Natural Language Querying on Domain-Specific NoSQL Database with Large Language Models

Wenlong Zhang[1], Chengyang He[1], Guanqun Yang[1], Dipankar Bandyopadhyay[2], Tian Shi[3], Ping Wang[1]
[1]*Department of Computer Science, Stevens Institute of Technology, Hoboken, NJ USA*
[2]*Department of Biostatistics, Virginia Commonwealth University, Richmond, VA USA*
[3]*BizLidar, Parsippany, NJ USA*
{wzhang71, che14, gyang16, pwang44}@stevens.edu, dbandyop@vcu.edu, researchtianshi@gmail.com

***Abstract*—Efficiently and accurately retrieving specific information from healthcare datasets, such as the Vaccine Adverse Event Reporting System (VAERS)[1], presents significant challenges. A promising solution to this problem is the Text-to-ESQ approach, which is akin to Text-to-SQL tasks but leverages NoSQL database Elasticsearch, to thoroughly explore VAERS data. Non-relational databases are particularly adept at managing complex and dynamic data formats, thereby enabling the extraction of more valuable insights. However, generating executable NoSQL queries is still challenging due to the limited availability of NoSQL query datasets, which constrains model training. One potential remedy involves the use of large language models (LLMs), which can be applied in few-shot and even zero-shot learning scenarios. Nonetheless, the lack of prior evaluation for this novel task, coupled with the absence of a comprehensive, unbiased assessment of existing LLMs and prompting strategies, impedes the development of a robust architecture. Motivated by these challenges, we introduce a new Instruction-Enhanced Explainable (InstructEx) Chain-of-Thought (CoT) prompting by integrating existing CoT prompts and conducting a comprehensive investigation of LLMs and CoT prompting. The extensive experimental analysis demonstrates the effectiveness of using LLMs for Text-to-ESQ when combined with the InstructExCoT prompting. It also sheds light on the strengths and weaknesses of these methods from multiple perspectives.**

***Index Terms*—Natural language querying, NoSQL, Text-to-ESQ, VAERS, Elasticsearch query.**

I. Introduction

Natural language querying (NLQ) is an important task that can enhance the accessibility of specially tailored databases for broader audiences [1], [2]. Prior research predominantly addresses the Text-to-SQL tasks and only focuses on generation and modeling challenges within this domain [3], [4]. In fact, the functional limitation of SQL is one of the main reasons for the suboptimal results due to its limited capabilities in full-text searching and handling multifaceted information types. Additionally, SQL's design is inherently suited for static datasets and thus, it lacks proficiency in integrating external knowledge bases.

The NoSQL database fills this gap with its rapid response time and versatile handling of diverse data types within industrial applications. Recently, the task of NLQ on NoSQL database was initially proposed and explored on the Elasticsearch database (i.e. Text-to-ESQ task) by proposing a two-stage controllable framework along with the VAERSESQ dataset for training and validation [5]. However, compared to the industry, the academic utilization of NoSQL still presents several challenges that warrant further exploration. This study aims to address several primary concerns that limit the efficacy of Text-to-ESQ tasks. First, the reliance on Elasticsearch queries for data extraction is a significant barrier for researchers accustomed to the more universally applied SQL syntax. The diversity of Elasticsearch queries may cause further problems, such as different strategies when designing prompts for large language models (LLMs). Second, the lack of prior assessments for this innovative task underscores the need for a thorough investigation to formulate a viable solution. For executable queries, both keys and values contribute to forming a correct executable query. Therefore, it is critical to examine several important research questions, such as "What causes this negative outcome when we receive a negative query result?" and "Which part of the query is problematic?" There is a lack of detailed evaluation regarding the performance of this specific Text-to-ESQ task. Third, domain-specific language like Elasticsearch limits the availability of frameworks that can effectively adapt the database for specialized academic research tasks. This limitation restricts the potential for developing tailored solutions that can seamlessly integrate Elasticsearch into diverse research methodologies.

Recently, LLMs like Llama [6] and ChatGPT [7] have demonstrated significant potential in various natural language processing (NLP) tasks. Those methods offer an alternative approach to address existing challenges by applying specially designed prompts to guide the LLMs in completing Elasticsearch query generation without requiring extensive training data. Particularly, the prompt processing technique known as Chain of Thought (CoT) prompting, as introduced by [8], has enabled LLMs to handle more complex tasks like Text-to-ESQ. Despite numerous advancements, there are few fair evaluations across different tasks due to the diversity of tasks and datasets, making it challenging to apply these advanced methods to novel tasks. To address this, in this work, we systematically explore several CoT-style prompting strategies for the specific Text-to-ESQ task and examine their impact on generating different components of an executable query. Specifically, we propose a new instruction-enhanced explainable CoT prompting that integrates key elements from

[1]https://vaers.hhs.gov/

two widely used CoT prompting, including Instructional CoT [9] and Explainable CoT [10], to enhance the performance of LLMs for the Text-to-ESQ task. Our main contributions are summarized as follows:

- **Introduce the instructional-enhanced explainable (InstructEx) CoT prompting for the Text-to-ESQ task on NoSQL databases with LLMs**. To the best of our knowledge, this represents the first comprehensive exploration of LLMs applied to the NLQ task on the NoSQL database.
- **Conduct a comprehensive investigation of state-of-the-art LLMs and CoT prompting strategies**. Our study systematically explores CoT prompt engineering for the Text-to-ESQ task by incorporating nine LLMs and three types of baseline prompting strategies.
- **Conduct an extensive experimental analysis with both qualitative and quantitative results**. This analysis underscores the effectiveness of utilizing LLMs for the Text-to-ESQ task, particularly when combined with the newly proposed InstructEx CoT prompting, offering empirical evidence of how different methods contribute to enhancing model output.

