Systematic Evaluation of Large Language Models in NL-to-SQL Translation

Background

In recent years, Large Language Models (LLMs) like GPT have significantly advanced Natural Language Understanding (NLU), showing remarkable ability in mimicking human text (Brown et al., 2020). A key NLU challenge is translating Natural Language (NL) into Code. This offers several significant benefits that streamline software development processes, enhance data accessibility, and democratize programming.

Within the realm of NL-to-Code, the NL-to-SQL translation has emerged as a critical field. This reflects the growing emphasis on refining LLMs to transform natural language into SQL queries. The key motivation behind this is to enhance human-computer interactions and also to facilitate direct, user-friendly access to vast databases without requiring specialized SQL knowledge (Iyer et al., 2018). This has spurred numerous research, leading to few benchmarking studies of comparing and fine tuning LLMs, aimed at optimizing NL-to-SQL tasks.

While substantial progress has been made in the NL-to-SQL domain, several pivotal gaps remain unaddressed. First of all, benchmarking studies in this field have been limited so far, offering scant insights beyond basic model evaluations. Secondly, an analysis of how Large Language Models (LLMs) fare across different hosting platforms—such as cloud platforms like AnyScale, Bedrock, Anthropic, OpenAI versus self-hosted environment—remains largely absent. This overlooks how environmental factors can significantly influence model performance, a gap that is crucial for informed deployment and resource optimization.

Moreover, most existing efforts have narrowly focused on accuracy as the sole metric for assessing LLM efficacy, sidelining other critical dimensions like throughput, latency, cost, etc. These aspects are vital for gauging the practicality of LLMs in real-world applications, suggesting a pressing need for studies that offer a holistic view by comparing all relevant metrics concurrently. Additionally, there's a lack of comprehensive analysis on the economic impact of deploying LLMs, including the cost-performance trade-offs of various LLM architectures and deployment strategies.

Aim of this report

In light of these gaps in the existing literature, our research aims to provide a comprehensive evaluation of LLMs' performance in NL-to-SQL tasks. Our work is aimed at providing profound insights, enabling informed decision making about usage of LLMs, deployment, strategic resource allocation and budget planning. Our findings will not only provide valuable insights for researchers and practitioners, but also contribute to the ongoing efforts to improve the practical utility and scalability of LLMs in real-world applications.

Methods

Selection of Dataset

We chose the BIRD dataset for our NL-to-SQL translation tasks due to its complexity and the rich challenges it presents, offering a more rigorous testing ground than alternatives like Spider (Li et al., 2024; Yu et al., 2018). Despite the potential for lower accuracy scores given BIRD's complexity, we believe it better tests our models' capabilities in handling challenging queries. The BIRD dataset's inclusion of an 'evidence' column further supports nuanced query generation and evaluation.

Sample Size Determination and Stratification

Our study used 360 question-query pairs from BIRD's dev dataset, categorized into three difficulty levels: simple, moderate and challenging. This sample size, determined through iterative testing with the GPT-3.5 model, ensures comprehensive coverage and a balanced representation of the dataset's challenges. This sample was stratified to include an equal distribution of difficulty levels, with 120 pairs selected from each category ensuring a balanced representation of the dataset's complexity.

Design

The design involves running the 360 question-query pairs for each model across platforms, conducting five trials per model. The five trials vary in the number of instructions provided to the models. The instructions are general rules that LLMs need to know to provide correct SQL responses. The aim here is to assess adding more information to the model incrementally impact SQL query generation capabilities. First trial will not have any instructions or guided prompts. From trial 2 to 5 we had instructions in the sets of 5, 7, 9 and 11, incrementing them by 2 in each trial.

Platforms and Models

We ran the 360 question-query pairs across several platforms: Anyscale, Bedrock, Anthropic, Google Gemini 1.0 Pro, and a self-hosted setup. We tested a variety of models including CodeLlama, Mistral, Claude 3 models, GPT family of models (Table 1). We have also included Google’s Gemini 1.0 Pro and Databricks’ DBRX Instruct. This allowed us to compare performances, costs, and speed across different hosting environments.

