Can LLMs Answer Investment Banking Questions? Using Domain-Tuned Functions to Improve LLM Performance on Knowledge-Intensive Analytical Tasks

Nicholas Harvel, Felipe Bivort Haiek, Anupriya Ankolekar, David James Brunner
ModuleQ, Inc.
10080 N. Wolfe Rd., Suite SW3-200
Cupertino, CA 95014
{nicholas.harvel, felipe.bivort, anupriya, djb}@moduleq.com

Abstract

Large Language Models (LLMs) can increase the productivity of general-purpose knowledge work, but accuracy is a concern, especially in professional settings requiring domain-specific knowledge and reasoning. To evaluate the suitability of LLMs for such work, we developed a benchmark of 16 analytical tasks representative of the investment banking industry. We evaluated LLM performance without special prompting, with relevant information provided in the prompt, and as part of a system giving the LLM access to domain-tuned functions for information retrieval and planning. Without access to functions, state-of-the-art LLMs performed poorly, completing two or fewer tasks correctly. Access to appropriate domain-tuned functions yielded dramatically better results, although performance was highly sensitive to the design of the functions and the structure of the information they returned. The most effective designs yielded correct answers on 12 out of 16 tasks. Our results suggest that domain-specific functions and information structures, by empowering LLMs with relevant domain knowledge and enabling them to reason in domain-appropriate ways, may be a powerful means of adapting LLMs for use in demanding professional settings.

Introduction

With their prodigious memory, flexible natural language reasoning and ease of verbal expression, Large Language Models (LLMs) show great promise for question-answering across a variety of domains. They have been integrated into widely-used search tools such as Google, with Search Generative Experience (SGE) and Microsoft Bing. Significant effort and resources have been expended to apply them to general knowledge work (Microsoft Copilot) as well as specific professional domains including financial services (BloombergGPT (Wu et al. 2023)), software engineering (Github CoPilot, Google Duet), and law (Casetext), to name prominent examples.

Recent research showed that LLMs can dramatically increase the productivity of sophisticated professionals (Dell'Acqua et al. 2023). However, the same research found that reliance on LLMs may increase the risk of inaccurate answers. In professional settings, inaccurate answers can cause financial damage, legal liability, and other adverse consequences. Relatively little is understood about how to effectively utilize LLMs to support professionals with accurate, up-to-date, trustworthy and reliable answers in specific professional industries.

Investment bankers advising on corporate mergers and acquisitions are an exemplary case of knowledge-intensive professionals whose judgments, often made under intense time pressure, may have massive financial impacts. Investment bankers depend on accurate, current, and comprehensive information about their clients and competitors, industry trends, capital markets activity, and analyst opinions in order to provide strategic advice and structure corporate transactions. Investment banks have developed their own processes, methods and models to operate in this complex, high-stakes and fast-moving domain. To support dealmaking and provide strategic advice, senior bankers rely on analysts to gather extensive current and historical information about clients and companies in their coverage sectors. Analysts make use of existing internal models to value companies, and they monitor industry trends from news and external information sources.

To be useful as a question-answering tool in an investment banking setting, an LLM must be able to provide accurate and complete answers. This would likely entail analyzing multiple structured and unstructured, often noisy, sources; cross-referencing and comparing data; working with sophisticated valuation models; and interpreting all of this information from the perspective of an investment banker. Research is needed to evaluate LLM performance on such types of knowledge-intensive professional tasks that require domain-specific reasoning and knowledge.

LLMs face several well-known limitations. To begin with, not all knowledge stored in LLMs can be easily and reliably elicited (Su et al. 2023) and LLM awareness of recent events is limited by their "knowledge cutoff", i.e. the date at which the training data was acquired. Relevant information provided in the prompt context has been shown to help LLMs generate more accurate and grounded responses (Ram et al. 2023; Xu et al. 2023) without requiring the LLM to be retrained or fine-tuned. Nevertheless, we find that LLM-enhanced search engines, such as Microsoft Bing and Google SGE, struggle to give accurate answers to questions about recent high-profile mergers and acquisitions even when citing sources containing the relevant information.

The need for accurate answers has led to substantial effort in developing retrieval-augmented generation (RAG), methods to increase and optimize the context window, e.g. using function calling (Packer et al. 2023), and prompt engineering methods, such as chain-of-thought (CoT) reasoning, to improve LLM performance on questions about structured data in tables and graphs (Sui et al. 2024; Guo et al. 2023). While these show great potential, it remains unclear how well LLMs can utilize contextual information and domain knowledge for professional question-answering.

In this paper, we address this gap by proposing a compact benchmark of data and associated question-answering tasks for investment banking. Using this benchmark, we evaluate LLM performance, both without special prompting and with a number of existing methods for augmenting LLMs with domain knowledge. In particular, we examine the effect on LLM performance of different prompt methods and different variants of knowledge presentation. We focus on function calling as an effective foundational method to enable LLMs to perform domain-appropriate reasoning by integrating domain knowledge sources and quantitative models.