II. Related Work

A. Natural Language Querying (NLQ)

NLQ is a critical task in the field of natural language processing (NLP), enabling users to interact with databases using natural language instead of structured query languages. Several types of natural language querying have emerged, each addressing different aspects of the task. Machine Reading Comprehension (MRC) focuses on extracting information from unstructured text by understanding and answering questions based on a given passage [11]. Knowledge Base Question Answering (KBQA) involves querying structured knowledge bases to provide precise answers to factual questions [12]. Text-to-SQL translates natural language queries into SQL commands to interact with relational databases [1]. Recently, the Text-to-ESQ task has been proposed to facilitate more accurate and efficient exploration of complex healthcare information with a NoSQL database [5]. This creates a new research direction for NLQ on NoSQL database and will significantly benefit the retrieval of complex information from non-relational databases. Meanwhile, a new dataset, VAERSESQ, is also proposed for querying and retrieving health information from the VAERS data in Elasticsearch database [5]. Despite its potential, this area remains largely underexplored. In this work, we aim to fill this gap and explore the power of LLMs for the NLQ task on the NoSQL databases. Our objective is to leverage the advantage of the NoSQL dataset in complex healthcare contents and enhance the performance and usability of NLQ systems by addressing the unique challenges posed by NoSQL databases. By leveraging the strengths of both LLMs and non-relational querying approaches, we seek to provide a more robust and versatile solution for Text-to-ESQ tasks in healthcare fields.

B. LLMs and CoT Prompting

In recent years, LLMs have revolutionized the field of NLP by achieving unprecedented performance on a variety of linguistic tasks. The evolution of LLMs, from models like OpenAI's GPT-2 [13] to the more recent GPT-4o [7], has demonstrated their increasing capability to understand and generate coherent and contextually relevant text across diverse domains. One of the significant breakthroughs of LLMs is their ability to perform zero-shot, one-shot, and few-shot learning, as highlighted in the work on GPT-3 [14]. This capability allows these models to generalize from minimal task-specific examples, making them highly versatile and reducing the need for extensive labeled datasets. Despite their impressive capabilities, there are still several ways to further improve the performance of these models. One of the most significant methods is Chain-of-Thought (CoT) prompting. It is designed to enhance the reasoning capabilities of LLMs by breaking down complex problems into a series of intermediate steps [15]. Now it has become a pivotal strategy in advancing AI reasoning. Based on CoT, there came a lot of different kinds of CoT-style branches. Part of them we call *Instructional CoT*, which means using instructions only to guide LLMs to decompose complex problems. For example, in [16], the API DOCs prompt is used this way for the Text-to-SQL task. The other part of them we call *Explainable CoT* since they use normal words instead of instructions to explain every step. Least-to-Most [17] is one of the successful examples among them. However, both methods have their limitations, which we will discuss further in Section 3. Therefore, in an LLM with limited input, it is crucial to balance the two prompts and leverage each of their strengths.

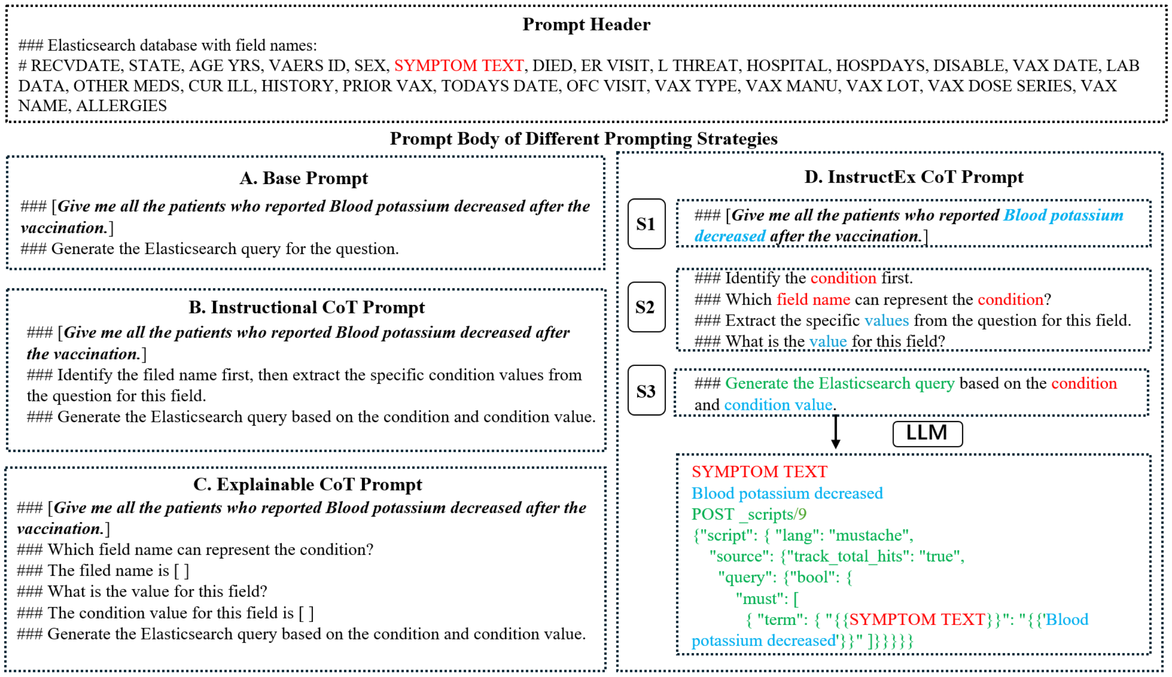
C. Domain Specific Language (DSL) Generation

DSL generation is an important field in NLP that focuses on generating text or code tailored to specific industries or fields. This specialized approach contrasts with general language models by incorporating domain-specific knowledge and terminology, resulting in more accurate and relevant outputs. The key challenge in DSL lies in the lack of training datasets and how to apply them to the language models pre-trained by domain knowledge. We explore the potential of LLMs for DSL generation tasks, specifically the Text-to-ESQ task. Unlike previous methods that require an extensive pre-training process, LLMs utilize prompt engineering and in-context learning to generate Elasticsearch queries. Additionally, the use of specially designed prompts enables LLMs to better adapt to the requirements of Elasticsearch queries, greatly enhancing model performance.