Platforms	Models
Anyscale	Llama2-70B Mistral 7B (Version 1) Mixtral8x7B CodeLlama-70B
Amazon Bedrock	Llama2-70B Mistral-7B (Version 2) Mixtral8x7B Claude3 - Sonnet Claude3 - Haiku
OpenAI	GPT3.5 Turbo GPT4 Turbo
Anthropic	Claude3 - Opus Claude3 - Sonnet Claude3 - Haiku
Google	Gemini 1.0 Pro
Self-Hosted	DBRX Instruct CodeLlma-70B CodeLlma-34B Mistral-7B (Version 1) Mistral-7B (Version 2) Mixtral8x7B Sql Code 7B2 Sql Code 70B Alpha WizardCoder-33B

Table 1. List of all models used and hosted in different platforms

Results

Our Benchmark study focuses on studying 3 important metrics: (1) Execution Accuracy (2) Throughput, and (3) Total Cost of Models compared alongside with number of tokens.

Execution Accuracy (EX)

To evaluate the SQL queries generated by LLMs against the BIRD dataset given SQL queries, we employed Execution Accuracy (EX) metrics. This metric helps us to quantitatively assess the accuracy of LLM-generated queries, providing insights into the models' applicability in the real-world. This approach involves evaluating the results of LLM generated queries against the expected results. For this we have written our eval script, by doing few optimal changes on the BIRD's eval script.

Overall Comparison of EX

To compare overall accuracy between models, we averaged the accuracy of each model across 5 instruction trials (360 inferences per trial), summing up to 1800 inferences per model. We found that Anthropic's Claude3-Opus and OpenAI's GPT 4 Turbo are the models with the highest accuracy of 39.55% and 39.39% respectively. Following these are the Claude3-Sonnet (35.94% in Bedrock; 34.67% in Anthropic), Claude3-Haiku (34.22% in Bedrock; 34.72% in Anthropic), and Open AI's GPT 3.5 Turbo (33.83%). Self-hosted models such as Wizardcoder 33B and SQLcoder-70B-alpha achieved accuracy of 31% and 30% (Figure 1). Google Gemini 1.0 Pro also stood at 26.89%.