Our results show that enabling LLMs to use Chain-of-Thought (CoT) reasoning with function calls to retrieve relevant semi-structured information yields the best performance. Examining LLM errors, we find that their performance is affected by the way the functions are designed and the way contextual knowledge is presented. This indicates a need for careful design and engineering of functional APIs for LLMs and effective knowledge structures to enable LLMs to support professional question-answering.

Related Work

In-Context Learning with LLMs LLMs, such as GPT-3.5 and GPT-4, are flexible few-shot reasoners in natural language tasks. Relevant information provided in the prompt context has been shown to help LLMs generate more accurate and grounded responses (Ram et al. 2023; Xu et al. 2023) without requiring the LLM to be retrained or fine-tuned.

Prompt context can extend LLM internal memory and ground LLM responses on pre-specified information, improving the accuracy and reliability of LLM output. As all LLMs face limits on context size, various methods have been explored to increase the size of the context window without retraining the model (Pal et al. 2023).

Unfortunately, LLMs do not attend to the entirety of the context equally and tend to focus more on information located at the beginning and end of the context (Liu et al. 2023). Thus, increasing context window length tends to degrade LLM performance in utilizing pertinent information buried in the middle of the context.

Instead of increasing the size of the context, another approach is to select and possibly summarize relevant information for inclusion in the context while respecting context limits (Wang et al. 2023). Prior work (Xu et al. 2023) has demonstrated the value of well-chosen context, as in retrieval-augmented generation (RAG) (Ram et al. 2023), as being more effective than simply increasing the context window of LLMs for long context tasks such as query-based summarization and document question-answering.

Function Calling An alternative technique to increasing context window size is to use the emerging function calling capability[1]. This enables LLMs to fetch additional context via function calls to external APIs. Function calls have been used to enable LLMs to utilize different types of context far beyond the context window limit (Packer et al. 2023). This indicates an intriguing ability of LLMs to independently formulate and compose function calls to retrieve relevant contextual information for question-answering. In our investigation, we rely heavily on function-based prompting and probe LLM ability to effective utilize available external functions.

Prompt Engineering Several methods have been explored to improve LLM performance on natural language tasks. A popular technique using LLMs over 100B parameters is Chain-of-Thought (CoT) prompting (Wei et al. 2022). CoT has been shown to encourage structured reasoning by LLMs and significantly improve outcomes in numerous tasks, including entity recognition (Ashok and Lipton 2023), mathematical problem-solving and multi-step reasoning tasks (Liu and Tan 2023; Ziqi and Lu 2023) using function calls. There are multiple versions of CoT, ranging from zero-shot prompting ("Let's think step by step" (Wei et al. 2022)) to few-shot versions, where several examples are provided for the LLM to consult when constructing its response. Our investigation focuses on the former.

Text-to-SQL In answering factual queries on structured data, the use of function calls is akin to the task of writing SQL queries to retrieve and manipulate data in relational tables. There has been significant effort to develop effective few-shot prompting techniques for LLMs to convert natural language questions into SQL queries (Gao et al. 2023), ranging from improving instructions, including schema information, providing valid sample queries, consistency checking over multiple rounds of prompting and LLM review-and-refine iterations. To our knowledge, text-to-SQL methods have not yet informed the design of function-based prompting. We examine the performance of text-to-SQL methods and function-based prompting inspired by them on our evaluation set.

Benchmarks for Querying Structured Data Several evaluation benchmarks exist for querying and extraction of structured information, e.g. Spider (Yu et al. 2018), Unite (Lan et al. 2023), FinLMEval (Guo, Xu, and Yang 2023) and SUC (Sui et al. 2024). These benchmarks focus on the tabular manipulation of data and do not probe LLMs on domain-specific questions, e.g. comparing different labels for the same industry. Spider-Syn (Gan et al. 2021) does include cases where an entity name in a question must be matched against a different value in a field; however, it does not consider one-to-many mappings that are commonly assumed in professional speech. For example, encryption software and cloud security companies can arguably be consid-

[1]Currently available for OpenAI LLMs GPT-3.5 and GPT-4, Llama 2 and Anthropic's Claude.

ered as companies in the cybersecurity business by an investment banker. We include such questions in our benchmark.

Method

We investigate LLM performance on professional question-answering by developing a dataset relevant to investment bankers and a set of associated questions that represent typical information needs of investment bankers. We then evaluate LLM performance on this set of questions (often referred to as *tasks* in the literature) with and without additional contextual knowledge, and examine the impact of different ways to present contextual knowledge, and of various prompting methods to procure and process the additional information.

Benchmark

Tasks To examine the abilities of LLMs on typical questions by investment bankers, we developed several questions referring to various types of corporate transactions, e.g. mergers and acquisitions, by Microsoft and its subsidiaries between 1986 to the present day. These questions were informed by interviews with investment bankers from a medium-size boutique investment banking firm in the US and attempt to represent the types of questions that a senior banker may ask an analyst when evaluating a deal. The questions[2] are categorized in Table 1 in terms of the abilities they probe, namely:

Date ranges: These investigate LLM comprehension and manipulation of dates and time periods. E.g. years ("in 2019", "2020 through 2022" or "since 2010") or monthly periods (such as "calendar Q3 of 2015", "in the twelve months beginning January 1, 2015"). Some date ranges are implicit, e.g. "since the acquisition of X".

