



Can LLMs Infer Domain Knowledge from Code Exemplars? A Preliminary Study

Jiajing Guo
jiajing.guo@us.bosch.com
Bosch Research North America
Sunnyvale, California, USA

Vikram Mohanty
vikrammohanty@acm.org
Bosch Research North America
Sunnyvale, California, USA

Hongtao Hao*
hongtao.hao@us.bosch.com
Bosch Research North America
Sunnyvale, California, USA

Liang Gou
liang.gou@us.bosch.com
Bosch Research North America
Sunnyvale, California, USA

Liu Ren
liu.ren@us.bosch.com
Bosch Research North America
Sunnyvale, California, USA

ABSTRACT

As organizations recognize the potential of Large Language Models (LLMs), bespoke domain-specific solutions are emerging, which inherently face challenges of knowledge gaps and contextual accuracy. Prompt engineering techniques such as chain-of-thoughts and few-shot prompting have been proposed to enhance LLMs' capabilities by dynamically presenting relevant exemplars. Are LLMs able to infer domain knowledge from code exemplars involving similar domain concepts and analyze the data correctly? To investigate this, we curated a synthetic dataset containing 45 tabular databases, each has domain concepts and definitions, natural language data analysis queries, and responses in the form of Python code, visualizations, and insights. Using this dataset, we conducted a within-subjects experiment to evaluate the effectiveness of domain-specific exemplars versus randomly selected, generic exemplars. Our study underscores the significance of tailored exemplars in enhancing LLMs' accuracy and contextual understanding in domain-specific tasks, paving the way for more intuitive and effective data analysis solutions.

CCS CONCEPTS

• **Human-centered computing** → **Empirical studies in HCI**.

KEYWORDS

domain-specific data analysis, large language model, LLM, LLM evaluation, prompt engineering

ACM Reference Format:

Jiajing Guo, Vikram Mohanty, Hongtao Hao, Liang Gou, and Liu Ren. 2024. Can LLMs Infer Domain Knowledge from Code Exemplars? A Preliminary Study. In *29th International Conference on Intelligent User Interfaces - Companion (IUI Companion '24)*, March 18–21, 2024, Greenville, SC, USA. ACM, New York, NY, USA, 6 pages. <https://doi.org/10.1145/3640544.3645228>

*This work was completed during an internship at Bosch Research North America.

Permission to make digital or hard copies of part or all 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. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

IUI Companion '24, March 18–21, 2024, Greenville, SC, USA

© 2024 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-0509-0/24/03

<https://doi.org/10.1145/3640544.3645228>

1 INTRODUCTION

In recent times, Large Language Models (LLMs) have demonstrated various capabilities, ranging from writing, editing, and summarizing text [11, 24], to generating and debugging code [18], generating annotations [6], simulating user personas [15], and facilitating user interactions with different UIs [19], among many more emerging capabilities. These capabilities are being increasingly leveraged across diverse industries, including healthcare [4], customer service [16], education, entertainment, and cybersecurity.

As organizations recognize the potential of LLMs, we are starting to see specialized LLM-based solutions that enhance operational efficiency and scalability. In sectors like healthcare, exploratory efforts are being made to utilize LLMs for sorting patient feedback [4], while in the realm of customer service, there is a growing interest in employing these models to analyze and respond to customer inquiries [16]. However, the development of bespoke LLM-integrated solutions is confronted with challenges, notably in bridging domain-specific knowledge gaps and in overcoming potential contextual misunderstandings arising from the absence of industry-specific training data [10].

These challenges become particularly pronounced in the context of data analysis — a pivotal task for enhancing decision-making and boosting operational efficiency within organizations. Domain knowledge relevant to analytics often lacks comprehensive documentation, making it challenging for individuals to learn and apply effectively [9, 21]. Although LLM-driven solutions such as GitHub's CoPilot and ChatGPT's code interpreter [12] have started to demonstrate potential in assisting with data science tasks [3, 14], yet their effectiveness in specialized domains remains to be rigorously assessed.