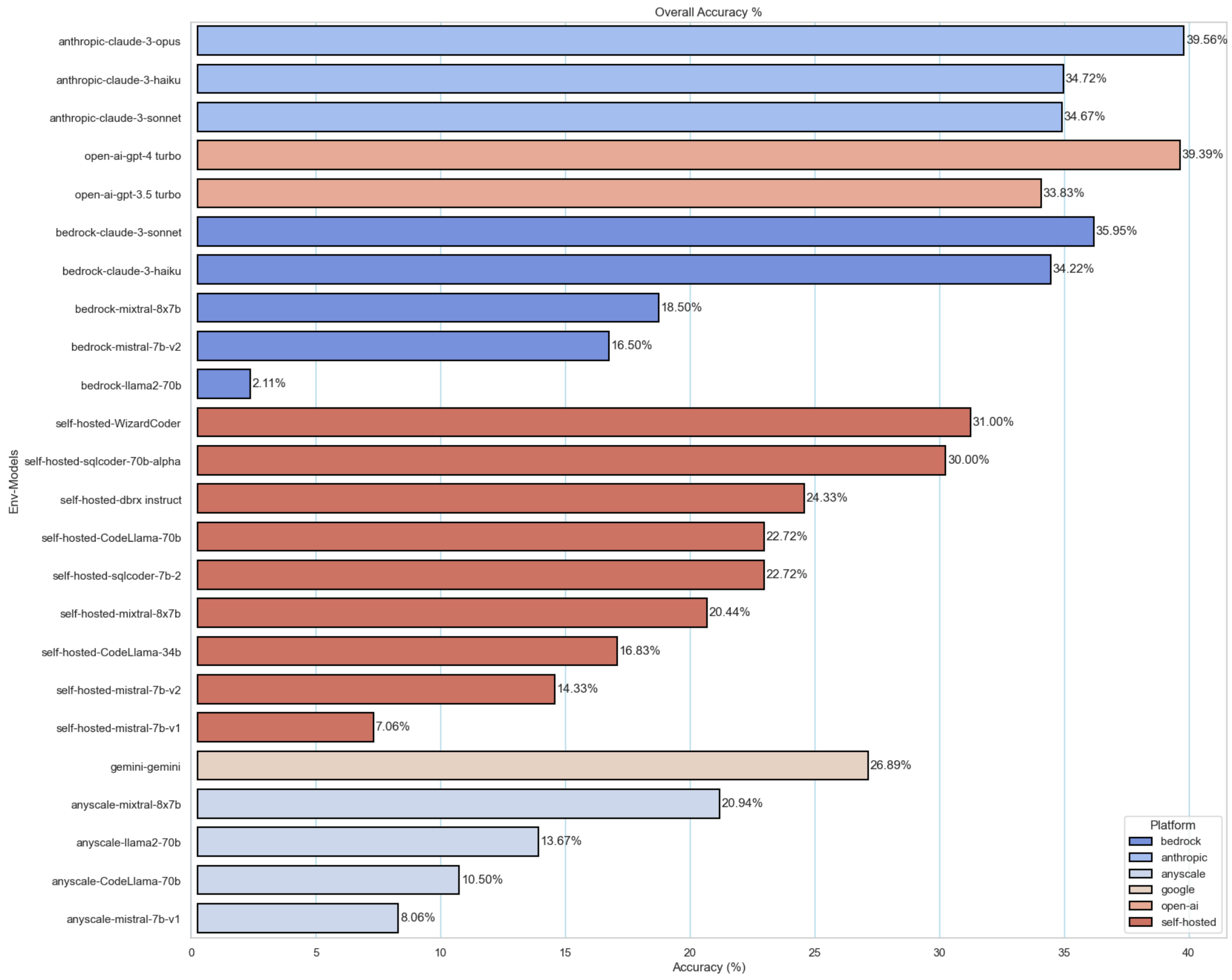


Figure 1. Average EX of each model compared to others.

We also compared models within each platform where they are hosted. In anyscale Mixtral-8x7B had a maximum average accuracy of 23.06% (average of all 5 instruction trials). This is followed by Llama2-70B (13.67%) and Codellama-70B (10.5%). In Bedrock, Claude 3 models of Sonnet and Haiku exhibited closer accuracy of 35.94% and 34.33%, respectively. OpenAI's GPT 4 turbo model had a higher accuracy (39.39%) compared to GPT 3.5 turbo (33.83%). Similarly, Claude3 Opus had the highest accuracy

(39.55%), amongst all three Anthropic models. Amongst self-hosted, Wizardcoder 33B (31%) and SQLcoder-70B (30%) exhibited highest accuracy. DBRX Instruct, the latest model of Databricks released recently, had an average accuracy of 24.33%.

Comparison of EX across Instructions within Platforms

At the next level, we analyzed how model accuracy varies within a platform when increasing the number of instructions provided in the prompt from 0, 5, 7, 9 to 11. For example, in Anyscale, we observed that for all 4 models accuracy increased when 5 instructions were provided compared to no instructions (Figure 2). Models such as Mixtral-8x7B, Llama2-70B and Codellama-70B attained peak performance in the trial of 11 instructions (23.06%, 15.28%, and 12.5%). Mistral-7B-V1 peaked in the trial of 5 instructions (9.44%). We also noticed in all the 4 models, when the number of instructions increases, that doesn't necessarily reflect an increase in accuracy. On the contrary we found that there is a marginal drop in accuracy from 5 to 7, 7 to 9 or both.

Similar pattern observed in Bedrock, Open AI and Anthropic (Figure 2 & 3). In Bedrock, in Llama-70B, Mistral-7B-V2 and Mixtral 8x7B, we can observe that the variance in accuracy % when additional instructions are added, are higher compared to Claude3-Haiku and Sonnet models. Both in Bedrock and Anthropic, all Claude3 models did not exhibit any drastic fluctuations in accuracy when more instructions were added. Similarly, both GPT 3.5 Turbo and GPT 4 Turbo displayed a near constant trend in accuracy moving from 0 to 11 instructions.