Arithmetic: These require the ability to count, rank, round and compare. E.g. they request the "value of acquisitions ... to the nearest billion", "largest acquisitions by value" or request a comparison of numbers, e.g. "how much more [was spent] on acquisitions" and "does the total known value of acquisitions exceed 1T$".

Multi-step: Questions requiring multiple steps to produce an answer. These are often comparison questions, but also include questions such as "In the year of the Github acquisition, ..." requiring an LLM to first resolve the year of the Github acquisition before proceeding to answer the rest of the question.

Semantics (of a domain): Such questions assume domain knowledge of investment industry sectors and the meaning of acquisition, e.g. "which video games businesses" or "which companies were acquired by Microsoft's Github subsidiary".

Open-ended questions: These questions probe domain comprehension and require a reasoned explanation, e.g. around the impact of events on companies or peer/subsidiary connections between companies.

Geographic questions: These questions require an understanding of terrestrial geography, relating e.g. to "European companies", "Israel-based companies", "headquarters outside the US".

[2]The full set of questions is available on request.

Data The dataset is a table of acquisitions and mergers, stake purchases and divestitures by Microsoft, drawn from the corresponding Wikipedia page[3]. It contains the date, transaction value (if known), name of the acquired company, the industry sector or business of the acquired company, the country where the acquired company is headquartered, and a list of news articles and press releases announcing the acquisition. The press releases typically mention the same information in unstructured form and enable the comparison of structured and unstructured information presentation on LLM responses.

The data was scraped from Wikipedia using Selenium and Beautiful Soup. There are two parts to the data: tabular and links to textual data. The tabular data was pre-processed and saved in CSV format. Links to external resources (news and press releases) were followed, parsed and the resulting text constituted the unstructured component of the data set. In cases where the press releases were unavailable or the links were dead, we manually supplemented the dataset with similar press releases obtained by web search.

Experiments

Our experiments investigate the effect on LLM performance of different prompt methods and different forms of knowledge presentation. We test LLM performance on a baseline of no additional context, context provided within a prompt and on-demand via function calling. Given the existence of several questions requiring multi-step computations, we also investigate combining function calls with CoT reasoning and alternative usage of an intermediate SQL engine.

The different variants of knowledge presentation we test are: unstructured natural language text (*Text*), i.e. news and press releases only, (b) structured, tabular data (*Table*), i.e. the extracted Wikipedia table, and (c) a mix of unstructured and structured data (*Combined*). The *Combined* case consists of data from *Table* and *Text* conditions.

One question specifically examines the ability of LLMs to extract data present only in unstructured form, i.e. to determine the acquirer of a company from the press release. It is therefore excluded from the *Table* condition.

Baseline In this condition, we assess the performance of OpenAI GPT-4 without any additional context. Microsoft Bing and Google SGE have access to Internet search results. For these systems, we lightly modified questions as needed to enable these systems to respond accurately and to the best of their knowledge.

Prompt All relevant context was placed in the prompt, namely the relevant rows from the table, including references (URLs) and their text. The current date and time was provided at the end of the prompt in this and all the following conditions.

Function In the *Function* conditions, the LLM is provided with functions to retrieve additional information, effectively increasing its context window. It is instructed to use the functions to retrieve more accurate, up-to-date information and to make as many function calls as it likes.

[3]https://en.wikipedia.org/wiki/List_of_mergers_and_acquisitions_by_Microsoft

Question Type	Count	Sample Questions
Date ranges	15	What are the names of the companies acquired by Microsoft in the twelve months beginning January 1, 2015?
Multi-step	12	How much more did Microsoft spend on acquisitions from 2016 to 2020 compared to 2020 to 2015? In the 2 years prior to the Nemesys Games acquisition by Microsoft, which companies with headquarters outside the US were also acquired by the same company? Has Microsoft taken a stake in any company at least 2 years before acquiring it?
Arithmetic	10	What were Microsoft's three largest acquisitions by value, 2020 through 2022, according to public sources?
Semantics	6	Since 2010, in which year did Microsoft make the most cybersecurity acquisitions? What are the names of the companies acquired by Microsoft's GitHub subsidiary in 2019?
Open-ended	3	Wang Laboratories has declared bankruptcy. What impact will this have on Microsoft?
Geography	3	How many European companies did Microsoft acquire in 2016?

Table 1: A categorization of the 21 evaluation tasks in terms of the comprehension and reasoning abilities they require. See the Benchmarks section for a more detailed explanation of the categories.

Figure 1 shows the process used for iterative function-based prompting. For each user question, an LLM prompt is assembled with base and concluding instructions, the user question itself, and a (initially empty) history buffer consisting of prior LLM responses and the results of the last function call. The LLM invocation is augmented with an array of possible functions it can call, each specified with a function signature and a textual description of the function and its parameters. The LLM determines the data it requires to answer the question and formulates a function call (if needed) to request that data. The response to the function call, e.g. the extracted data, is then appended to the original prompt and resent as a new prompt to the LLM.