One promising approach to addressing these challenges involves utilization of prompt engineering techniques, especially few-shot prompting [1], Chain-of-Thought (CoT) [20], and their variations [5, 7, 17, 25]. Few-shot prompting enables LLMs to "universally undertake a multitude of tasks" without gradient updates [25]. When questions pertain to specific domain-specific concepts, prompting LLMs with a few exemplars of implementation code relevant to the domain concepts can potentially generate better responses, as opposed to scenarios where such exemplars are absent. Previous studies have introduced techniques for dynamically choosing these exemplars based on similarity [7, 17]. However, the effectiveness of

these methods in the context of data analysis tasks remains largely unexplored.

To envision LLM-based solutions tailored for domain-specific data analysis, it is imperative to establish foundational hypotheses that may influence the design and effectiveness of such systems. In this paper, we specifically investigate two hypotheses:

- (1) *LLMs may underperform if they lack domain-specific knowledge.*
- (2) *When LLMs are provided with exemplars of data analysis that incorporate specific domain concepts, they may be capable of inferring and applying these concepts to conduct similar analyses within the domain.*

To validate these hypotheses, we curated a synthetic dataset comprising 45 databases selected from Spider [23], a complex, cross-domain semantic parsing, and text-to-SQL dataset. This curated dataset was augmented with domain-specific concepts and definitions. Additionally, we crafted exemplar analyses featuring *questions, corresponding Python code, and insights*. We then conducted a within-subjects experiment, where we used GPT-4 to perform data analysis on this dataset, comparing the effectiveness of domain-specific data analysis exemplars against generic exemplars. This experimental design aims to shed light on the potential of LLMs to adapt to and operate within specialized data analysis contexts.

2 DATABASE FOR DOMAIN-SPECIFIC DATA ANALYSIS

A common use case of LLMs in data analysis tasks involves prompting LLMs with natural language queries to generate data retrieval or manipulation scripts in programming languages such as Python and SQL, then subsequently execute the code. Existing datasets that merge natural language with data analysis code [13] are often not adequately tailored for nuanced, domain-specific tasks. In response, we created DomainQuery, a custom-designed dataset designed to facilitate the investigation of LLMs' performance in domain-specific data analysis.

Our dataset is derived from the Spider dataset, originally created for text-to-SQL tasks. Spider has 200 databases across 138 domains, where each database has multiple tables. From Spider's extensive collection, we initially selected 78 databases with a limitation of no more than three tables each to ensure manageability and clarity, which we ultimately refined to 45 databases.

Our process involved a detailed review of the tables within each database to conceptualize domain-specific concepts and definitions. These concepts varied from quantifiable measures like indices and scores to more complex constructs such as matrices or charts. Although most concepts we created are artificial and are not typically used in real-world data analysis tasks, we strived to make these concepts semantically intuitive, ensuring that their definitions are straightforward and easy to understand. Two of the authors reviewed these tables and formulated analytical questions related to each concept. For instance, the database "musical" has two tables: musical and actor. The two authors first reviewed the table columns and defined a domain concept "Actor Performance Index (API)", which is calculated by dividing the actor's age by the number of musicals they have performed in. This gives an idea

of the experience level of an actor relative to their age. Later, questions like "Who are the top 5 actors with the highest Actor Performance Index (API)?" were crafted to guide the creation of exemplars.

Following the identification of domain concepts, we generated corresponding Python code and insights, which served as exemplars. These exemplars were not merely illustrative of a method but embedded with domain knowledge pertinent to the questions posed. To ensure precision and relevance, all code was executed in a Jupyter Notebook environment, validating the accuracy of the exemplars. We used ChatGPT to polish domain names, define domain concepts, and generate code and insights. This procedure involved iterative interactions with the ChatGPT web application. Figure 1 shows an example of an item in the database¹.

