetuning (left), given document question and answer triples, the model is trained to generate the answer given the document and question (left). IN the supervised phase of Context Tuning we introduce negative documents (red) in the input context, and train the model to first generate the supporting document before the answer.

We use both randomly selection documents as negative documents as well documents that have the highest scores with *contriever* [23]. As our evaluation metric, we follow [38] and use exact substring match, where the prediction is considered correct if the gold short answer appears as a substring in the model generation. We evaluate the task with total of 10 and 20 documents.

Multi-Context Relevance: To test ability for identifying multiple context that are relevant, we also experiment with a similar setup using HotPotQA [83]. In this setup, there are k documents, and j documents that are relevant to the question, where $0 < j \leq k$. We then also modulate the position of the supporting documents, selecting j positions to place the supporting documents.

"Prompt: Below is a set of documents, some of which are related to each other. Output the ones that are relevant to the last document and then continue writing the last document
 Input:[DOC] Stadion Ljajkovići is a football stadium in Ljajkovići Urban Municipality of Golubovci Podgorica Capital Montenegro It is the home ground of FK Bratstvo The stadium holds 300 seats History Until 2007 FK Bratstvo played their home games in the village of Cijevna near the Aluminium Plant Podgorica But during 2007 a new stadium was built in the neighbouring village of Ljajkovići Since then FK Bratstvo plays all their games at Ljajkovići stadium The stadium meets criteria for Second League games but not for top league matches See also FK Bratstvo Cijevna Golubovci Zeta Plain Podgorica External links Stadium information References [DOC] Ljajkovići Football in Montenegro Sport in Podgorica Buildings and structures in Podgorica [DOC]
 Torjus Hansén born 29 October 1973 is a Norwegian former professional footballer who played as a defender Club career Hansén was born in Skien He formerly played for Gulset Lillestrøm Arminia Bielefeld and Rosenborg For his last club Odd Grenland has played total eleven seasons and 267 matches and scoring 6 goals He retired after the 2010 season On 8 February 2011 he sign a contract until 1 August 2011 On 1 August 2011 he retired from football International career He also played three matches for the national team and three for the under 21 References External links 1973 births [DOC] Red Jets is a 2006 3D combat flight simulator inspired by the Cold War conflict and the USSR Air Force It was developed by Graffiti Entertainment Gameplay The player can fly four famous Soviet jets the MiG 29 Fulcrum Yak 141 Freestyle Su 27 Flanker and Su 25 Frogfoot The game features relatively realistic flight physics and detailed cockpit and external views Twenty missions and thirty scenarios come with the game The landscapes were modelled from actual satellite maps In addition a detailed visual weather system to simulate flight in almost unlimited combinations of weather and day is included in [DOC] the game It also features a multiplayer mode where up to 16 players could play simultaneously The game also came with a mission creator which was criticized for its lack of openness Combat flight simulators [DOC] Living people Association football defenders Norwegian footballers Norway international footballers Norway youth

Figure 4.5: Example of running Context Tuning to generate data examples on the Wikipedia. Text before the cyan text corresponds to the segments of the positive document that are part of the document that is reconstructed. In the output, the blue corresponds to text that is present in the input context. The green corresponds to the tokens that complete the document. (part 1 of 2)

Response: answer: Torjus Hansén (born 29 October 1973) is a Norwegian former professional footballer who played as a defender. Club career Hansén was born in Skien. He formerly played for Gulset, Lillestrøm, Arminia Bielefeld and Rosenborg. For his last club, Odd Grenland, has played total eleven seasons and 267 matches and scoring 6 goals. He retired after the 2010 season. On 8 February 2011 he sign a c ontract until 1 August 2011. On 1 August 2011 he retired from football. International career He also played three matches for the national team and three for the under-21. References External links 1973 births Living people Association football defenders Norwegian footballers Norway international footballers Norway youth [ANS] international footballers Odd Grenland players Rosenborg BK players Lillestrøm SK players Arminia Bielefeld players Gulset IF players"

Figure 4.6: Example of running Context Tuning to generate data examples on the Wikipedia. Text before the cyan text corresponds to the segments of the positive document that are part of the document that is reconstructed. In the output, the blue corresponds to text that is present in the input context. The green corresponds to the tokens that complete the document. (part 2 of 2)

Implementation Details

Split Point Selection: we choose the split point uniformly between the start and the end tokens, with the constraint that there is always 16 BPE tokens of the positive document at the end of the input prefix, and 16 BPE tokens as part of the competition tokens of the positive document.

Negative Example Selection: we select negative documents for training with two settings. First, we randomly sample uniformly from document titles in the Wikipedia corpus. Secondly, we compute the closest passage embedding of the title of the positive document contriever, and then sample from the top 10 documents [22]

Training We use a batch size of 32, and gradient accumulation of 4. We use the the AdamW optimizer with weight decay of 0.01 and gradient clipping of 1. We use learning rate: $3e-5$ with linear decay, with 500 warmup steps and train for 10k training steps. For the supervised stage, we train for 1k steps with a learning rate sweep between $3e-6$, $3e-5$, and $3e-4$.

Models We run experiments on two open-weight models: Llama 2 Chat 7B model. Both models are transformer-based models that have first been run trained with language modeling followed by instruction tuning. ² and MistralAI’s Mistral-7B-v0.1 ³.