This condition initially specifies only two functions: *Extract Data* to retrieve data with an optional start or end date and by acquisition target name, i.e. the name of the company being acquired; and the *Respond* function to simply return an answer to the user. Other conditions add more functions to the menu, as described below.

Note that it is not sufficient for the LLM to simply formulate a function call. Although the functions are domain-specific, they are general enough that they will typically not return a direct answer to a question. The LLM must additionally review the data returned, extract any relevant information, and analyse it to construct an answer to the original question. In addition, depending on the formulation, the functions could either reduce the volume of information that the LLM must analyze or return a large volume of irrelevant data that may overwhelm the LLM context. Finally, the LLM typically needs to string together function calls in varying ways to answer questions. Hence, this condition, like the following ones, is designed to test the ability of LLMs to identify a correct sequence of function calls to make, intelligently formulate function calls and use function call output to construct answers to user questions.

Function + CoT This condition extends the previous one by allowing the LLM to call an additional function *Plan* to capture a plan of action on how to answer the query. The prompt instructions are modified to instruct the LLM to first outline this execution plan and only then proceed to actually execute the plan. The *Plan* function allows the LLM to review information received from prior function calls (if any) and determine next steps—possibly deviating from its initial plan—without messaging the user.

SQL In this condition, the *Extract Data* function is replaced by a call to an SQL engine. The new function requires a valid SQL query (for a given SQL engine) as a parameter. In order to create usable SQL queries, the LLM is provided with a schema of an SQL table in JSON format within the prompt. The table enables the querying of data by date ranges, by acquired companies and by the country in which an acquired company is headquartered. This condition uses a pared-down variation of the *Function* prompt, modified to include a couple of hints to improve SQL query formulation and avoid common failure modes.

SQL + Semantic Match This condition was introduced upon observing the LLM's strong preference for using the SQL 'LIKE' operator to answer questions requiring domain understanding (*Semantics*), rather than using its innate world knowledge, which led to several missed opportunities. To nudge the LLM into identifying semantically similar phrases, an additional function *Semantic Match* was included. The LLM was forced to use this function to identify all values in a table column that are similar to a given phrase. This enabled a company providing a 'gaming backend service' to be included as a result when searching for 'video games' businesses.

Function + CoT + SQL This condition applies prompting methods from the text-to-SQL literature in two key changes: (1) the table schema is included in JSON format within the prompt (with minor renaming of columns) and (2) an additional function is added to enable the LLM to select individual columns in the data, thus permitting the retrieval of more targeted and less noisy data.

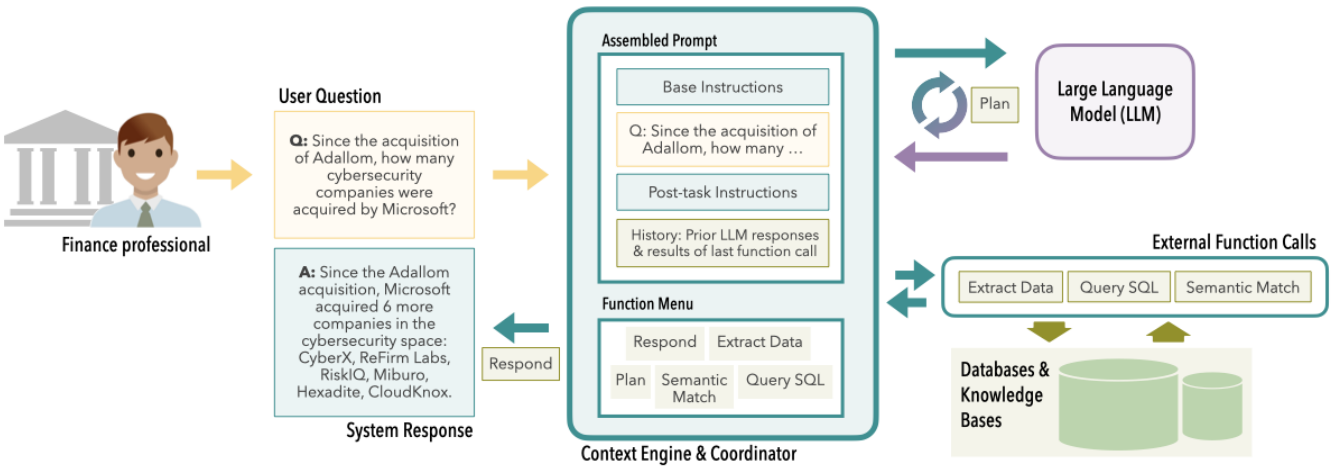

Figure 1: High-level diagram of our function-based prompting system for the following prompting methods: [Function], [Function + CoT], [SQL], [SQL + Semantic Match] and [Function + CoT + SQL]. The Method section describes which functions are available to each prompting method.