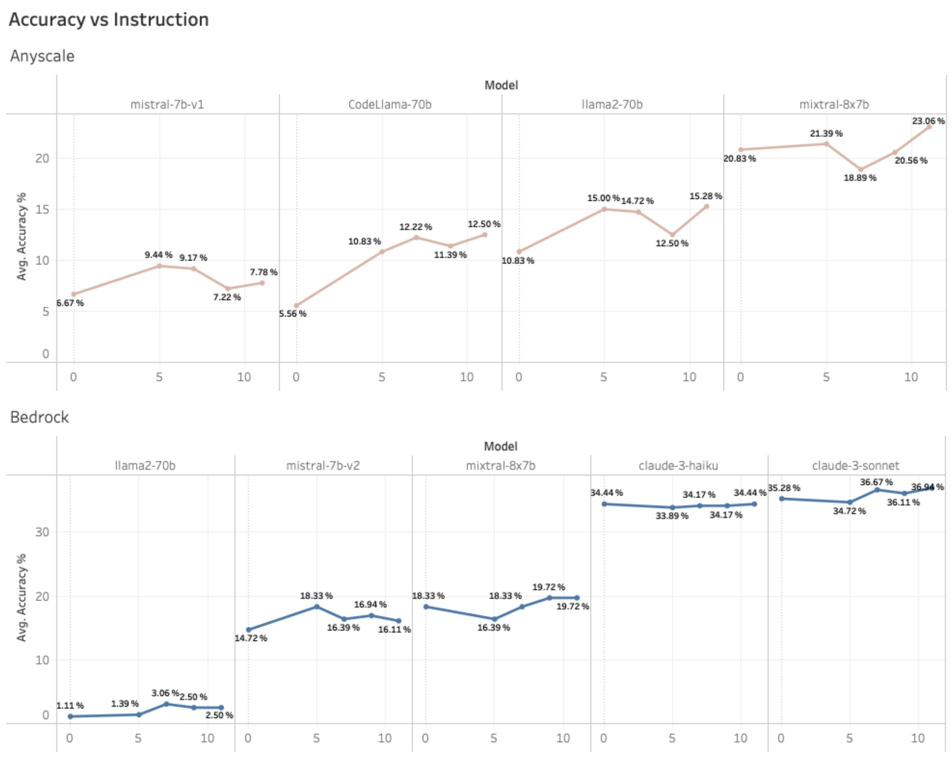


Figure 2. EX % comparison between models within Anyscale and Bedrock. The chart shows how EX changes from 0 to 11 instructions.

Amongst the self-hosted models, we observed that the variations in accuracy when we increase instructions from 0 to 11 is big (Figure 3). For example, for models such as Wizardcoder, Sqlcoder-70B, Sqlcoder-7B-2, Mixtral 8x7B, Mistral 7B-V1 and V2, the accuracy % increases drastically when we move from no instruction to 5 instructions. As observed in other platforms, here too the accuracy % drops from 5 to 7, 7 to 9, or 9 to 11. Interestingly, for DBRX Instruct and Codellama-70B, we noticed a massive drop in accuracy for 0 to 5 instructions, from 26.67% to 22.5% and 29.17% to 23.33%, respectively. Also each model attained its own peak accuracy in any instruction trial, exhibiting no pattern or consistency in accuracy in relation to number of instructions.

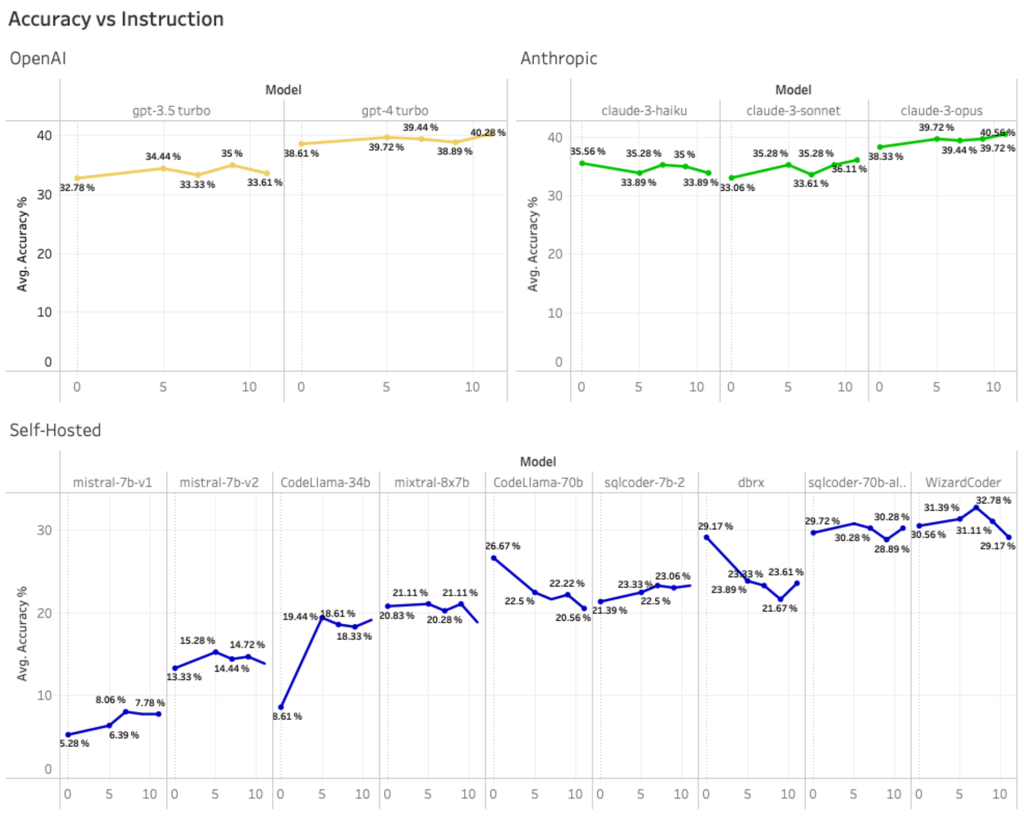


Figure 3. EX % comparison between models within OpenAI, Anthropic and Self-Hosted. The chart shows how EX changes from 0 to 11 instructions.