III. Instruction-Enhanced Explainable CoT Prompting

A. Preliminaries

Most CoT prompts can be categorized into two categories, including Instructional CoT [9] and Explainable CoT [10]. Our newly designed Instruction-Enhanced CoT prompt combines the fundamental elements of both categories.



Prompt Header

Elasticsearch database with field names:
RECVDATE, STATE, AGE YRS, VAERS ID, SEX, SYMPTOM TEXT, DIED, ER VISIT, L THREAT, HOSPITAL, HOSPDAYS, DISABLE, VAX DATE, LAB DATA, OTHER MEDS, CUR ILL, HISTORY, PRIOR VAX, TODAYS DATE, OFC VISIT, VAX TYPE, VAX MANU, VAX LOT, VAX DOSE SERIES, VAX NAME, ALLERGIES

Prompt Body of Different Prompting Strategies

A. Base Prompt

[***Give me all the patients who reported Blood potassium decreased after the vaccination.***]
Generate the Elasticsearch query for the question.

B. Instructional CoT Prompt

[***Give me all the patients who reported Blood potassium decreased after the vaccination.***]
Identify the filed name first, then extract the specific condition values from the question for this field.
Generate the Elasticsearch query based on the condition and condition value.

C. Explainable CoT Prompt

[***Give me all the patients who reported Blood potassium decreased after the vaccination.***]
Which field name can represent the condition?
The filed name is [ ]
What is the value for this field?
The condition value for this field is [ ]
Generate the Elasticsearch query based on the condition and condition value.

D. InstructEx CoT Prompt

S1
[***Give me all the patients who reported Blood potassium decreased after the vaccination.***]

S2
Identify the condition first.
Which field name can represent the condition?
Extract the specific values from the question for this field.
What is the value for this field?

S3
Generate the Elasticsearch query based on the condition and condition value.

LLM

```
SYMPTOM TEXT
Blood potassium decreased
POST _scripts/9
{"script": { "lang": "mustache",
   "source": {"track_total_hits": "true",
     "query": {"bool": {
        "must": [
          { "term": { "{{SYMPTOM TEXT}}": "{{'Blood
potassium decreased'}}" ]}}}}}
```

Fig. 1: The overall structure of the InstructEx CoT prompting compared with three base prompts. Colors show the correspondence between various components in the question, prompt, and generated Elasticsearch queries. Here, "track_total_hits" indicates the query will retrieve all the relevant records, and "term" indicates a specific field to search, which takes variables provided in the "must" section to query. The "query" sections are designed for specific options to improve retrieval accuracy.

1) Instructional CoT Prompt: Instructional CoT prompt emphasizes providing only the essential instructions required to guide the LLMs effectively [18] and has been adopted in many NLP tasks such as question answering and sentiment analysis [19]–[21]. Clear instructions help minimize misleading, particularly in areas with limited pre-training data. Most LLMs are pre-trained on data in the general domain, with only a small portion dedicated to specialized tasks. This imbalance often results in suboptimal performance when these models are applied to domain-specific tasks. In this context, designing appropriate prompts becomes crucial, as misleading prompts can easily result in incorrect outcomes. Another advantage of Instructional CoT prompt is its brevity, which minimizes the length of the prompt and creates more space for in-context learning examples. This efficiency is particularly beneficial in tasks with constrained answer formats, such as named entity recognition [22] and code generation [23]. By focusing on key points, Instructional CoT ensures a focused and directed interaction with the LLMs, thereby enhancing the model's performance in tasks requiring precise outputs. However, providing only necessary instructions is insufficient to address complex and domain-specific problems, particularly those involving logical reasoning.

2) Explainable CoT Prompt: Explainable CoT is the other category of most widely used prompting approaches that emulate human societal interactions by guiding the model through problem-solving steps and incorporating extensive explanations [24]. This strategy closely aligns with human cognitive processes, enhancing the intuitiveness and user-friendliness of interactions with language models. Moreover, a sufficiently explanatory prompt can help the model focus on logical connections, thereby enhancing the retrieval of similar logical information. Empirical evidence supporting the efficacy of this approach is presented in studies such as [25], [26], where a range of studies consistently demonstrate improved performance when employing the Explainable CoT method. Compared to Instructional CoT prompts, this method excels in handling complex logic problems, such as content with structure, where the rules of explanation are more important than in-context examples. However, this approach still has its drawbacks, such as consuming a significant number of input tokens due to too many explanation words and potentially misleading the model's results through the explanation process.

B. InstructEx CoT Prompting Design

Many complex problems, for example, Text-to-ESQ tasks in this work, tend to involve both intricate logic judgments and domain-specific language explanation. When handling these tasks with LLMs, it is essential to explain the necessary rules in the prompts and ensure that the generated results are not misleading. Therefore, in this paper, we combine the advantages of both Instructional and Explainable CoT prompts to address these challenges and introduce the Instruction-Enhanced Explainable (InstructEx) CoT prompt. Specifically, the InstructEx CoT prompt represents a hybrid approach that integrates elements from both Instructional CoT and Explainable CoT prompts. Figure 1 provides the overall prompt structure of different prompting strategies with an illustrative example. We focus on zero-shot learning in this paper.