Models Given the niche domain and relatively complex questions, we use GPT-4, specifically gpt-4-0613[4], as representing the best of current LLM capabilities.

Evaluation LLM performance is assessed primarily on the basis of accuracy (correctness and completeness) of answers and, to a secondary degree, on the relevance of the information provided. While accuracy in paramount in a professional setting, users are often able to compensate for a system's deficits and still make use of partially relevant information.

Results

Table 2 summarizes the overall results of LLM performance in terms of the number of correct, partially correct and incorrect responses to questions depending on the prompting method and the form of contextual information provided.

For questions with lists of answers (e.g. company names), we further analyze the partially correct responses and classify them into answers that were correct, missed and wrong (Table 3). If the LLM failed to provide an answer in the ground truth, then that was considered to be a *missed* answer. If the LLM provided an answer that was not present in the ground truth, we considered it a *wrong* answer.

Baseline The baseline experiments with Base LLM (GPT-4), Bing and SGE do not perform well. Of 16 questions, each system gets only one question completely correct. For Base LLM and Bing, the remaining questions are almost evenly divided between partially correct and wholly incorrect responses. Google SGE seems to prefer descriptive responses at times and seems to perform relatively worse.

Prompt Performance is only marginally improved when relevant context is pasted into the prompt. The number of correct responses rises to 2, but the number of incorrect responses remains high at 10, with 4 partially correct responses.

Function In comparison with the previous conditions, the ability to make function calls has a significant positive effect on LLM performance. LLMs are able to make surprisingly effective use of function calls: identifying the need for additional data and formulating queries based on the function signature and documentation. Performance is also considerably improved by the presence of contextual information in concise, structured form.

In other words, when provided with additional relevant information purely in unstructured, natural language form, the LLM performs only mildly better than in the *Prompt* condition. However, changing the form of the relevant information to either being completely structured or to be a mix of structured and unstructured data, the LLM is able to perform significantly better. This suggests that although LLMs are able to process unstructured and structured content, they can more effectively utilize concise, well-structured information.

The effect of the form of contextual information is even clearer when further examining the partially correct answers in Table 3. Textual information caused the highest proportion of missed answers, suggesting the reduced LLM performance may be due to the lower information density of unstructured content. Structured information approximately halves the proportion of missed answers and significantly increases the proportion of correctly identified answers.

Function + CoT Chain-of-thought (CoT) reasoning has been shown to improve step-by-step reasoning in LLMs. Our results also show that the LLM makes fewer mistakes when using function calls if it is explicitly encouraged to formulate a plan of action beforehand. While this does not improve its performance overall in Table 2 significantly, it does

[4]with model parameters: `temperature` set to 0.5, `top_p` to 0.95.

Context Design		Results			
Form	Condition	Correct	Partially Correct	Incorrect	Tasks
None	Base LLM	1	6	9	16
	Microsoft Bing	1	8	7	
	Google SGE	0	4	12	
Text	Function	3	5	8	16
	Function + CoT	3	8	5	
Table	Function	8	4	3	15
	Function + CoT	8	3	4	
Combined	Prompt	2	4	10	16
	Function	10	3	3	16
	Function + CoT	9	3	4	
	Function + CoT + SQL	**12**	**1**	**3**	
	SQL	10	1	5	16
	SQL + Semantic Match	**12**	**1**	**3**	

Table 2: Results of LLM performance in terms of number of Correct, Partially Correct and Incorrect answers depending on prompting method and the form of contextual information provided. We use the most frequent response in three runs as the representative response. Not all tasks were appropriate for all forms of information, hence the number of "Tasks" is also noted in the table. The best-performing prompting methods are highlighted in bold.

markedly change the nature of its partially correct answers, especially with structured information. As shown in Table 3, the combination of CoT reasoning and structured information (as in the *Table* and *Combined* conditions) leads to considerably fewer missed and wrong answers, while boosting the proportion of correct answers.

SQL and SQL + Semantic Match LLM performance is further improved with Text-to-SQL prompt techniques, e.g. by including the table schema in the context and providing informative column names. Table 3 further highlights the value of using a query engine on structured data for question-answering: while this method missed a few answers, it never resulted in a wrong answer in our testing.

It is noteworthy that the LLM needed to be forced to use an explicit semantic matching function in order to find similar businesses in an industry, even though it could have done so itself. In fact, the Semantic Match function merely performs an independent call to an LLM requesting a semantic match. After receiving the results of an SQL query, the LLM would not process them any further and would simply return the results. This reluctance only manifests in processing the output of an SQL query; the LLM manages to successfully leverage its internal knowledge when creating queries. For example, the LLM successfully answered a question about European companies by first creating a list of all European countries and then creating an SQL query to check for the presence of any country in the table in that list.

Function + CoT + SQL By using techniques inspired from text-to-SQL prompting and combining them with CoT reasoning and function calls, comparable performance to pure text-to-SQL can be achieved. In this condition, the table schema was included in the prompt and the LLM was allowed to choose the fields of data it wished to retrieve. These two changes significantly improve the performance of CoT reasoning with function calls (Table 2). Examining the results in Table 3, we see that it is the best-performing method, leading to the smallest proportion of missed answers, very few wrong answers and the most correct answers.