Comparison of EX Between Platforms

It is worth noting that one of the key goals of this study is to find out how the same model when hosted in different platforms behave differently. From this interesting results have emerged (Figure 4). Firstly, we looked at the Claude3-Sonnet and Haiku models hosted in both Anthropic and Amazon Bedrock.

Interestingly, Claude3-Sonnet's accuracy % (averaged across 0 to 11 instructions) is marginally higher in Bedrock (35.94%) than in Anthropic (34.67%). In Haiku, the accuracy % in Bedrock (34.22%) and Anthropic (34.72%) are nearly identical.

Secondly, the biggest difference between platforms is observed for Llama2-70B. In Bedrock, we got an accuracy of 2.11% and in Anyscale it was 13.67%, revealing a nearly 6x times difference. Similarly, a big difference was also observed in Codellama-70B hosted with Anyscale (10.5%) and Self-hosted (22.72%), more than a 2x times difference. Thirdly, the models of Mistral including Mistal-7B versions and Mixtral 8x7B, exhibited a significant difference mainly between Bedrock and self-hosted environment (Figure 4). For example, Mistral 7B-V2 at Bedrock showed a higher accuracy (16.5%) compared to self-hosted (14.33%). On the other hand Mixtal 8x7B showed a higher accuracy (20.44%) in self-hosting compared to Bedrock (18.5%). For the Mistral family of models, not much notable difference is observed between Anyscale and self-hosted.