²<https://huggingface.co/meta-llama/Llama-2-7b-chat-hf>

³<https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.1>

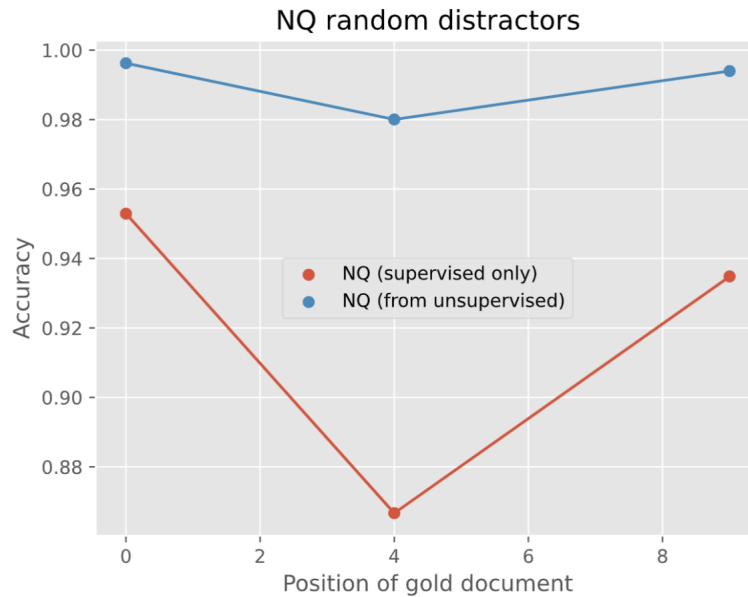


Figure 4.7: Results comparing Llama 2 Chat fine tuned directly on supervised data with first using Context Tuning for 10 total documents on the NaturalQuestions task with random Wikipedia distractors. Context Tuning improves results over directly fine-tuning on supervised data.

4.4 Limitations

One limitation is that the tasks considered in this study are both retrieve and read style tasks, where a small subset of the input contexts are relevant. However, for many tasks that long-context models would be useful for eg. summarization of a book, do not closely match the Context Tuning training objective. Another limitation is that the generation tokens during Context Tuning are extract, as they reconstruct the positive document from the noisy contexts. During generation, this often encourages the model to copy from the input contexts which is not necessarily desirable, and prevents the model from generating additional tokens more flexibly that lead to the final response. One limitation of the experimental setting is that it is possible that the models ignore context and are able to answer the questions without the supporting context. In previous work, it has been shown that in HotPotQA [44], many of the questions can be solved without actually reasoning across the contexts.

4.5 Results and Discussion

Context Tuning Improves Performance: Context Tuning shows improvements on both the NaturalQuestions setup as well as the HotPotQA setups. Figure 4.7 shows the results

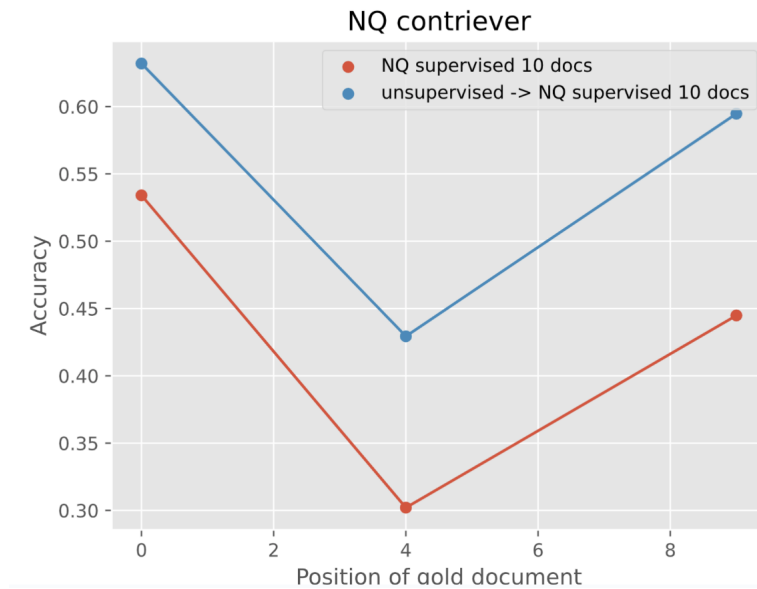


Figure 4.8: Results comparing Llama 2 Chat fine tuned directly on supervised data with first using Context Tuning for 10 total documents on the NaturalQuestions task with contriever Wikipedia distractors. Context Tuning improves results over directly fine-tuning on supervised data.

Model	Index 0	Index 4	Index 9
Llama 2 Chat	0.95	0.87	0.93
Llama 2 Chat (Context Tuning)	0.99	0.98	0.99
Mistral-7B-v0.1	0.97	0.91	0.95
Mistral-7B-v0.1 (Context Tuning)	0.99	0.99	0.99

Table 4.1: Model performance comparing only tuning on supervised data on total 10 documents in the NaturalQuestions task. Context Tuning corresponds to our full approach of first training on unsupervised examples followed by supervised examples.

comparing directly finetuning on supervised examples on the NaturalQuestions task, tested with random distractors from Wikipedia, showing that using context tuning has up to 11% gain in accuracy. Figure 4.8 shows the results comparing directly finetuning on supervised examples on the NaturalQuestions task, tested but with using the most relevant examples using a state-of-the-art retrieval model, showing up to 12% gain in accuracy. Context Tuning also improves on setups that require locating multiple pieces of relevant context in the input context. Across all positions of the supporting contexts, we see improvements of Context Tuning before supervised fine-tuning. For the single-relevance context setting with