Noise As a side investigation, we also examined the effect of the presence of irrelevant information on LLM performance by presenting it with information about all acquisitions, stakes and divestitures, not just a relevant subset of the information. The presence of noise appears to only slightly degrade LLM performance and the results are generally inconclusive. We see this as an area for future research.

Performance by Question Type

Arithmetic and Date Range Calculations Arithmetic and date computations are known to be challenging for LLMs. As expected, many numerical questions and date computations were incorrectly answered in *Baseline* conditions. That said, the LLM seemed to have less difficulty in formulating function calls with date ranges, both relative and absolute. The function calls typically had correct date parameters, with relatively few mistakes. LLMs have sporadic formatting inconsistency in their responses. During our experiments, this also manifested itself with inconsistent date formatting, e.g. reverting to timestamps instead of ISO dates, and consequently causing the extraction function to choke intermittently.

Multi-step Computations These remain challenging for LLMs. Although the LLM is often able to formulate a reasonable plan with CoT reasoning, it regularly fails to execute all components of its plan when dealing with complex queries. In this case, SQL-inspired methods were more suc-

Context Design		Results			
Form	Condition	Correct	Missed	Wrong	Total Incorrect
Text	Function	0.56	0.44	0.07	0.52
	Function + CoT	0.56	0.44	0.22	0.52
Table	Function	0.78	0.22	0.02	0.24
	Function + CoT	0.85	0.15	0.07	0.22
Combined	Function	0.78	0.22	0.19	0.41
	Function + CoT	**0.93**	0.07	0.07	0.15
	Function + CoT + SQL	**0.95**	**0.05**	**0.04**	**0.09**
	SQL	0.80	0.20	**0.00**	0.20
	SQL + Semantic Match	0.91	0.09	**0.00**	**0.09**

Table 3: An analysis of LLM performance on questions with lists as answers. For such questions, an incomplete answer may be better than no answer; however, wrong answers are clearly undesirable. We characterize LLM performance in terms of the proportion of total list elements correctly identified, missed and wrongly identified. The total incorrect proportion is simply the sum of the missed and wrong proportions. The best-performing prompting method and results are highlighted in bold.

cessful, as the LLM could sometimes encode a multi-step plan in the SQL query and then outsource the execution to an SQL engine. For example, when answering the question "How much more did Microsoft spend on acquisitions from 2016 to 2020 compared to 2020 to 2015?", the answer can be encoded as the result of an SQL query.

Domain Knowledge & Semantics In answering questions that require (relatively shallow) domain knowledge, the LLM was able to find good and unexpected answers. For example, when asked about "cybersecurity", it identified companies in the data protection and data governance business. Similarly, it correctly identified a "gaming backend service" and an "industrial AI platform" as companies related to "cloud computing". As discussed previously, these were more difficult with the SQL function-based prompting methods.

Open-Ended Open-ended questions were the most challenging. Although they could be answered with some real-world knowledge and the retrieval of appropriate information from the table, no method we tested induced the LLM to answer them successfully.

Geography The geographic components of questions posed no problems whatsoever for the LLMs, suggesting they are adept in reasoning about geography and have extensive knowledge of cities, countries and larger geographic regions.

Discussion

Our results show that, without special prompting, a state-of-the-art LLM and major LLM-enhanced search engines are generally unable to answer accurately the kinds of straightforward questions that investment bankers would expect their analysts to handle with ease. Adding relevant information to the prompt, as in a naive RAG implementation, is helpful, but inadequate. Importantly, we find that domain-tuned function calls can have a transformative impact on LLM performance, increasing accuracy from at most 12.5% to as high as 75%, and providing a method for LLMs to incorporate domain-specific knowledge and reasoning. While these results still fall short of the accuracy demanded in professional settings such as investment banking, the magnitude of the improvement suggests that function calls are a promising direction for research into applying LLMs in such task domains.

An interesting side effect of using functions is that the function calls generated by the LLM provide some insight into the LLM's reasoning process. This led to several intriguing observations in the course of our experiments, summarized below.

Function-induced bias: When functions are available, LLMs seem to prefer to use the functions, even without being explicitly instructed to do so. Moreover, functions appear to prime reasoning in particular ways. When asked to identify the acquisitions made by Github in a given year and provided with a function allowing for retrieval of additional data about acquired companies, the LLM frequently attempts to retrieve data for the Github acquisition by Microsoft, rather than acquisitions by Github. The correct approach would have been to retrieve data about all acquisitions in the given year, and then to read through the text of press releases to identify the ones made by Github. When the questions are about acquired companies, and thus match the function signature provided, the LLM performs very well.