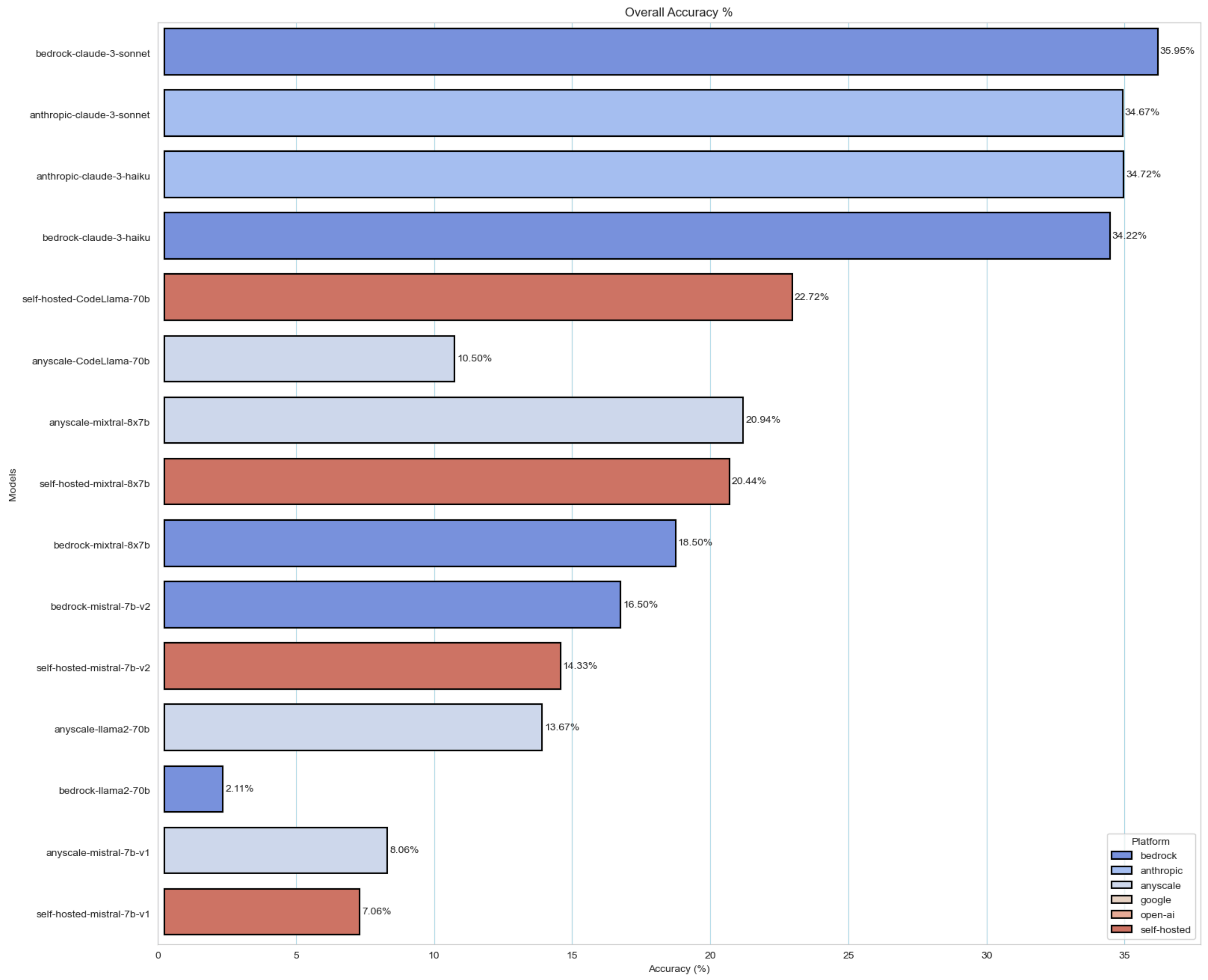


Figure 4. EX % comparison of models between platforms.

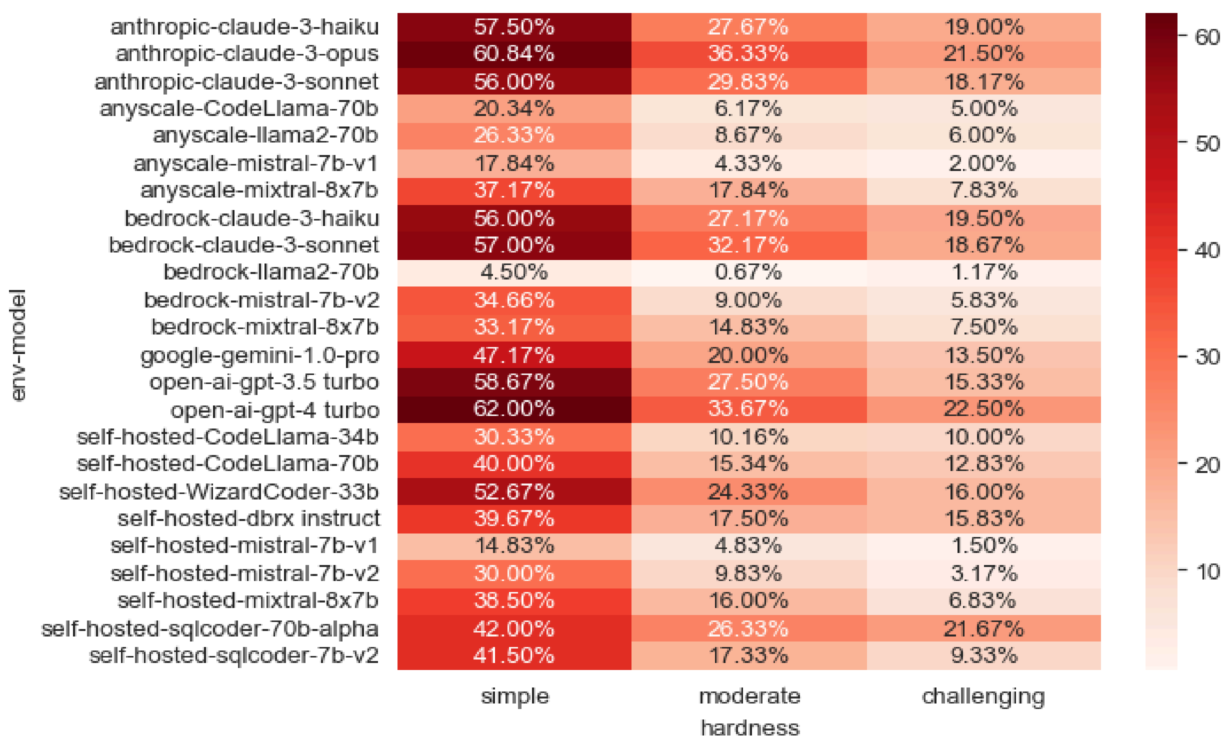
EX across Hardness Levels

The sample chosen from the BIRD dataset has questions with three levels of hardness: simple, moderate and challenging. We compared the accuracy in relation to these hardness levels. We found that across models accuracy dropped when hardness levels increased from simple to moderate, and from moderate to challenging (Figure 5).