1) Prompt Formats: In the development of the InstructEx prompt architecture, we drew inspiration from two established prompt formats, including API Documentation [16] and QDecomp [20]. Each of these formats exhibits distinct advantages and is suitable for various sub-tasks. First, the QDecomp format is adopted for the basic structure design, as shown in Figure 1, which separates the prompts into two parts: Prompt header and Prompt body. Further, we apply the API DOCs format to depict all rest prompts through Python API comments. This deliberated design minimizes the length of prompts for each instance, thereby providing sufficient space for additional in-context examples within a constrained input space [20]. This method significantly differs from the Explainable CoT prompts, which typically require more verbose descriptions. Such extensive descriptions can potentially lead to ambiguities or misinterpretations when processed by large language models. For example, when using the Explainable CoT prompts in our experiments with the Falcon model, the employment of general prompts resulted in the generation of SQL queries as opposed to the intended Elasticsearch queries. Therefore, the API Docs format has been one of our primary choices for enhancing efficiency and accuracy in solving this complex Text-to-ESQ task.

2) The Design of InstructEx CoT Prompting: Unlike traditional CoT prompts that separate tasks by the logic chain, InstructEx CoT emphasizes breaking down the task itself. Specifically, we adopted a modified version of the CoT prompt style to deconstruct complex tasks into more manageable subtasks. This strategy departs from the linear, step-by-step logic commonly employed in solving mathematical problems. Instead, it embraces a task-based thinking paradigm, which focuses on the holistic understanding and resolution of the problem rather than following a sequential logical progression. In general, as shown in Figure 1, the InstructEx prompt includes two essential components, the prompt header and the prompt body.

Prompt header include the database schema at the beginning of the entire prompt which encapsulates essential information, establishing a contextual foundation for subsequent prompts. Compared to the task of generating codes directly from a guide model, the task-specific header enhances the generation of the crucial condition fields. Additionally, it reduces the prompt length for each example compared to other similar methods. This component will also be adopted in the design of all baseline prompts for a fair comparison in our experiments.

Prompt body is designed to offer more comprehensive guidance, clarifying specific tasks and expectations. The additional information will guide the model behavior and help improve the accuracy of the generated query. The prompt body consists of three key sub-components.

- ***Natural Language Question (S1)***: The input questions will be provided at the beginning of the prompt body. Square brackets are used to highlight questions. This basic sub-component is also adopted in other different baseline prompts. Excessive decomposition can mislead the model's output direction. To avoid this, like most CoT methods, it's beneficial to separate the input questions from the decomposition processes, ensuring the intended goal remains clear.
- ***Instructions with Stepwise Explanations (S2)***: This step is a critical sub-component of the InstructEx CoT prompt. We employ a task-specific strategy to break down the complex task of generating an Elasticsearch query into two manageable steps, including (1) identifying the condition words related to field names listed in the prompt header, and (2) identifying the corresponding values with conditions. Each step is initially articulated through an ***instructional-style prompt*** to avoid any ambiguity, then followed by an ***explanatory expression*** to guide the accurate extraction. By combining these two strategies, we leverage their respective advantages and achieve significantly improved results.
- ***Query Generation Instruction (S3)***: This sub-component aims to combine all information extracted from the previous stages and guide the model to complete the query generation. We provide this sub-component at the end of the prompt to clarify the task's purpose and ensure the generation of the expected results.

The deliberately designed process culminates with an instructional prompt, designed to synthesize the various components into a cohesive Elasticsearch query. Our new structured approach blends directive and conversational elements, giving new direction in prompt design for complex query generation tasks, especially in DSL-generating fields like Text-to-ESQ. All the prompting strategies in experiments utilize the same structure for fair comparisons.

C. Investigated LLMs

We evaluate the performance of the proposed InstructEx prompt by comparing it with three baseline prompts with multiple state-of-the-art LLMs, Including GPT series (gpt-3.5-turbo, gpt-4-1106, and gpt-4o) [27], Llama series (Llama2, Llama3, LlamaChat, and CodeLlama) [6], Falcon[2], and StarCoder [28].

IV. Experiments

In this section, we address the following research questions (RQs) through experimental analysis. (1) **RQ1**: How do various LLMs perform on the domain-specific Text-to-ESQ task? (2) **RQ2**: Does the proposed InstructExCoT prompting improve the performance of the Text-to-ESQ task? (3) **RQ3**: What is the performance of different LLMs on various data types in the Elasticsearch queries? (4) **RQ4**: How do we interpret the generated queries obtained from the LLMs with different prompting strategies?

A. Experimental Settings

1) Datasets Used: We utilize the VAERSESQ dataset released in [5] for our experimental analysis on the Text-to-ESQ task. Specifically, the Elasticsearch database is created based on the publically available *Vaccine Adverse Event Reporting*

[2]https://falconllm.tii.ae/falcon-models.html

TABLE I: Basic statistics of VAERSESQ dataset.

Data	Value
Number of tables	3
Number of fields/columns in tables	35/8/11
Number of questions	13,040
Average question length (in words)	12.13
Average query length	167.65

System (VAERS) dataset[3]. VAERS covers patients' demographics, diagnoses, a standardized list of symptoms, and additional context related to vaccine adverse events. It also provides details on vaccine categorization and manufacturing. VAERS is designed to detect potential safety issues with vaccines and ensure that the benefits outweigh the risks.