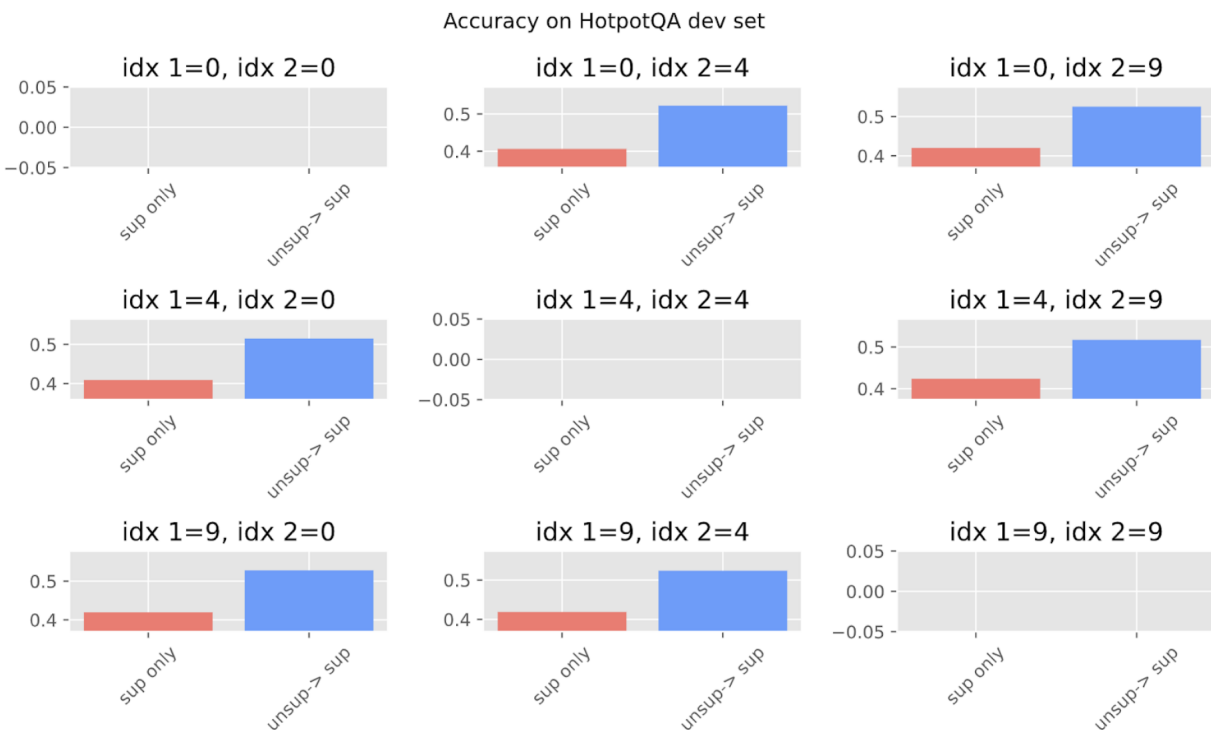


Figure 4.9: Results on the development set of HotPotQA comparing directly finetuning Llama 2-Chat on examples of HotPotQA (red) with first applying Context Tuning on the subset of data where two contexts are relevant according to the original HotpotQA annotations. Index 1 and index 2 correspond to the position of supporting contexts.

Model	Index 0	Index 4	Index 9	Index 14	Index 19
Llama 2 Chat	0.69	0.55	0.54	0.50	0.51
Llama 2 Chat (Context Tuning)	0.74	0.66	0.60	0.62	0.61
Mistral-7B-v0.1	0.72	0.60	0.61	0.59	0.60
Mistral-7B-v0.1 (Context Tuning)	0.78	0.70	0.65	0.61	0.62

Table 4.2: Model performance comparing only tuning on supervised data on total 10 documents in the NaturalQuestions task. Context Tuning corresponds to our full approach of first training on unsupervised examples followed by supervised examples.

NaturalQuestions, full results are shown for the settings for total 10 documents in Table 4.1 and for total 20 documents in Table 4.2. For the multi-supporting document setting using HotPotQA, full results are shown for the settings for total 10 documents in Table 4.3 and for

Model	Indices 0, 4	Indices 0, 9	Indices 4, 0	Indices 4, 0	Indices 9, 0	Indices 9, 4
Llama 2 Chat	0.41	0.41	0.42	0.41	0.42	0.43
Llama 2 Chat (CT)	0.52	0.52	0.51	0.51	0.52	0.52
Mistral-7B-v0.1	0.55	0.54	0.55	0.53	0.55	0.56
Mistral-7B-v0.1 (CT)	0.58	0.58	0.58	0.57	0.57	0.57

Table 4.3: Model performance comparing only tuning on supervised data on total 10 documents in the HotPotQA task. Context Tuning (CT) corresponds to our full approach of first training on unsupervised examples followed by supervised examples.

Model	Indices 0, 14	Indices 0, 19	Indices 14, 0	Indices 14, 0	Indices 19, 0	Indices 19, 14
Llama 2 Chat	0.25	0.25	0.24	0.25	0.25	0.25
Llama 2 Chat (CT)	0.30	0.30	0.30	0.29	0.29	0.30
Mistral-7B-v0.1	0.38	0.38	0.38	0.37	0.37	0.38
Mistral-7B-v0.1 (CT)	0.41	0.42	0.42	0.38	0.41	0.41

Table 4.4: Model performance comparing only tuning on supervised data on total 20 documents in the HotPotQA task. Context Tuning (CT) corresponds to our full approach of first training on unsupervised examples followed by supervised examples.

total 20 documents in Table 4.4.

Positional Bias Persist in Context Tuning Our previous work showed that models often exhibit positional bias [38]. We show that both the primacy effect, models paying more attention to the beginning of context and the recency bias, models paying more attention to the end of contexts are exhibited after fine-tuning the models with Context Tuning. In this study, we placed the relevant contexts uniformly in the context window when applying Context Tuning; however, as the structure of the data is synthetic controlled, one natural question is whether the positional biases would persist if we apply different placements of the relevant context, eg. if the relevant context were placed inversely to the performance of the models without Context Tuning, the positional biases may be alleviated.