In the simple level NL-to-SQL tasks, GPT-4 Turbo (62%) and Claude3-Opus (60.84%) are models with highest accuracy. Followed by it are GPT-3.5 Turbo (58.67%), Claude3-Sonnet and Haiku. They have the same level of accuracy in both Anthropic and Bedrock platforms. In fact, it is interesting to note that Claude3-Haiku has achieved a marginally higher accuracy (57.5%) compared to Sonnet (56%) for simple category tasks. Amongst self-hosted models, Wizardcoder-33B achieved the highest accuracy of 52.67%, followed by SQLcoder-70B Alpha (42%).

In moderate level tasks, we noticed a drop in accuracy for all models (Figure 5). We also noticed that only in moderate category tasks Claude3 Opus (36.33%) outperformed GPT-4 Turbo (33.67%). Similarly, Claude3-Sonnet (32.17% in Bedrock & 29.83% in Anthropic) outperformed Haiku (27.17% in Bedrock & 27.67% in Anthropic), only in moderate category tasks. Amongst self-hosted models, Sqlcoder-70B alpha (26.33%) has greater accuracy than Wizardcoder-33B (24.33%). We noticed that the drop in accuracy from simple to moderate difficulty is also least in Sqlcoder-70B alpha, Claude3-Opus and Sonnet compared to other models.

In challenging tasks, GPT-4 Turbo has the highest accuracy (22.5%). This is followed by SQLcoder-70B alpha (21.67%) which stands at par with both GPT-4 and Claude3 Opus (21.5%). The accuracy drop is observed across all the models. But the drop was least for Codellama-34B, DBRX Instruct and SQLcoder-70B alpha.



env-model	simple	moderate	challenging
anthropic-claude-3-haiku	57.50%	27.67%	19.00%
anthropic-claude-3-opus	60.84%	36.33%	21.50%
anthropic-claude-3-sonnet	56.00%	29.83%	18.17%
anyscale-CodeLlama-70b	20.34%	6.17%	5.00%
anyscale-llama2-70b	26.33%	8.67%	6.00%
anyscale-mistral-7b-v1	17.84%	4.33%	2.00%
anyscale-mixtral-8x7b	37.17%	17.84%	7.83%
bedrock-claude-3-haiku	56.00%	27.17%	19.50%
bedrock-claude-3-sonnet	57.00%	32.17%	18.67%
bedrock-llama2-70b	4.50%	0.67%	1.17%
bedrock-mistral-7b-v2	34.66%	9.00%	5.83%
bedrock-mixtral-8x7b	33.17%	14.83%	7.50%
google-gemini-1.0-pro	47.17%	20.00%	13.50%
open-ai-gpt-3.5 turbo	58.67%	27.50%	15.33%
open-ai-gpt-4 turbo	62.00%	33.67%	22.50%
self-hosted-CodeLlama-34b	30.33%	10.16%	10.00%
self-hosted-CodeLlama-70b	40.00%	15.34%	12.83%
self-hosted-WizardCoder-33b	52.67%	24.33%	16.00%
self-hosted-dbrx instruct	39.67%	17.50%	15.83%
self-hosted-mistral-7b-v1	14.83%	4.83%	1.50%
self-hosted-mistral-7b-v2	30.00%	9.83%	3.17%
self-hosted-mixtral-8x7b	38.50%	16.00%	6.83%
self-hosted-sqlcoder-70b-alpha	42.00%	26.33%	21.67%
self-hosted-sqlcoder-7b-v2	41.50%	17.33%	9.33%

Figure 5. EX % across 3 levels of Question Hardness.

Cost

Total_cost serves as the primary metric for evaluating the expenditure associated with operating various large language models (LLMs) across different hosting environments. Our analysis also considered how costs fluctuate based on factors such as model size, question complexity, and the number of instructions provided.

Overall Observations

Our findings indicate that self-hosted models consistently incur significantly higher costs compared to those hosted on other platforms. Notably, the models Codellama70b and DBRx Instruct lead in expenses, costing $94.82 and $88.81, respectively. Other models like Mixtral 8x7b and Coellama 34b also show considerable costs. Following closely are anthropic claude 3 Opus at $47.3 and GPT 4 turbo at $22.61. The lowest expenses were observed for models operated on Anyscale (Figure 6).