By integrating domain-specific insights directly into the dataset, we aim to evaluate whether LLMs exhibit improved performance when domain knowledge is intricately woven into their prompts in the format of code exemplars, as opposed to generic exemplars that lack contextual depth.

3 EXPERIMENT: DOMAIN-SPECIFIC EXEMPLARS V/S GENERIC EXEMPLARS

To investigate the efficacy of LLMs in domain-specific data analysis, we conducted a within-subjects experiment contrasting the effectiveness of domain-specific exemplars with that of generic exemplars.

3.1 LLM Chain Setup

We configured a ReAct-based LLM chain [22], which orients the LLM to iterate through reasoning, action, and observation cycles. Our prompts consisted of a preamble to establish the context, a detailed data profile, and format instructions, followed by exemplars that included a user question, Python code, and insights (see Figure 2). This approach primed LLMs to generate informed reasoning paths and relevant actions in response to the data analysis queries presented.

The LLM chain allows a maximum of four iterations. Upon each execution failure, error messages are appended to the chat history, and a new generation is requested. If the maximum iterations are reached without successful code execution, the output will be empty.

3.2 Experiment Setup

We designed two experimental conditions, with the only distinction being the exemplar employed in the initial prompt.

- **Baseline Condition (Generic Exemplars):** For the baseline, we used generic exemplars, which we designed to act as templates guiding the LLMs, akin to the established practice in few-shot prompting experiments. We crafted three generic exemplars, unrelated to the 45 databases in our primary experiment, and randomly selected one to serve as a prompt example.
- **Experimental Condition (Domain-Specific Exemplars):** Under the experimental condition, exemplars were tailored to address the same domain concept pertinent to the question