4.6 Related Work

Several lines of work have the goal of improving the overall use of the context. A common line trend is staged fine tuning, where the training process is divided into phases. In each phase, the language model trains on a set context length, and is scaled up from shorter to longer context lengths. The primary benefit of this approach is that its simplicity and the efficiency

gains from initially tuning on shorter sequences. Some notable implementations include MPT-30B model, which successfully extended context length to 8k tokens through staged fine-tuning of their base checkpoint ⁴, Salesforce’s XGen model series ⁵ and Together.ai’s implementation with Llama-2-7B, extending context to 32k tokens ⁶. However, a significant limitation is the extended adaptation period required for models to effectively handle longer sequences.

Another line of work is positional interpolation leveraging the properties of rotary embeddings. Rotary embeddings can better handle context extension by projecting them onto the contexts lengths that models have been trained on [8]. Compared to directly finetuning on longer sequences, there are clear improvement gains in efficiency compared to staged finetuning, requiring as little as 1k steps to get improvements.

Equipping models with external memory systems is another approach to extending context. For example Focused Transformer [74] and Memorizing Transformer [80] are two prominent examples of architectures that involve explicit memory systems that enable longer context windows by adding in memory mechanisms to help the model make select operations on data with long term dependencies, up to thousands of tokens away.

Our construction objective is related to previous denoising objectives used in earlier sequence-to-sequence models such as BART, where the model is presented corrupted text and is tasked with reconstructing the original text [33]. More recently [66], shared similar data-oriented perspectives for creating contexts that enable better long-context use, and focus on selecting documents that enable long-context capabilities by concatenating relevant documents together to form the input context.

4.7 Conclusion

We present a data oriented approach to improving how language models use context context. Based on the observation that context windows of LLMs are quickly outpacing that of the corpora that they are trained on, and that the training signal for LLMs are not diffuse throughout their context windows, we propose Context Tuning, a data oriented approach that uniformly spaces relevant contexts in the context window. We experiment with two document question answering tasks that ground answers in documents, and construct targeted tasks that probe LLMs with irrelevant contexts. Compared to direct fine-tuning we see gains of up to 12% absolute improvement in accuracy.

⁴<https://huggingface.co/mosaicml/mpt-30b>

⁵<https://www.salesforce.com/blog/xgen/>

⁶<https://www.together.ai/blog/llama-2-7b-32k>

Chapter 5

Conclusion

This dissertation contributes to the abstractions and computational patterns for structuring the context of large language models. As large language models become a core computational substrate for larger intelligence systems, the ability to achieve more complex tasks will depend on what context language models are given, and how they compute over those contexts.

In chapter 2, we introduced context decomposition, using large language models to break down more complex queries into shorter queries that specialized models can handle. In chapter 3, we introduced context rewriting, showing that using multiple models to manage complex utterances. In chapter 4, we presented context-tuning, a technique for structuring the input and output token space that allow language models to better handle complex contexts. Across complex tasks in various domains: information retrieval, semantic parsing, and document question answering, this dissertation demonstrates paradigms for improving language models by structuring their contexts.

The rest of this chapter provides an overview of projects that were developed during work on the dissertation, and concludes with future directions.

5.1 Additional Projects

Compute Efficiency In [35], we showed that training larger language models can be more efficient than smaller models. For smaller computational budgets, the conventional wisdom is that models with fewer parameters were more efficient, because larger models were too expensive per step. We showed that training larger models is more efficient because models do not necessarily need to be trained to convergence, and larger models are far more sample efficient than smaller models. We further showed that for compression techniques such as weight pruning and quantization, larger models are more robust to being compressed. Taken together, showing that larger models could be trained more efficiently and then heavily compressed to match the parameter counts of smaller models, and the optimal strategy is given fixed compute is to "train large, then compress". The ideas developed in this work are more studied as part of neural scaling laws [19].

Taxonomy Construction In [7], we present a method for constructing taxonomic trees (e.g., WordNet) using language models. In this task, the model is given a set of words and the output is a taxonomic tree. Our approach used a modular approach where a language scores parenthood scores given pairs of terms, creating a graph of parenthood scores. Then, another module reconciles the predictions by treating the task as graph optimization problem and outputting the maximum spanning tree. Furthermore, we showed that incorporating additional context from by retrieving glosses from the web improved performance even more. On the task of constructing subtrees of English WordNet, our approach improved the best previous approach by 20.0% relative increase. We also convert the original dataset into nine additional languages by using Open Multilingual Wordnet.

Context Freshness In [78], we build a system for improving context freshness for model serving systems. The de-facto way of giving up to date information to language models is by maintaining a feature store (otherwise known as embedding store), and inputting these as context to the language model. As items are updated, they features over the items need to be recomputed eg. if a document is edited and the representation of the document was a neural embedding then it would then need to be recomputed. Depending on the scale and the frequency of the updates, maintaining up to date features can be prohibitively expensive. Typically, feature stores apply a fixed strategy for updating these features eg. re-compute the features each time there is an updates or update at fixed time intervals. We build a system that orchestrates feature updates by leveraging downstream feedback, i.e. how much not computing the right features affect downstream metrics. For example, small formatting changes to text may not be as important to re-compute features for than changes to text that dramatically alter the semantics. We show that on realistic feature store workloads in anomaly detection and recommendation of up to 31.7% reduction in prediction error.