Unexpected function behavior: We observed a different type of error when function behavior did not match the LLM's expectation. When the LLM attempts to answer a question about acquisitions within a range of years, the LLM invokes the function with the correct start and end dates. However, the underlying function (in Python) excludes the end of the data range. This causes an entire year of data to be omitted from the results. Although the incomplete results are appended to the prompt, subsequent LLM reasoning does not spot the problem and blithely continues processing the data. This problem was solved by instructing the LLM to specify dates completely, including the day and month.

Inefficient and incomplete reasoning: The LLM does not

always invoke function calls efficiently, despite instructions to do so. For example, to answer the question "Since 2010, in which year did Microsoft make the most cybersecurity acquisitions?", rather than retrieving only acquisitions since 2010, the LLM tends to query for all transactions and then filter to those satisfying the date filter. This occurs even in the CoT condition, and would clearly be very inefficient when accessing a large database. Similarly, without additional prompting, the LLM often stops short of performing the final step in an execution plan. For example, when using SQL to fetch press releases, it may subsequently forget to examine them for answers, or when asked for the total acquisition expenditure in a given year, it fetches the numbers, but subsequently fails to sum them up.

These observations highlight the sensitivity of LLM processing to the way that functions and data are made available to the LLM. Thoroughly-documented APIs are likely important, as well as having a range of multiple domain-tuned functions to choose from. Performance improvements could be achieved by presenting the same underlying functionality in multiple forms, with varying descriptions, signatures, and return values, so as to enable the LLM to better incorporate the functionality in different types of analysis. For example, multiple functions for calculating specific variants of financial ratios might map to the same underlying logic, but instead of relying on the LLM to perform conversions, having functions with exactly matching signatures could yield more reliable reasoning.

In real-world applications, LLMs may benefit from access to an extensive set of domain-tuned functions. Even if all these functions were well-documented and potentially accessible to the LLM, making every function available in every LLM request would be impractical, at least with current techniques. Function definitions are injected into the LLM prompt, so incorporating a large number of functions could consume too much of the limited context window, and LLMs struggle to effectively utilize long context. This suggests that LLMs could benefit from an intermediate function-selection step, perhaps integrated with CoT reasoning and planning, to examine a user question and identify a subset of the available functions that appear most relevant to answering the question. Research is needed to determine robust, scalable mechanisms for enabling LLMs to make effective use of large function collections.

Additional considerations in real-world settings include input validation and verification, to ensure that functions behave as documented and as understood by the LLM, possibly by applying formal (automated) verification techniques. Furthermore, the use of function calls can multiply the number of LLM queries, resulting in higher cost and slower response time. Further research is needed, both to determine how best to design and incorporate functions and knowledge structures so that LLM performance is maximized, and to develop techniques for eliciting more efficient and reliable LLM behaviour.

Finally, a surprising anecdote highlights the risk of using closed-source LLMs subject to change without notice. While querying GPT-4 on a question related to Israel-based companies, after initially responding as expected, it refused to answer any questions related to Israel. Since we were using a specific snapshot of the GPT-4 model, such instability was not expected. It is possible that the OpenAI hidden prompt had been modified. While such controls may be necessary in public consumer applications, in professional contexts, such changes in behaviour could decrease reliability, undermine user trust, and potentially cause financial loss, regulatory breach, or legal liability. Adoption of closed-source models in business may require greater transparency and mechanisms for giving customers advance notice about potentially breaking changes.

Conclusion

Despite the promise of LLMs and the substantial effort put into tuning their performance in numerous domains with fine-tuning and prompt engineering, it remains unclear how well LLMs can utilize contextual information and domain knowledge for question-answering in professional settings. To address this gap, we assessed LLM performance in supporting investment bankers on a custom benchmark of 16 relatively straightforward analytical tasks representative of the investment banking industry. We evaluated LLM performance without special prompting, with relevant information provided in the prompt, and as part of a system giving the LLM access to domain-tuned functions for information retrieval and planning.

Our results show that without access to functions, state of the art LLMs performed poorly, completing two or fewer tasks correctly. Prompting with functions resulted in significantly better results, although LLM performance was highly sensitive to the design of the functions and the structure of the information they returned. In our experiments, the most effective design was a combination of CoT reasoning with function calls, access to the information schemata and where the LLM was permitted to choose which information the functions returned. This enabled the LLMs to effectively retrieve and process relevant information, resulting in correct answers on 12 out of 16 tasks. Our results show that one way to empower LLMs with domain knowledge is to incorporate domain-specific functions and information structures that enable LLMs to reason in domain-appropriate ways. These techniques may prove to be a powerful means of adapting LLMs for use in demanding professional settings.

References

Ashok, D.; and Lipton, Z. C. 2023. PromptNER: Prompting For Named Entity Recognition. ArXiv:2305.15444 [cs].

Dell'Acqua, F.; McFowland, E.; Mollick, E. R.; Lifshitz-Assaf, H.; Kellogg, K.; Rajendran, S.; Krayer, L.; Candelon, F.; and Lakhani, K. R. 2023. Navigating the Jagged Technological Frontier: Field Experimental Evidence of the Effects of AI on Knowledge Worker Productivity and Quality. *Harvard Business School Technology & Operations Mgt. Unit Working Paper No. 24-013*.