To test and evaluate different approaches for Text-to-ESQ, we utilize the question-query pairs from the VAERSESQ Dataset, which is specifically created based on three different tables in the VAERS data. Table I provides the basic statistics of the VAERSESQ data. The dataset covers 13,040 questions, with an average question length of 12.13 words. The average query length is 167.65, and each Elasticsearch query is associated with a natural language question obtained by rephrasing a template question. For our experimentation, we randomly selected a sample of 1,000 instances for analysis.

2) Evaluation Metrics: We mainly utilize three metrics to evaluate the generated Elasticsearch queries. (1) **BLEU Score** [29] is originally designed for assessing machine-translated text against human benchmarks and has expanded its utility to various NLP tasks, including code generation. In the context of this DSL, popular metrics like CodeBLEU are less applicable due to a lack of a trained matrix specifically tailored for DSLs. (2) **Query Frame Similarity** measured by Jaccard Distance is used to evaluate the fuzzy similarity between the ground truth and generated Elasticsearch queries. This is particularly relevant in scenarios where multiple correct Elasticsearch queries might exist for a single question. Unlike exact matching, fuzzy similarity prevents the omission of the generated queries that can accurately extract information despite some minor differences from the ground truth. (3) **Query keyword accuracy** measured by **condition match** and **value match** between the ground truth and generated queries. Here, "*condition*" means where to extract information, specifically indicating the column or field name the question is related to. While "*value*" represents the specific data cell retrieved from the database. This allows us to further verify the accuracy of the generated query by focusing on the two critical components in Elasticsearch queries, including the searching field and the searching value.

3) Implementation Details: All experiments except for the GPT series were implemented using NVIDIA Quadro RTX 5000 GPUs. The implementation of all LLMs was performed with CUDA 12.0 and Ubuntu 22.04.3, ensuring efficient parallel processing and optimization on the GPU hardware. Other dependencies include PyTorch version 2.0.1 and Transformers version 4.34.0. For the GPT series, we use the OpenAI API in our experiments. To balance the generation quality and diversity, the temperature parameter was set to 0.7. Additionally, a repetition penalty of 1.1 was applied to discourage models from generating repetitive sequences, thus enhancing the variety and informativeness of the outputs. By maintaining these consistent experimental settings, we ensured a fair comparison across different models and prompting strategies. The implementation of the InstructExis made publically available at this GitHub repository[4].

B. Experimental Results

*1) Query Generation Performance of Different LLMs (**RQ1**):* Table II and Table III provide the performance comparison of different LLMs with different prompting strategies for Text-to-ESQ. Table III demonstrate that our proposed method surpasses baseline approaches, achieving higher overall BLEU scores, which indicate improved code structure and accuracy relative to the ground truth. The significant enhancement observed in GPT-4o underscores its heightened sensitivity to prompts and its ability to adapt effectively to complex tasks, with our method further augmenting the model's performance. In Table II, it can be observed that, for each specific prompting strategy, gpt-4o demonstrates a superior and notable performance on this zero-shot DSL learning task compared with other LLMs, which is evident in two key aspects. First, its excellence in generating accurate condition values in Elasticsearch queries is particularly noteworthy, leading to high scores in terms of value match. Second, gpt-4o's superior code frame generation capabilities are further highlighted by its leading performance in Frame Similarity, surpassing all other examined LLMs. Besides LLMs in the GPT series, Llama series also demonstrates effective performance across different evaluation metrics. Specifically, Llama3 shows improvements over Llama2 in all three evaluation metrics, despite utilizing a smaller model size of 8 billion parameters compared to Llama2's 13 billion. While Llama2 exceeds LlamaChat in code framework proficiency, its performance diverges significantly concerning the evaluation metrics of Condition match and Value match. Additionally, StarCoder demonstrates strong performance in generating the Value field, but its performance in other areas is lacking.

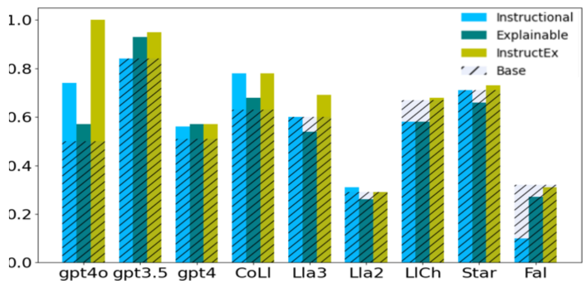
Our analysis indicated that not all LLMs can adapt well to Text-to-ESQ task. Models such as Llamachat and Falcon exhibit limited proficiency in generating Elasticsearch queries, showing only marginal improvements observed after prompt tuning. In Table II, it is important to highlight that although CodeLlama operates with significantly fewer parameters (13 billion), it still exhibits competitive performance, especially after applying InstructExprompt method. Notably, it surpasses GPT series in Condition Match, achieving an accuracy of 0.520, which highlights CodeLlama's proficiency in extracting complexly formulated information.

[3]https://vaers.hhs.gov/

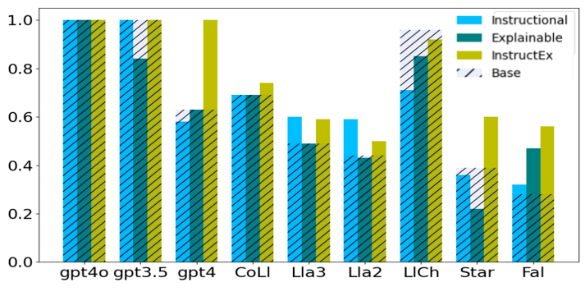
[4]https://github.com/SteveZwl/LLM-for-Text-to-ESQ

TABLE II: The Text-to-ESQ task performance of different LLMs with four prompting strategies. The underline indicates the best result in the same method separately and we bold the best results on different evaluation matrices.