¹<https://github.com/boschresearch/DomainQuery>

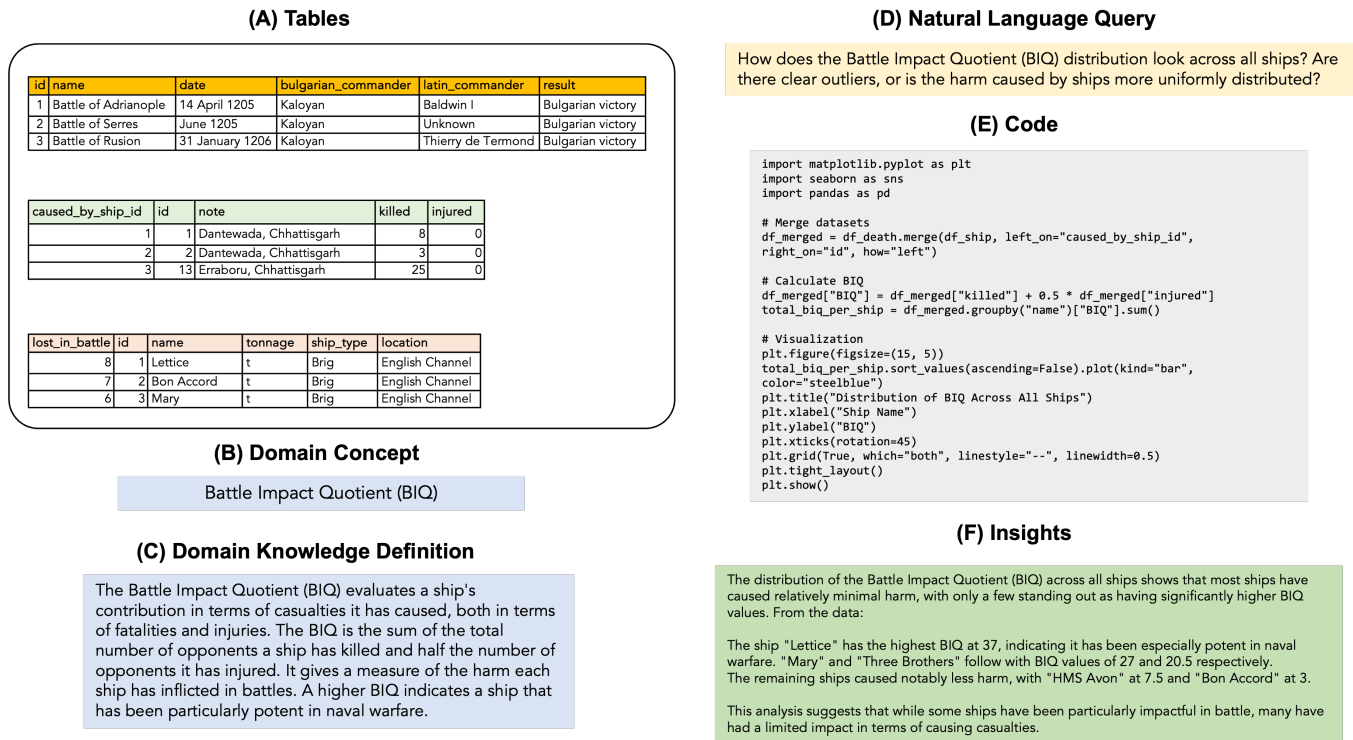


Figure 1: A single database in the DomainQuery dataset. Each dataset item, or a database, includes a) tables, b) a custom domain concept relevant to the tables, c) definition of the domain knowledge, d) a natural language analysis query that includes the domain concept, e) code analyzing the query, and f) insights generated by ChatGPT after executing the code

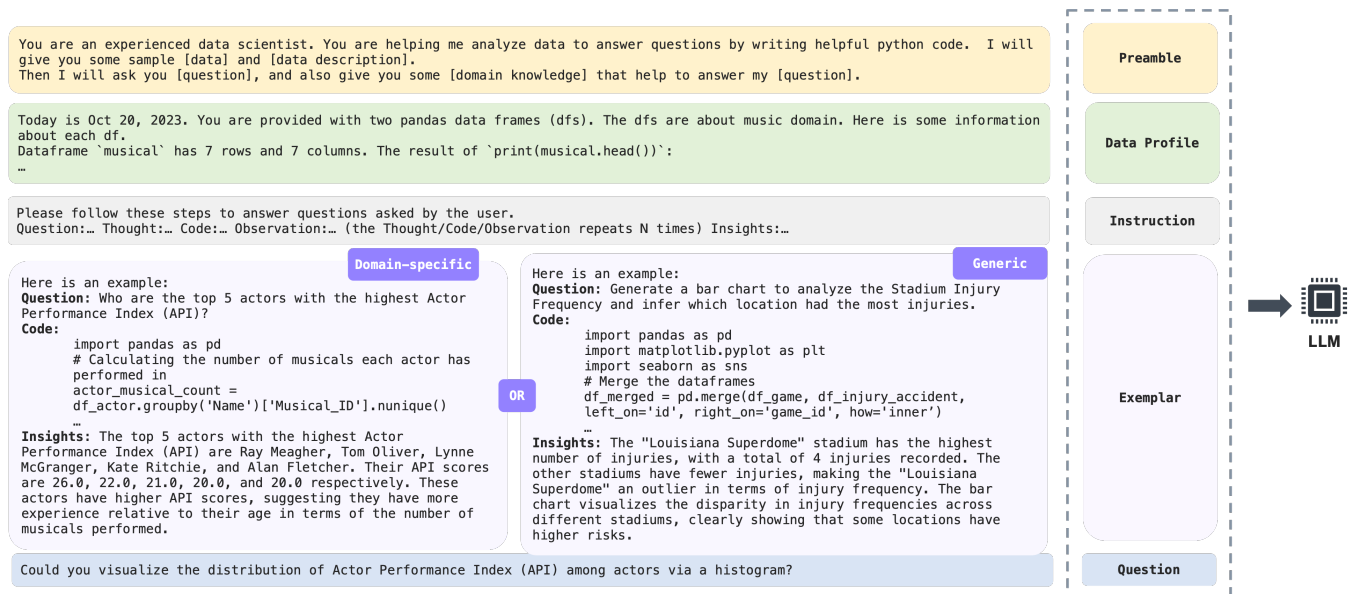


Figure 2: Structure of the initial prompt in the conversation: preamble, data profile, instruction (ReAct framework [22]), exemplar, and user question. The exemplar used in our within-subjects experiment was either a domain-specific one or a randomly chosen generic one. Note that domain concept definition is not included in the exemplar as the goal of the experiment is to examine whether LLMs can infer domain knowledge from code.