Self-Editing Memory In [47], we build an open-source system, MemGPT that manages its own context. Drawing from how operating systems manage memory, we propose virtual context management, a technique that gives the appearance of larger context windows by managing data movement between the context window and external data stores for language models. MemGPT manages different memory tiers and uses the interrupts and function calling to orchestrate data movement between the different memory tiers. We show experiments on processing long documents, as well as perpetual dialogue.

5.2 Future Work

Reasoning Over Extended Context: Much of the work on reasoning is focused on evaluating reasoning where the relevant information is not diffuse throughout the input context. An area of promising work is developing methods to identify, connect, and synthesize relevant information that is scattered across long documents or multiple sources. This

requires models to maintain coherent chains of logic while tracking dependencies between distant pieces of information. Creating evaluations and methods for such task is a promising area of future work. Current methods either suffer from attention being too diffuse and distracted by irrelevant context or explicit memory structures that are less flexible.

Self-Structuring Context The work in this thesis involves designing the structure eg. leveraging the schema of the document representations in chapter 2, using the structure that utterances refer to previous utterances in chapter 3 and chapter 4 designing the input and output context sections. One natural area of future work is to automatically discover the underlying structures without having to manually specify them with LLMs. For example, LLMs could first cluster the document representations before decomposing queries, uncovering better decompositions than ones specified by the document schemas.

Inference Context Structuring Before Test-Time: Another dimension of inference compute scaling is additional inference on context that is given before test time. Throughout this thesis, we see how structuring the context based on input queries, utterances, and questions at test time. However, a lot of context is given to the language model before the test-time, i.e. documents and previous interactions in chapter 2, previous conversations in chapter 3, and retrieval corpora in chapter 4. Additional compute could be spent on structuring and processing these input contexts before the test-time context.

Bibliography

- [1] Jacob Andreas et al. “Neural Module Networks”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2016, pp. 39–48.
- [2] Jaime Arguello et al. “Tip of the Tongue Known-Item Retrieval: A Case Study in Movie Identification”. In: *Proceedings of the 2021 Conference on Human Information Interaction and Retrieval*. CHIIR '21. Canberra ACT, Australia: Association for Computing Machinery, 2021, pp. 5–14. ISBN: 9781450380553. DOI: 10.1145/3406522.3446021. URL: <https://doi.org/10.1145/3406522.3446021>.
- [3] Leif Azzopardi, Maarten de Rijke, and Krisztian Balog. “Building Simulated Queries for Known-Item Topics: An Analysis Using Six European Languages”. In: *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '07. Amsterdam, The Netherlands: Association for Computing Machinery, 2007, pp. 455–462. ISBN: 9781595935977. DOI: 10.1145/1277741.1277820. URL: <https://doi.org/10.1145/1277741.1277820>.
- [4] Samarth Bhargav, Georgios Sidiropoulos, and Evangelos Kanoulas. “‘It’s on the Tip of My Tongue’: A New Dataset for Known-Item Retrieval”. In: *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. WSDM '22. Virtual Event, AZ, USA: Association for Computing Machinery, 2022, pp. 48–56. ISBN: 9781450391320. DOI: 10.1145/3488560.3498421. URL: <https://doi.org/10.1145/3488560.3498421>.
- [5] Michael K. Buckland. “On types of search and the allocation of library resources”. In: *Journal of the American Society for Information Science* 30.3 (1979), pp. 143–147. DOI: <https://doi.org/10.1002/asi.4630300305>. eprint: <https://asistdl.onlinelibrary.wiley.com/doi/pdf/10.1002/asi.4630300305>. URL: <https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/asi.4630300305>.
- [6] Daniel Fernando Campos et al. “MS MARCO: A Human Generated MACHine Reading COMprehension Dataset”. In: *ArXiv abs/1611.09268* (2016).
- [7] Catherine Chen, Kevin Lin, and Dan Klein. “Constructing Taxonomies from Pretrained Language Models”. In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2021, pp. 4687–4700.

- [8] Shouyuan Chen et al. “Extending context window of large language models via positional interpolation”. In: *arXiv preprint arXiv:2306.15595* (2023).
- [9] Zheng Chen, Xing Fan, and Yuan Ling. “Pre-Training for Query Rewriting in a Spoken Language Understanding System”. In: *Proceedings of 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2020, pp. 7969–7973. DOI: 10.1109/ICASSP40776.2020.9053531.
- [10] Jianpeng Cheng et al. “Conversational Semantic Parsing for Dialog State Tracking”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 8107–8117. DOI: 10.18653/v1/2020.emnlp-main.651. URL: <https://aclanthology.org/2020.emnlp-main.651>.
- [11] Yizhou Chi et al. “CLARINET: Augmenting Language Models to Ask Clarification Questions for Retrieval”. In: *arXiv preprint arXiv:2405.15784* (2024).
- [12] Abhimanyu Dubey et al. “The llama 3 herd of models”. In: *arXiv preprint arXiv:2407.21783* (2024).
- [13] Jay Earley. “An Efficient Context-Free Parsing Algorithm”. In: *Communications of the ACM* 13.2 (1970), pp. 94–102. DOI: 10.1145/362007.362035. URL: <https://doi.org/10.1145/362007.362035>.
- [14] David Elsweiler et al. “Seeding Simulated Queries with User-Study Data for Personal Search Evaluation”. In: *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’11. Beijing, China: Association for Computing Machinery, 2011, pp. 25–34. ISBN: 9781450307574. DOI: 10.1145/2009916.2009924. URL: <https://doi.org/10.1145/2009916.2009924>.
- [15] Luyu Gao and Jamie Callan. “Unsupervised Corpus Aware Language Model Pre-training for Dense Passage Retrieval”. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 2843–2853. DOI: 10.18653/v1/2022.acl-long.203. URL: <https://aclanthology.org/2022.acl-long.203>.
- [16] Matthias Hagen, Daniel Wagner, and Benno Stein. “A Corpus of Realistic Known-Item Topics with Associated Web Pages in the ClueWeb09”. In: *Advances in Information Retrieval*. Ed. by Allan Hanbury et al. Cham: Springer International Publishing, 2015, pp. 513–525. ISBN: 978-3-319-16354-3.
- [17] Claudia Hauff and Geert-Jan Houben. “Cognitive Processes in Query Generation”. In: *Advances in Information Retrieval Theory*. Ed. by Giambattista Amati and Fabio Crestani. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 176–187. ISBN: 978-3-642-23318-0.