Gan, Y.; Chen, X.; Huang, Q.; Purver, M.; Woodward, J. R.; Xie, J.; and Huang, P. 2021. Towards Robustness of Text-to-SQL Models against Synonym Substitution. In Zong, C.; Xia, F.; Li, W.; and Navigli, R., eds., *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), 2505–2515. Online: Association for Computational Linguistics.

Gao, D.; Wang, H.; Li, Y.; Sun, X.; Qian, Y.; Ding, B.; and Zhou, J. 2023. Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation. ArXiv:2308.15363 [cs].

Guo, J.; Du, L.; Liu, H.; Zhou, M.; He, X.; and Han, S. 2023. GPT4Graph: Can Large Language Models Understand Graph Structured Data ? An Empirical Evaluation and Benchmarking. ArXiv:2305.15066 [cs].

Guo, Y.; Xu, Z.; and Yang, Y. 2023. Is ChatGPT a Financial Expert? Evaluating Language Models on Financial Natural Language Processing. *Findings of the Association for Computational Linguistics: EMNLP 2023*, 815–821. ArXiv:2310.12664 [cs].

Lan, W.; Wang, Z.; Chauhan, A.; Zhu, H.; Li, A.; Guo, J.; Zhang, S.; Hang, C.-W.; Lilien, J.; Hu, Y.; Pan, L.; Dong, M.; Wang, J.; Jiang, J.; Ash, S.; Castelli, V.; Ng, P.; and Xiang, B. 2023. UNITE: A Unified Benchmark for Text-to-SQL Evaluation. ArXiv:2305.16265 [cs].

Liu, N. F.; Lin, K.; Hewitt, J.; Paranjape, A.; Bevilacqua, M.; Petroni, F.; and Liang, P. 2023. Lost in the Middle: How Language Models Use Long Contexts. ArXiv:2307.03172 [cs].

Liu, X.; and Tan, Z. 2023. Divide and Prompt: Chain of Thought Prompting for Text-to-SQL. ArXiv:2304.11556 [cs].

Packer, C.; Fang, V.; Patil, S. G.; Lin, K.; Wooders, S.; and Gonzalez, J. E. 2023. MemGPT: Towards LLMs as Operating Systems. arXiv:2310.08560.

Pal, A.; Karkhanis, D.; Roberts, M.; Dooley, S.; Sundararajan, A.; and Naidu, S. 2023. Giraffe: Adventures in Expanding Context Lengths in LLMs. ArXiv:2308.10882 [cs].

Ram, O.; Levine, Y.; Dalmedigos, I.; Muhlgay, D.; Shashua, A.; Leyton-Brown, K.; and Shoham, Y. 2023. In-Context Retrieval-Augmented Language Models. *Transactions of the Association for Computational Linguistics*, 11: 1316–1331. Place: Cambridge, MA Publisher: MIT Press.

Su, D.; Patwary, M.; Prabhumoye, S.; Xu, P.; Prenger, R.; Shoeybi, M.; Fung, P.; Anandkumar, A.; and Catanzaro, B. 2023. Context Generation Improves Open Domain Question Answering. ArXiv:2210.06349 [cs].

Sui, Y.; Zhou, M.; Zhou, M.; Han, S.; and Zhang, D. 2024. Table Meets LLM: Can Large Language Models Understand Structured Table Data? A Benchmark and Empirical Study. In *The 17th ACM International Conference on Web Search and Data Mining (WSDM '24)*.

Wang, Q.; Ding, L.; Cao, Y.; Tian, Z.; Wang, S.; Tao, D.; and Guo, L. 2023. Recursively Summarizing Enables Long-Term Dialogue Memory in Large Language Models. ArXiv:2308.15022 [cs] version: 1.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.; and Zhou, D. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models.

Wu, S.; Irsoy, O.; Lu, S.; Dabravolski, V.; Dredze, M.; Gehrmann, S.; Kambadur, P.; Rosenberg, D.; and Mann, G. 2023. BloombergGPT: A Large Language Model for Finance. ArXiv:2303.17564 [cs, q-fin].

Xu, P.; Ping, W.; Wu, X.; McAfee, L.; Zhu, C.; Liu, Z.; Subramanian, S.; Bakhturina, E.; Shoeybi, M.; and Catanzaro, B. 2023. Retrieval meets Long Context Large Language Models. *Findings of the Association for Computational Linguistics: EMNLP 2023*, 815–821.

Yu, T.; Zhang, R.; Yang, K.; Yasunaga, M.; Wang, D.; Li, Z.; Ma, J.; Li, I.; Yao, Q.; Roman, S.; Zhang, Z.; and Radev, D. 2018. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In *EMNLP*.

Ziqi, J.; and Lu, W. 2023. Tab-CoT: Zero-shot Tabular Chain of Thought. In Rogers, A.; Boyd-Graber, J.; and Okazaki, N., eds., *Findings of the Association for Computational Linguistics: ACL 2023*, 10259–10277. Toronto, Canada: Association for Computational Linguistics.