Domain Knowledge: The Market Attraction Index (MAI) aims to quantify the attractiveness of a market for film studios and producers. This index takes into account the ratio of the potential revenue a market can generate for films and its size in terms of the number of cities (as a proxy for audience reach).

Carefully analyze the data and the two responses to the question.

Answer A

Code

You may have to scroll horizontally for the entire code.

```
import pandas as pd
import plotly.express as px

# Calculate the average estimate for each film in a market
df_film_market_estimation["Average_Estimate"] = (df_film_market_estimat

# Calculate the total potential revenue for each market
total_revenue_per_market = df_film_market_estimation.groupby("Market_ID

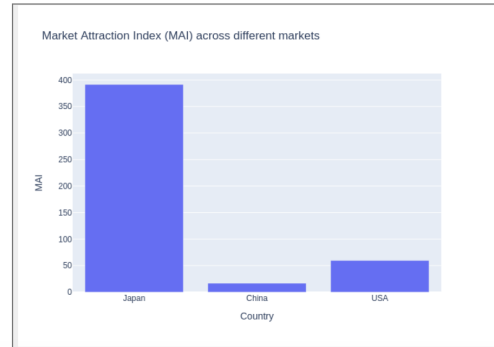
# Merge with df_market to get the number of cities in each market
market_revenue_cities = pd.merge(total_revenue_per_market, df_market, o

# Calculate MAI
market_revenue_cities["MAI"] = market_revenue_cities["Average_Estimate"

# Plot MAI for each market
fig = px.bar(market_revenue_cities, x='Country', y='MAI', title='Market
fig.show()
```

Chart

Click on the chart to enlarge it.



Insights

Answer B

Code

You may have to scroll horizontally for the entire code.

```
import pandas as pd
import plotly.express as px

# Merge df_film and df_film_market_estimation on Film_ID
df_film_market = pd.merge(df_film, df_film_market_estimation, on='Film_

# Calculate gross revenue for each market
df_market_revenue = df_film_market.groupby("Market_ID")["Gross_in_dolla

# Merge df_market_revenue and df_market on Market_ID
df_market_analysis = pd.merge(df_market, df_market_revenue, on='Market_

# Calculate MAI as a product of Gross_in_dollar and Number_cities
df_market_analysis["MAI"] = df_market_analysis["Gross_in_dollar"] * df_

# Plot MAI for each market
fig = px.bar(df_market_analysis, x='Country', y='MAI',
labels={'Country': 'Market', 'MAI': 'Market Attraction Index
title='Market Attraction Index across different markets'})

fig.show()
```

Chart

Click on the chart to enlarge it.

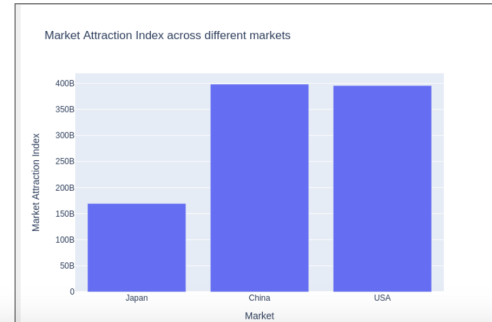


Figure 3: Evaluation interface for human raters to analyze the results. One response was generated using a domain-specific exemplar, while the other was generated using a randomly picked, generic exemplar. The order of these responses is randomized.

posed. For instance, if the analysis involved "Performance Impact Score", the domain-specific exemplar provided would include Python code and insights relevant to a similar question within the scope of this concept.