- [18] Claudia Hauff et al. “Towards Realistic Known-Item Topics for the ClueWeb”. In: *Proceedings of the 4th Information Interaction in Context Symposium*. IIX ’12. Nijmegen, The Netherlands: Association for Computing Machinery, 2012, pp. 274–277. ISBN: 9781450312820. DOI: 10.1145/2362724.2362773. URL: <https://doi.org/10.1145/2362724.2362773>.
- [19] Tom Henighan et al. “Scaling laws for autoregressive generative modeling”. In: *arXiv preprint arXiv:2010.14701* (2020).
- [20] Jeremy Howard and Sebastian Ruder. “Universal Language Model Fine-tuning for Text Classification”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 328–339. DOI: 10.18653/v1/P18-1031. URL: <https://aclanthology.org/P18-1031>.
- [21] Shumpei Inoue et al. “Enhance Incomplete Utterance Restoration by Joint Learning Token Extraction and Text Generation”. In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Seattle, United States: Association for Computational Linguistics, July 2022, pp. 3149–3158. DOI: 10.18653/v1/2022.naacl-main.229. URL: <https://aclanthology.org/2022.naacl-main.229>.
- [22] Gautier Izacard et al. “Towards unsupervised dense information retrieval with contrastive learning”. In: *arXiv preprint arXiv:2112.09118* (2021).
- [23] Gautier Izacard et al. “Unsupervised Dense Information Retrieval with Contrastive Learning”. In: *Transactions on Machine Learning Research* (2022). ISSN: 2835-8856. URL: <https://openreview.net/forum?id=jKN1pXi7b0>.
- [24] Harsh Jhamtani et al. “Natural Language Decomposition and Interpretation of Complex Utterances”. In: *arXiv preprint arXiv:2305.08677* (2023).
- [25] Christian Johnson. “Binary Encoded Word Mover’s Distance”. In: *Proceedings of the 7th Workshop on Representation Learning for NLP*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 167–172. DOI: 10.18653/v1/2022.repl4nlp-1.17. URL: <https://aclanthology.org/2022.repl4nlp-1.17>.
- [26] Vladimir Karpukhin et al. “Dense passage retrieval for open-domain question answering”. In: *arXiv preprint arXiv:2004.04906* (2020).
- [27] Omar Khattab and Matei 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*. 2020, pp. 39–48.
- [28] Tushar Khot et al. “Decomposed prompting: A modular approach for solving complex tasks”. In: *arXiv preprint arXiv:2210.02406* (2022).

- [29] Jinyoung Kim and W. Bruce Croft. “Retrieval Experiments Using Pseudo-Desktop Collections”. In: *Proceedings of the 18th ACM Conference on Information and Knowledge Management*. CIKM '09. Hong Kong, China: Association for Computing Machinery, 2009, pp. 1297–1306. ISBN: 9781605585123. DOI: 10.1145/1645953.1646117. URL: <https://doi.org/10.1145/1645953.1646117>.
- [30] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).
- [31] Tom Kwiatkowski et al. “Natural questions: a benchmark for question answering research”. In: *Transactions of the Association for Computational Linguistics* 7 (2019), pp. 453–466.
- [32] Jin Ha Lee, Allen Renear, and Linda C. Smith. “Known-Item Search: Variations on a Concept”. In: *Proceedings of the American Society for Information Science and Technology* 43.1 (2006), pp. 1–17. DOI: <https://doi.org/10.1002/meet.14504301126>. eprint: <https://asistdl.onlinelibrary.wiley.com/doi/pdf/10.1002/meet.14504301126>. URL: <https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/meet.14504301126>.
- [33] Mike Lewis et al. “BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 7871–7880.
- [34] Patrick Lewis et al. “PAQ: 65 Million Probably-Asked Questions and What You Can Do With Them”. In: *Transactions of the Association for Computational Linguistics* 9 (2021), pp. 1098–1115. DOI: 10.1162/tacl_a_00415. URL: <https://aclanthology.org/2021.tacl-1.65>.
- [35] Zhuohan Li et al. “Train big, then compress: Rethinking model size for efficient training and inference of transformers”. In: *International Conference on machine learning*. PMLR, 2020, pp. 5958–5968.
- [36] Kevin Lin, Patrick Xia, and Hao Fang. “Few-Shot Adaptation for Parsing Contextual Utterances with LLMs”. In: *Findings of the Association for Computational Linguistics: IJCNLP-AACL 2023 (Findings)* (2023).
- [37] Kevin Lin et al. “Decomposing Complex Queries for Tip-of-the-tongue Retrieval”. In: *Findings of Empirical Methods in Natural Language Processing* (2023).
- [38] Nelson F Liu et al. “Lost in the middle: How language models use long contexts”. In: *Transactions of the Association for Computational Linguistics* 12 (2024), pp. 157–173.
- [39] Qian Liu et al. “Incomplete Utterance Rewriting as Semantic Segmentation”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 2846–2857. DOI: 10.18653/v1/2020.emnlp-main.227. URL: <https://aclanthology.org/2020.emnlp-main.227>.