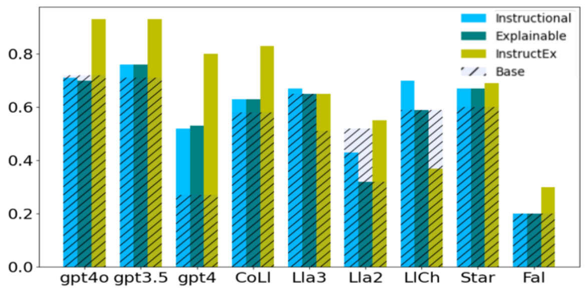
Prompt	Evaluation	gpt-4o	gpt-4	gpt-3.5	Llama2	Llama3	LlamaChat	CodeLlama	Falcon	StarCoder
Base	Frame similarity	0.453	0.328	0.367	0.289	0.257	0.141	0.312	0.234	0.140
	Condition match	0.427	0.345	0.412	0.105	0.213	0.192	0.472	0.235	0.392
	Value match	0.745	0.519	0.898	0.375	0.529	0.670	0.578	0.379	0.592
Instructional	Frame similarity	0.421	0.335	0.351	0.242	0.213	0.172	0.362	0.218	0.180
	Condition match	0.435	0.401	0.420	0.112	0.186	0.202	0.506	0.397	0.400
	Value match	0.867	0.611	0.961	0.407	0.629	0.667	0.788	0.166	0.615
Explainable	Frame similarity	0.454	0.421	0.383	0.313	0.273	0.148	0.356	0.328	0.320
	Condition match	0.381	0.368	0.376	0.093	0.316	0.202	0.376	0.110	0.190
	Value match	0.768	0.509	0.932	0.453	0.590	0.653	0.455	0.344	0.543
InstructEx	Frame similarity	**0.472**	0.382	0.461	0.281	0.287	0.226	0.352	0.341	0.328
	Condition match	0.477	0.422	0.432	0.150	0.396	0.196	**0.520**	0.236	0.303
	Value match	**0.985**	0.704	0.973	0.482	0.623	0.676	0.720	0.406	0.720



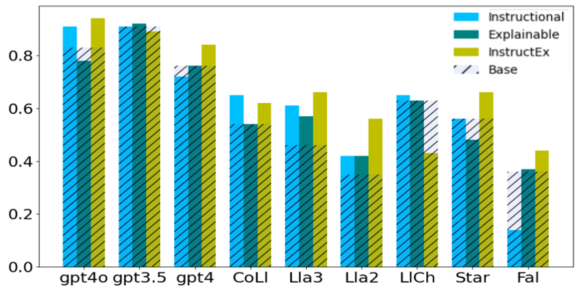
(a) Date Format



(b) Float Format



(c) Serial Format



(d) Text Format

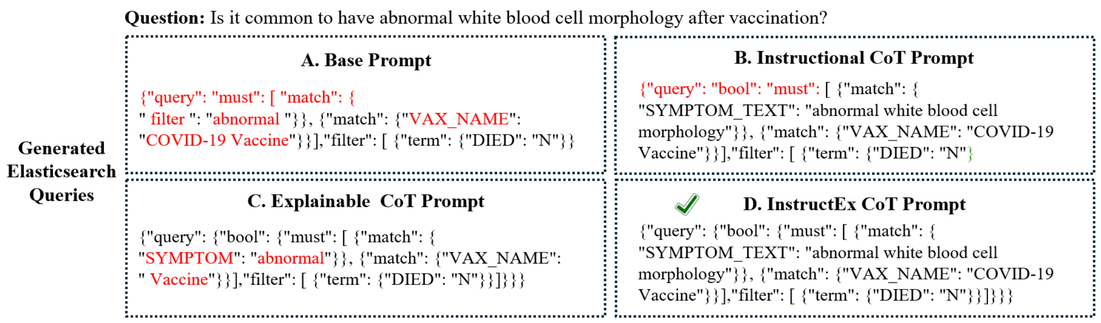
Fig. 2: Performance on generating different types of condition values in Elasticsearch queries. The x-axis labels represent LLMs, including gpt4o (gpt-4o), gpt4 (gpt-4-1106-preview), gpt3.5 (gpt-3.5-turbo), Fal (Falcon), Star (StarCoder), Lla (Llama2), Llch (LlamaChat), and Llco (CodeLlama).

TABLE III: Performance measured by BLEU Score for the Text-to-ESQ task.

LLMs	Base	Instructional	Explainable	InstructEx
gpt-4o	32.0	33.4	35.0	**45.4**
gpt-3.5-turbo	35.0	35.5	35.3	42.4
gpt-4	24.0	30.7	34.3	38.9
CodeLlama	26.5	34.0	37.8	39.4
Llama2	10.5	38.0	37.0	38.5
Llama3	30.5	30.0	28.4	33.0
LlamaChat	10.3	24.0	28.4	32.5
StarCoder	7.0	32.0	38.6	39.8
Falcon	17.0	6.0	26.0	17.6

*2) Performance of Different Prompting Strategies (**RQ2**):* Table II also provides a comprehensive comparative study across four prompting strategies for Text-to-ESQ tasks with different LLMs. In general, compared with the base prompt, we observe that all specifically designed prompting strategies demonstrated an enhanced performance for all LLMs, particularly notable in the context of code framework, as evidenced by the scores of Code Frame Similarity. Among different prompting strategies, the use of Instructional CoT prompts specifically improved model performance in terms of special keywords, notably in the Condition and Value segments. However, when employing Explainable CoT prompts, there was a noticeable improvement in the code framework, albeit with diminished effectiveness in handling special keywords.