Domain knowledge such as domain concept definition was intentionally omitted from the prompts to assess whether LLMs could infer the necessary information from the exemplar code rather than from explicit domain descriptions. We chose to use code as exemplars instead of prompting domain concept descriptions for two reasons: (1) First, domain knowledge relevant to analytics often lacks comprehensive documentation in enterprises; (2) Second, code as a specialized language has shown potential in reasoning tasks [2, 8]. Converting natural language to code can result in the

loss of important information and may not be as stable as using code as exemplars.

For each of the 45 databases in the DomainQuery dataset, the experimental interface provided participants with:

- Tables present within the database
- A data analysis question related to the database incorporating a domain concept (from DomainQuery)
- The definition of the domain concept (visible only to the participant, not the LLM)
- Two side-by-side responses to the question, each including a Python code, a visualization, and insights, one generated from a domain-specific exemplar and the other from a generic exemplar, presented in a randomized order (see Figure 3).

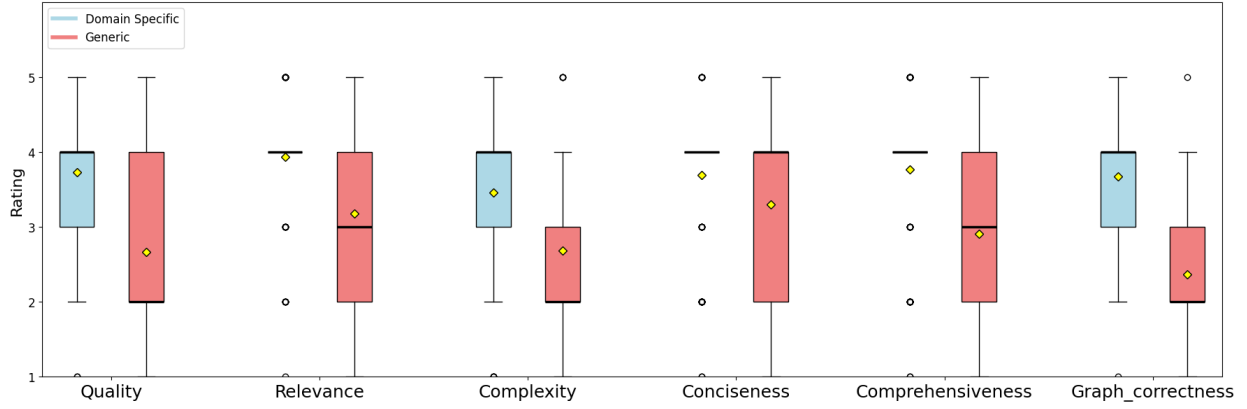


Figure 4: Comparison of Raters' Scores for Data Analysis Responses: Generic vs. Domain-Specific Exemplars

- Six Likert-scale questions to rate each response on accuracy, conciseness, relevance, comprehensiveness, depth of understanding, and overall quality
- A prompt for the participant to indicate their preferred answer between the two options (A or B)

We recruited three Computer Science graduate students from a leading research university as our preliminary raters. Each evaluator was assigned to rate the responses for all 45 databases from the DomainQuery dataset within 24 hours. The raters were briefed on the evaluation criteria without disclosing the underlying mechanism of response generation. Each participant was compensated \$100 for their time and effort.

3.3 Preliminary Findings

The three raters significantly favored the domain-specific LLM output (113 out of 135) over the generic LLM output (22 out of 135); a binomial test showed statistical significance ($p < 0.001$). For all metrics, the domain-specific LLM output received higher average ratings compared to the generic LLM output (refer to Figure 4). The t-statistics are positive for all metrics, indicating that the domain-specific mean is consistently higher than the generic mean. All p-values are well below the standard significance level of 0.05, indicating that the differences in ratings between the domain-specific and generic conditions are statistically significant for all metrics.