- [40] Tianyang Liu, Canwen Xu, and Julian McAuley. “Repobench: Benchmarking repository-level code auto-completion systems”. In: *arXiv preprint arXiv:2306.03091* (2023).
- [41] Yinhan Liu et al. “RoBERTa: A Robustly Optimized BERT Pretraining Approach”. In: *arXiv:1907.11692* (2019). URL: <https://arxiv.org/abs/1907.11692>.
- [42] Kelong Mao et al. “Large Language Models Know Your Contextual Search Intent: A Prompting Framework for Conversational Search”. In: *arXiv:2303.06573* (2023). URL: <https://arxiv.org/abs/2303.06573>.
- [43] Joram Meron. “Simplifying Semantic Annotations of SMCaFlow”. In: *Proceedings of the 18th Joint ACL - ISO Workshop on Interoperable Semantic Annotation within LREC2022*. Marseille, France: European Language Resources Association, June 2022, pp. 81–85. URL: <https://aclanthology.org/2022.isa-1.11>.
- [44] Sewon Min et al. “Compositional questions do not necessitate multi-hop reasoning”. In: *arXiv preprint arXiv:1906.02900* (2019).
- [45] Nikola Mrkšić et al. “Neural Belief Tracker: Data-Driven Dialogue State Tracking”. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, July 2017, pp. 1777–1788. DOI: 10.18653/v1/P17-1163. URL: <https://aclanthology.org/P17-1163>.
- [46] Sheshera Mysore, Arman Cohan, and Tom Hope. “Multi-Vector Models with Textual Guidance for Fine-Grained Scientific Document Similarity”. In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Seattle, United States: Association for Computational Linguistics, July 2022, pp. 4453–4470. DOI: 10.18653/v1/2022.naacl-main.331. URL: <https://aclanthology.org/2022.naacl-main.331>.
- [47] Charles Packer et al. “Memgpt: Towards llms as operating systems”. In: *arXiv preprint arXiv:2310.08560* (2023).
- [48] Ofir Press, Noah A Smith, and Mike Lewis. “Shortformer: Better Language Modeling using Shorter Inputs”. 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)*. 2021, pp. 5493–5505.
- [49] Alec Radford et al. “Language models are unsupervised multitask learners”. In: *OpenAI blog* 1.8 (2019), p. 9.
- [50] Alec Radford et al. “Learning transferable visual models from natural language supervision”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8748–8763.
- [51] Colin Raffel et al. “Exploring the limits of transfer learning with a unified text-to-text transformer”. In: *The Journal of Machine Learning Research* 21.1 (2020), pp. 5485–5551. URL: <https://jmlr.org/papers/volume21/20-074/20-074.pdf>.

- [52] Pushpendre Rastogi et al. “Scaling Multi-Domain Dialogue State Tracking via Query Reformulation”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Industry Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 97–105. DOI: 10.18653/v1/N19-2013. URL: <https://aclanthology.org/N19-2013>.
- [53] Machel Reid et al. “Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context”. In: *arXiv preprint arXiv:2403.05530* (2024).
- [54] Kirk Roberts et al. “TREC-COVID: rationale and structure of an information retrieval shared task for COVID-19”. In: *Journal of the American Medical Informatics Association* 27.9 (July 2020), pp. 1431–1436. ISSN: 1527-974X. DOI: 10.1093/jamia/ocaa091. eprint: <https://academic.oup.com/jamia/article-pdf/27/9/1431/34153771/ocaa091.pdf>. URL: <https://doi.org/10.1093/jamia/ocaa091>.
- [55] S. E. Robertson and S. Walker. “On Relevance Weights with Little Relevance Information”. In: *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '97. Philadelphia, Pennsylvania, USA: Association for Computing Machinery, 1997, pp. 16–24. ISBN: 0897918363. DOI: 10.1145/258525.258529. URL: <https://doi.org/10.1145/258525.258529>.
- [56] Stephen Robertson and Hugo Zaragoza. “The Probabilistic Relevance Framework: BM25 and Beyond”. In: *Foundations and Trends in Information Retrieval* 3.4 (Apr. 2009), pp. 333–389. ISSN: 1554-0669. DOI: 10.1561/1500000019. URL: <https://doi.org/10.1561/1500000019>.
- [57] Stephen Robertson and Hugo Zaragoza. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc, 2009.
- [58] Stephen E. Robertson and Stephen Walker. “Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval”. In: *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Dublin, Ireland, 1994, pp. 232–241. URL: <https://dl.acm.org/doi/pdf/10.5555/188490.188561>.
- [59] Subhro Roy et al. “BenchCLAMP: A Benchmark for Evaluating Language Models on Semantic Parsing”. In: *arXiv:2206.10668* (2022). URL: <https://arxiv.org/abs/2206.10668>.
- [60] Ohad Rubin, Jonathan Herzig, and Jonathan Berant. “Learning To Retrieve Prompts for In-Context Learning”. In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Seattle, United States: Association for Computational Linguistics, July 2022, pp. 2655–2671. DOI: 10.18653/v1/2022.naacl-main.191. URL: <https://aclanthology.org/2022.naacl-main.191>.