Our analysis reveals that both Explainable and InstructExCoT prompting significantly enhance the Code Frame Similarity Score, indicating their effectiveness in aiding LLMs to generate more accurate code frames. Notably, InstructExCoT prompts often outperform Explainable CoT prompts in this regard. However, the scenario alters when examining the keyword accuracy, and only Instructional and InstructExCoT prompts show a positive contribution to keyword accuracy. This suggests that these methods are more adept



Question: Is it common to have abnormal white blood cell morphology after vaccination?

A. Base Prompt

```
{"query": "must": [ "match": {
" filter ": "abnormal "}}, {"match": {"VAX_NAME":
"COVID-19 Vaccine"}}],"filter": [ {"term": {"DIED": "N"}}
```

B. Instructional CoT Prompt

```
{"query": "bool": "must": [ {"match": {
"SYMPTOM_TEXT": "abnormal white blood cell
morphology"}}, {"match": {"VAX_NAME": "COVID-19
Vaccine"}}],"filter": [ {"term": {"DIED": "N"}
```

C. Explainable CoT Prompt

```
{"query": {"bool": {"must": [ {"match": {
"SYMPTOM": "abnormal"}}, {"match": {"VAX_NAME":
" Vaccine"}}],"filter": [ {"term": {"DIED": "N"}}]}}}
```

✓ **D. InstructEx CoT Prompt**

```
{"query": {"bool": {"must": [ {"match": {
"SYMPTOM_TEXT": "abnormal white blood cell
morphology"}}, {"match": {"VAX_NAME": "COVID-19
Vaccine"}}],"filter": [ {"term": {"DIED": "N"}}]}}}
```

Fig. 3: An illustrative example of generated queries from different prompts. Red color indicates the generation errors.

at addressing specific format information within generated content than Explainable methods. Conversely, Explainable prompts appear to have a detrimental effect in this context. The overall performance metrics corroborate the efficacy of our approach InstructExCoT prompts in amalgamating these distinct prompt styles, thereby augmenting model performance across various dimensions. Based on our findings, we can summarize the advantages of Instructional and Explainable CoT prompting. (1) *Instructional CoT* guides the model to ignore the relationships between different parts of a sentence and instead focus on the accuracy of key values that positively impact the keywords, including conditions and values part in generating code, but it decreases the performance of Code Frame Similarity. Specifically, Instructional CoT prompting significantly enhances the performance of Value match accuracy, gpt-4o observes a 22% increment compared to the Base prompt. (2) *Explainable CoT* enhances the code frame but reduces the accuracy of the keyword match. Compared to concise SQL queries, Elasticsearch queries are more like a scripting language, and have complex terms for expressiveness, which means they contain more relatively detailed content. Additionally, Explainable CoT prompts perform better in generating this relatively detailed content, which is why Explainable CoT contributes more significantly to the Code Frame. Therefore, it is a trade-off strategy when given the limited input tokens for LLMs, and we have balanced both Instructional and Explainable CoT prompts in the proposed InstructExprompting, which has significantly enhanced model efficacy and demonstrated substantial positive effects across various aspects for Text-to-ESQ. The findings also highlight the critical importance of designing task-specific prompting for LLMs, tailored to the unique characteristics and capabilities of each model.

*3) Performance on Generating Different Types of Condition Values in Elasticsearch Queries (**RQ3**):* Figure 2 shows the performance of different LLMs with four prompting strategies for generating different data types, including Date, Floating-point numbers, Serial Numbers for vaccine product codes, and Texts in the generated queries. GPT series generally demonstrate the best performance across almost all data types. For the Llama series, CodeLlama and Llama3 show significantly more accurate judgments for date types compared with Llama2 and LlamaChat. LlamaChat excels in predicting floating-point numbers but is considerably less accurate in the remaining categories. Falcon and Llama2 exhibit mediocre performance across all domains. StarCoder outperforms CodeLlama in Serial Format and plain text but falls short in the other categories. In conclusion, the GPT series consistently achieves the highest overall evaluation. However, the Llama series demonstrates a strong cost-performance ratio, particularly in this experiment, where we utilized the smaller parameter versions due to computational resource limitations. Although other models may not match the power of the GPT series, they still offer valuable potential applications. For instance, StarCoder excels in information extraction, making it a viable solution in this domain. Similarly, CodeLlama can serve as an alternative to GPT series for some simple tasks and is also more cost-effective, demonstrating its utility in specific scenarios.

In Figure 2, we also observed that InstructExCoT prompting yielded the most positive impact across all data formats. Notably, significant improvements were seen in the Serial and Text Format, which are considered the most distinct aspects of the generation task. Generally, models with a higher number of parameters experienced greater enhancement following the application of the InstructExprompts. Conversely, the implementation of Explainable CoT prompts exhibited a negative influence on the Value match field, particularly in the Date and Float formats. This suggests that the inclusion of explainable words may hinder the extraction of purely numerical information in this context.