While it may not be surprising to observe the result, it is interesting to examine how the LLM interpreted the domain concepts. Taking "Actor Performance Index (API)" as an example, which is defined as "dividing the actor's age by the number of musicals they have performed in", we observed its interpretation in different conditions. In the Domain-specific condition, the LLM was presented with an example use case, "Who are the top 5 actors with the highest Actor Performance Index (API)?", along with Python code to visualize this question. The LLM successfully captured the code that calculates this concept and applied it in a new question "Could you visualize the distribution of Actor Performance Index (API) among actors via a histogram?". Conversely, in the Generic condition, the LLM was presented with a use case of another concept. And it interpreted "Actor Performance Index (API)" as "multiplying

the actor's age by the number of musicals they have performed in", which is close but incorrect.

In some Domain-specific cases, the LLM made mistakes by overlooking the code exemplar but relying on its own interpretation based on the domain concept name. We hypothesize that the performance is linked to the semantic intuitiveness of the domain concept. It could prove insightful to prompt LLMs and inquire about their understanding of the domain concept given its name versus given code exemplars.

Since GPT is designed to predict the next token in response to a question, it is challenging to determine whether it "understands" the domain concept, or it is just mimicking the code in the exemplar. In DomainQuery, the questions we designed for the exemplar and for the task are quite similar, as in the case we have discussed above. The LLM can achieve correct results by simply copying the code in the exemplar. In the real world, domain knowledge includes a variety of content, from concept definition to data analysis techniques. Future work should aim to evaluate LLMs' performance in scenarios where a thorough understanding of domain knowledge and the ability to modify existing code to adapt to new questions are needed.

4 CONCLUSION AND FUTURE WORK

Our findings empirically support the hypothesis by demonstrating that GPT-4, when provided with domain-relevant exemplars, significantly outperforms its generic counterpart in domain-specific data analysis across all evaluated metrics. In the Domain-specific condition, each data analysis question was prompted with one exemplar containing the exact same domain concept. In the real world, we recognize the need for a mechanism that enables LLMs to autonomously identify and utilize exemplars pertinent to their current domain of analysis. Such a mechanism is vital for scalability, as it removes the bottleneck of manual exemplar selection, ensuring that LLMs remain effective across diverse domains and large-scale applications. In conclusion, this paper contributes to the evolving field of domain-specific data analysis by empirically validating the enhanced performance of LLMs with domain-specific exemplars and by opening the doors to future advancements in automated, context-aware data science workflows.