- [61] Christopher Sciavolino et al. “Simple Entity-Centric Questions Challenge Dense Retrievers”. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 6138–6148. DOI: 10.18653/v1/2021.emnlp-main.496. URL: <https://aclanthology.org/2021.emnlp-main.496>.
- [62] Semantic Machines et al. “Task-Oriented Dialogue as Dataflow Synthesis”. In: *Transactions of the Association for Computational Linguistics* 8 (Sept. 2020), pp. 556–571. URL: https://doi.org/10.1162/tacl_a_00333.
- [63] Rico Sennrich, Barry Haddow, and Alexandra Birch. “Neural Machine Translation of Rare Words with Subword Units”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1715–1725. DOI: 10.18653/v1/P16-1162. URL: <https://aclanthology.org/P16-1162>.
- [64] Noam Shazeer and Mitchell Stern. “Adafactor: Adaptive learning rates with sublinear memory cost”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 4596–4604. URL: <https://proceedings.mlr.press/v80/shazeer18a/shazeer18a.pdf>.
- [65] Freda Shi et al. “Large language models can be easily distracted by irrelevant context”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 31210–31227.
- [66] Weijia Shi et al. “In-Context Pretraining: Language Modeling Beyond Document Boundaries”. In: *The Twelfth International Conference on Learning Representations*.
- [67] Richard Shin and Benjamin Van Durme. “Few-Shot Semantic Parsing with Language Models Trained on Code”. In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Seattle, United States: Association for Computational Linguistics, July 2022, pp. 5417–5425. DOI: 10.18653/v1/2022.naacl-main.396. URL: <https://aclanthology.org/2022.naacl-main.396>.
- [68] Richard Shin et al. “Constrained Language Models Yield Few-Shot Semantic Parsers”. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 7699–7715. DOI: 10.18653/v1/2021.emnlp-main.608. URL: <https://aclanthology.org/2021.emnlp-main.608>.
- [69] Shuangyong Song et al. “A Two-Stage Conversational Query Rewriting Model with Multi-Task Learning”. In: *Companion Proceedings of the Web Conference 2020*. WWW ’20. Taipei, Taiwan: Association for Computing Machinery, 2020, pp. 6–7. ISBN: 9781450370240. DOI: 10.1145/3366424.3382671. URL: <https://doi.org/10.1145/3366424.3382671>.

- [70] Alane Suhr, Srinivasan Iyer, and Yoav Artzi. “Learning to Map Context-Dependent Sentences to Executable Formal Queries”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 2238–2249. DOI: 10.18653/v1/N18-1203. URL: <https://aclanthology.org/N18-1203>.
- [71] Simeng Sun et al. “Do Long-Range Language Models Actually Use Long-Range Context?”. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 2021, pp. 807–822.
- [72] James Thorne et al. “FEVER: a Large-scale Dataset for Fact Extraction and VERification”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 809–819. DOI: 10.18653/v1/N18-1074. URL: <https://aclanthology.org/N18-1074>.
- [73] George Tsatsaronis et al. “An overview of the BIOASQ large-scale biomedical semantic indexing and question answering competition”. In: *BMC Bioinformatics* 16 (2015).
- [74] Szymon Tworkowski et al. “Focused transformer: Contrastive training for context scaling”. In: *Advances in Neural Information Processing Systems* 36 (2024).
- [75] David Wadden et al. “SciFact-Open: Towards open-domain scientific claim verification”. In: *Findings of the Association for Computational Linguistics: EMNLP 2022*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 4719–4734. URL: <https://aclanthology.org/2022.findings-emnlp.347>.
- [76] Jianyou Wang et al. “DORIS-MAE: Scientific Document Retrieval using Multi-level Aspect-based Queries”. In: 2023. URL: <https://api.semanticscholar.org/CorpusID:263829433>.
- [77] Jason Wei et al. “Chain-of-thought prompting elicits reasoning in large language models”. In: *Advances in neural information processing systems* 35 (2022), pp. 24824–24837.
- [78] Sarah Wooders et al. “RALF: Accuracy-Aware Scheduling for Feature Store Maintenance”. In: *Proceedings of the VLDB Endowment* 17.3 (2023), pp. 563–576.
- [79] Jeff Wu et al. “Recursively summarizing books with human feedback”. In: *arXiv preprint arXiv:2109.10862* (2021).
- [80] Yuhuai Wu et al. “Memorizing transformers”. In: *arXiv preprint arXiv:2203.08913* (2022).
- [81] Eugene Yang et al. “Retrieval and Richness when Querying by Document”. In: *Biennial Conference on Design of Experimental Search & Information Retrieval Systems*. 2018.

- [82] Zhilin Yang et al. “HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 2369–2380. DOI: 10.18653/v1/D18-1259. URL: <https://aclanthology.org/D18-1259>.
- [83] Zhilin Yang et al. “HotpotQA: A dataset for diverse, explainable multi-hop question answering”. In: *arXiv preprint arXiv:1809.09600* (2018).
- [84] Shi Yu et al. “Few-Shot Generative Conversational Query Rewriting”. In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’20. Virtual Event, China, 2020, pp. 1933–1936. URL: <https://doi.org/10.1145/3397271.3401323>.
- [85] Tao Yu et al. “CoSQL: A Conversational Text-to-SQL Challenge Towards Cross-Domain Natural Language Interfaces to Databases”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 1962–1979. DOI: 10.18653/v1/D19-1204. URL: <https://aclanthology.org/D19-1204>.
- [86] Luke Zettlemoyer and Michael Collins. “Learning Context-Dependent Mappings from Sentences to Logical Form”. In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Suntec, Singapore: Association for Computational Linguistics, Aug. 2009, pp. 976–984. URL: <https://aclanthology.org/P09-1110>.
- [87] Rui Zhang et al. “Editing-Based SQL Query Generation for Cross-Domain Context-Dependent Questions”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 5338–5349. DOI: 10.18653/v1/D19-1537. URL: <https://aclanthology.org/D19-1537>.
- [88] Wayne Xin Zhao et al. “Dense Text Retrieval based on Pretrained Language Models: A Survey”. In: *ArXiv abs/2211.14876* (2022).