*4) Case Study with Different Prompting Strategies (**RQ4**):* Figure 3 provides the generated queries with different prompting strategies for a specific question example. It can be observed that all baseline prompts, except for the InstructExCoT prompt, exhibited errors in the generation. Specifically, the base prompt demonstrated a code frame error, characterized by the absence of crucial content and the presence of multiple brackets. The Instructional CoT prompts, on the other hand, contained two code frame errors, both attributable to missing brackets. In the scenario involving the Explainable CoT prompt, two value match errors and one condition match error were identified. Contrarily, the InstructExCoT prompts accurately yielded the correct results, effectively addressing the example question regarding patients who exhibited abnormal white blood cell counts post-COVID vaccination and remained alive. In the segment of code frames, a predominant source of errors is attributed to the omission of brackets. Furthermore, within the keyword section, it is observed that the condition aspect is more prone to errors compared to the value segment. The reason is that condition fields include some specific symbols and unordered letters. Therefore, the key issues include the selection of an unsuitable condition field and the failure to fully encapsulate the condition name.

V. Conclusions

In this paper, we employ large language models (LLMs) for natural language querying (NLQ) on NoSQL databases to address the challenges of retrieving complex health information. We systematically examine Chain-of-Thought (CoT) style prompting to enhance LLMs' capabilities in the Text-to-ESQ task. Specifically, we integrate reasoning steps from two existing methods (Instructional CoT and Explainable CoT), and further introduce a novel approach for generating Elasticsearch queries in the Text-to-ESQ task. Our extensive experiments demonstrate that: (1) different models vary in performance when generating distinct parts of the queries, and (2) different prompt styles impact multiple data types differently. The introduction of the InstructExCoT prompting marks an initial exploration of merging diverse prompt styles for solving the Text-to-ESQ task on NoSQL databases with LLMs.

Acknowledgment

This work was supported in part by the US National Science Foundation grant IIS-2245907 and the startup funding from the Stevens Institute of Technology.

References

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

[2] H. Kim, B.-H. So, W.-S. Han, and H. Lee, "Natural language to sql: where are we today?" *Proceedings of the VLDB Endowment*, vol. 13, no. 10, pp. 1737–1750, 2020.

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

[4] T. Scholak, N. Schucher, and D. Bahdanau, "Picard: Parsing incrementally for constrained auto-regressive decoding from language models," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 9895–9901.

[5] W. Zhang, K. Zeng, X. Yang, T. Shi, and P. Wang, "Text-to-esq: A two-stage controllable approach for efficient retrieval of vaccine adverse events from nosql database," in *Proceedings of the 14th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, 2023, pp. 1–10.

[6] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[7] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[8] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," 2023.

[9] J. Zhang, B. Wang, L. Li, Y. Nakashima, and H. Nagahara, "Instruct me more! random prompting for visual in-context learning," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, January 2024, pp. 2597–2606.

[10] Y.-F. Yeh, M.-C. Chen, P.-H. Hung, and G.-J. Hwang, "Optimal self-explanation prompt design in dynamic multi-representational learning environments," *Computers & Education*, vol. 54, no. 4, pp. 1089–1100, 2010.

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

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

[13] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[14] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[15] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[16] N. Rajkumar, R. Li, and D. Bahdanau, "Evaluating the text-to-sql capabilities of large language models," *arXiv preprint arXiv:2204.00498*, 2022.

[17] D. Zhou, N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, C. Cui, O. Bousquet, Q. Le *et al.*, "Least-to-most prompting enables complex reasoning in large language models," *arXiv preprint arXiv:2205.10625*, 2022.

[18] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–35, 2023.

[19] B. Paranjape, S. Lundberg, S. Singh, H. Hajishirzi, L. Zettlemoyer, and M. T. Ribeiro, "Art: Automatic multi-step reasoning and tool-use for large language models," 2023.

[20] C.-Y. Tai, Z. Chen, T. Zhang, X. Deng, and H. Sun, "Exploring chain-of-thought style prompting for text-to-sql," 2023.

[21] Y. Shen, K. Song, X. Tan, D. Li, W. Lu, and Y. Zhuang, "Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face," 2023.

[22] X. Ren, Y. Song, C. Yan, Y. Xiong, F. Kong, and X. Fu, "Cmed-baichuan: Task explanation-enhanced prompt method on promptcblue benchmark," in *Health Information Processing. Evaluation Track Papers*, H. Xu, Q. Chen, H. Lin, F. Wu, L. Liu, B. Tang, T. Hao, Z. Huang, J. Lei, Z. Li, and H. Zong, Eds. Singapore: Springer Nature Singapore, 2024, pp. 31–48.

[23] Z. Liu, Y. Tang, X. Luo, Y. Zhou, and L. F. Zhang, "No need to lift a finger anymore? assessing the quality of code generation by chatgpt," *IEEE Transactions on Software Engineering*, vol. 50, no. 6, pp. 1548–1584, 2024.

[24] K. Berthold, H. Röder, D. Knörzer, W. Kessler, and A. Renkl, "The double-edged effects of explanation prompts," *Computers in Human Behavior*, vol. 27, no. 1, pp. 69–75, 2011.

[25] O. Press, M. Zhang, S. Min, L. Schmidt, N. A. Smith, and M. Lewis, "Measuring and narrowing the compositionality gap in language models," 2023.

[26] Z. Zhang, A. Zhang, M. Li, and A. Smola, "Automatic chain of thought prompting in large language models," 2022.

[27] L. Gao, T. D. la Tour, H. Tillman, G. Goh, R. Troll, A. Radford, I. Sutskever, J. Leike, and J. Wu, "Scaling and evaluating sparse autoencoders," *arXiv preprint arXiv:2406.04093*, 2024.

[28] R. Li, L. B. Allal, Y. Zi, N. Muennighoff, D. Kocetkov, C. Mou, M. Marone, C. Akiki, J. Li, J. Chim *et al.*, "Starcoder: may the source be with you!" *arXiv preprint arXiv:2305.06161*, 2023.

[29] M. Post, "A call for clarity in reporting bleu scores," *arXiv preprint arXiv:1804.08771*, 2018.