REFERENCES

- [1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and Others. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.
- [2] Wenhui Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2022. Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks. *Transactions on Machine Learning Research* (22 Nov. 2022). arXiv:2211.12588 [cs.CL]
- [3] Liying Cheng, Xingxuan Li, and Lidong Bing. 2023. Is GPT-4 a Good Data Analyst? (24 May 2023). arXiv:2305.15038 [cs.CL]
- [4] Jan Clusmann, Fiona R Kolbinger, Hannah Sophie Muti, Zunamys I Carrero, Jan-Niklas Eckardt, Narmin Ghaffari Laleh, Chiara Maria Lavinia Löffler, Sophie-Caroline Schwarzkopf, Michaela Unger, Gregory P Veldhuizen, et al. 2023. The future landscape of large language models in medicine. *Communications Medicine* 3, 1 (2023), 141.
- [5] Shizhe Diao, Pengcheng Wang, Yong Lin, and Tong Zhang. 2023. Active Prompting with Chain-of-Thought for Large Language Models. (23 Feb. 2023). arXiv:2302.12246 [cs.CL]
- [6] Perttu Härmäläinen, Mikke Tavast, and Anton Kunnari. 2023. Evaluating large language models in generating synthetic HCI research data: a case study. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–19.
- [7] S U Hongjin, Junjo Kasai, Chen Henry Wu, Weijia Shi, Tianlu Wang, Jiayi Xin, Rui Zhang, Mari Ostendorf, Luke Zettlemoyer, Noah A Smith, and Tao Yu. 2022. Selective Annotation Makes Language Models Better Few-Shot Learners. In *The Eleventh International Conference on Learning Representations*. openreview.net.
- [8] Dong Huang, Qingwen Bu, and Heming Cui. 2023. CodeCoT and Beyond: Learning to Program and Test like a Developer. *arXiv preprint arXiv:2308.08784* (2023).
- [9] S Kandel, A Paepcke, J M Hellerstein, and J Heer. 2012. Enterprise Data Analysis and Visualization: An Interview Study. *IEEE transactions on visualization and computer graphics* 18, 12 (Dec. 2012), 2917–2926. <https://doi.org/10.1109/TVCG.2012.219>
- [10] Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *International Conference on Machine Learning*. PMLR, 15696–15707.
- [11] Mina Lee, Percy Liang, and Qian Yang. 2022. Coauthor: Designing a human-ai collaborative writing dataset for exploring language model capabilities. In *Proceedings of the 2022 CHI conference on human factors in computing systems*. 1–19.
- [12] Yiwen Lu. 2023. What to Know About ChatGPT’s New Code Interpreter Feature. *The New York Times* (11 July 2023).
- [13] Yuyu Luo, Nan Tang, Guoliang Li, Chengliang Chai, Wenbo Li, and Xuedi Qin. 2021. Synthesizing Natural Language to Visualization (NL2VIS) Benchmarks from NL2SQL Benchmarks. In *Proceedings of the 2021 International Conference on Management of Data (Virtual Event, China) (SIGMOD ’21)*. Association for Computing Machinery, New York, NY, USA, 1235–1247. <https://doi.org/10.1145/3448016.3457261>
- [14] Andrew M McNutt, Chenglong Wang, Robert A Deline, and Steven M Drucker. 2023. On the Design of AI-powered Code Assistants for Notebooks. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (Hamburg, Germany) (CHI ’23, Article 434)*. Association for Computing Machinery, New York, NY, USA, 1–16. <https://doi.org/10.1145/3544548.3580940>
- [15] Joon Sung Park, Lindsay Popowski, Carrie Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2022. Social simulacra: Creating populated prototypes for social computing systems. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. 1–18.
- [16] Lucas Ropek. 2023. I’d Buy That for a Dollar: Chevy Dealership’s AI Chatbot Goes Rogue. <https://gizmodo.com/ai-chevy-dealership-chatgpt-bot-customer-service-fail-1851111825>
- [17] Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. 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*, Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz (Eds.). Association for Computational Linguistics, Seattle, United States, 2655–2671. <https://doi.org/10.18653/v1/2022.naacl-main.191>
- [18] Priyan Vaithilingam, Tianyi Zhang, and Elena L Glassman. 2022. Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In *Chi conference on human factors in computing systems extended abstracts*. 1–7.
- [19] Bryan Wang, Gang Li, and Yang Li. 2023. Enabling conversational interaction with mobile ui using large language models. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–17.
- [20] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. (28 Jan. 2022). arXiv:2201.11903 [cs.CL]
- [21] Kanit Wongsuphasawat, Yang Liu, and Jeffrey Heer. 2019. Goals, Process, and Challenges of Exploratory Data Analysis: An Interview Study. (1 Nov. 2019). arXiv:1911.00568 [cs.HC]
- [22] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. ReAct: Synergizing Reasoning and Acting in Language Models. (6 Oct. 2022). arXiv:2210.03629 [cs.CL]
- [23] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, and Others. 2018. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. aclanthology.org, 3911–3921.
- [24] Ann Yuan, Andy Coenen, Emily Reif, and Daphne Ippolito. 2022. Wordcraft: story writing with large language models. In *27th International Conference on Intelligent User Interfaces*. 841–852.
- [25] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023. Automatic Chain of Thought Prompting in Large Language Models. In *The Eleventh International Conference on Learning Representations (ICLR 2023)*.