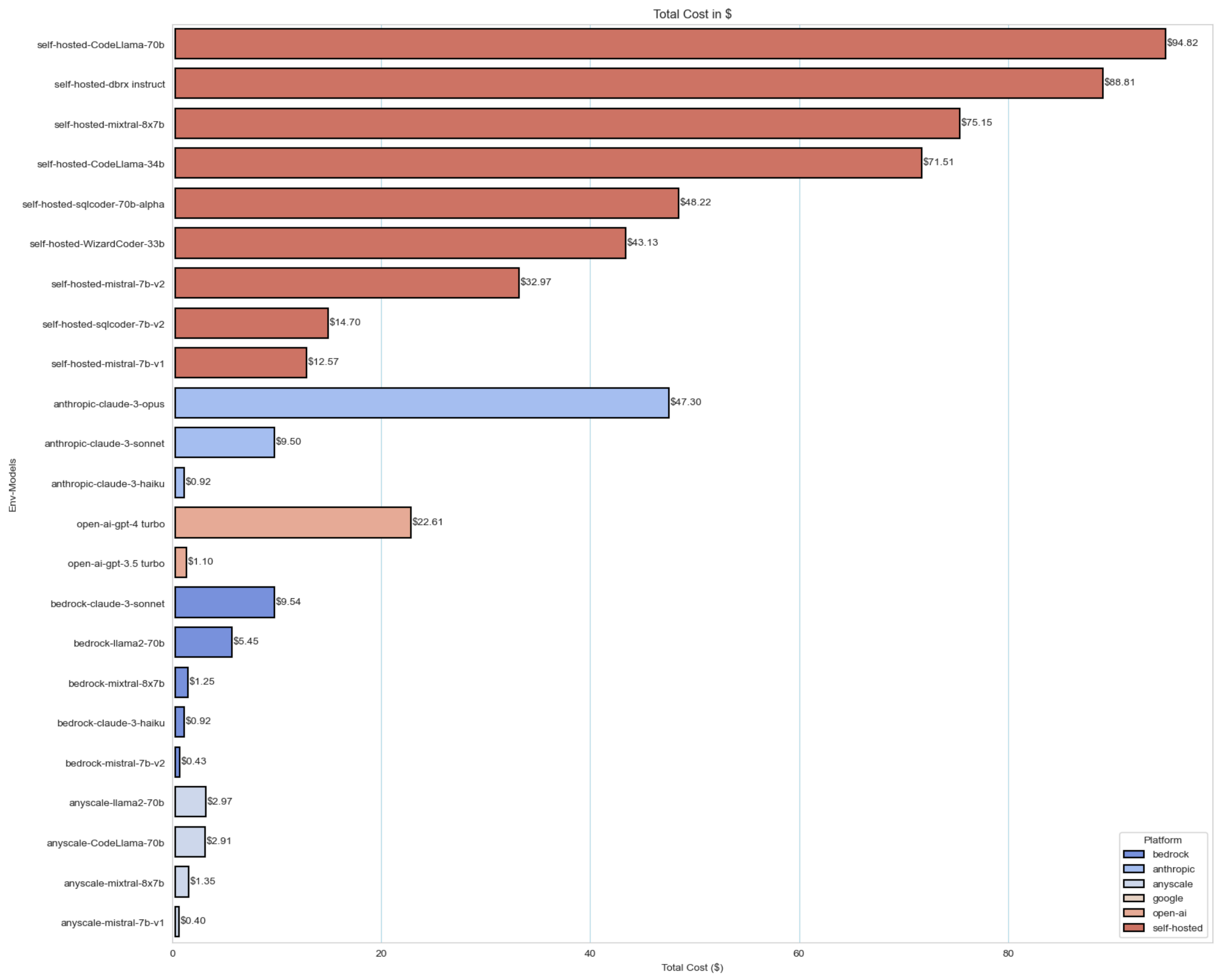


Figure 6. Total cost spent to run the models across platforms.

From Figure 7, we observe a pronounced disparity in costs between self-hosted and other hosting platforms. For instance, the "codelama70b" model on a self-hosted platform incurs a cost of $94.82 compared to just $2.91 on Anyscale. Similarly, the "mixtral 8x7b" model costs approximately the same on Anyscale and Bedrock; however, it requires $75.15 to run on a self-hosted platform. While the cost differences among Anyscale, Bedrock, and Anthropic platforms are minimal, self-hosting significantly escalates the costs. This further emphasizes the economic impact of the hosting choice in deploying large language models.

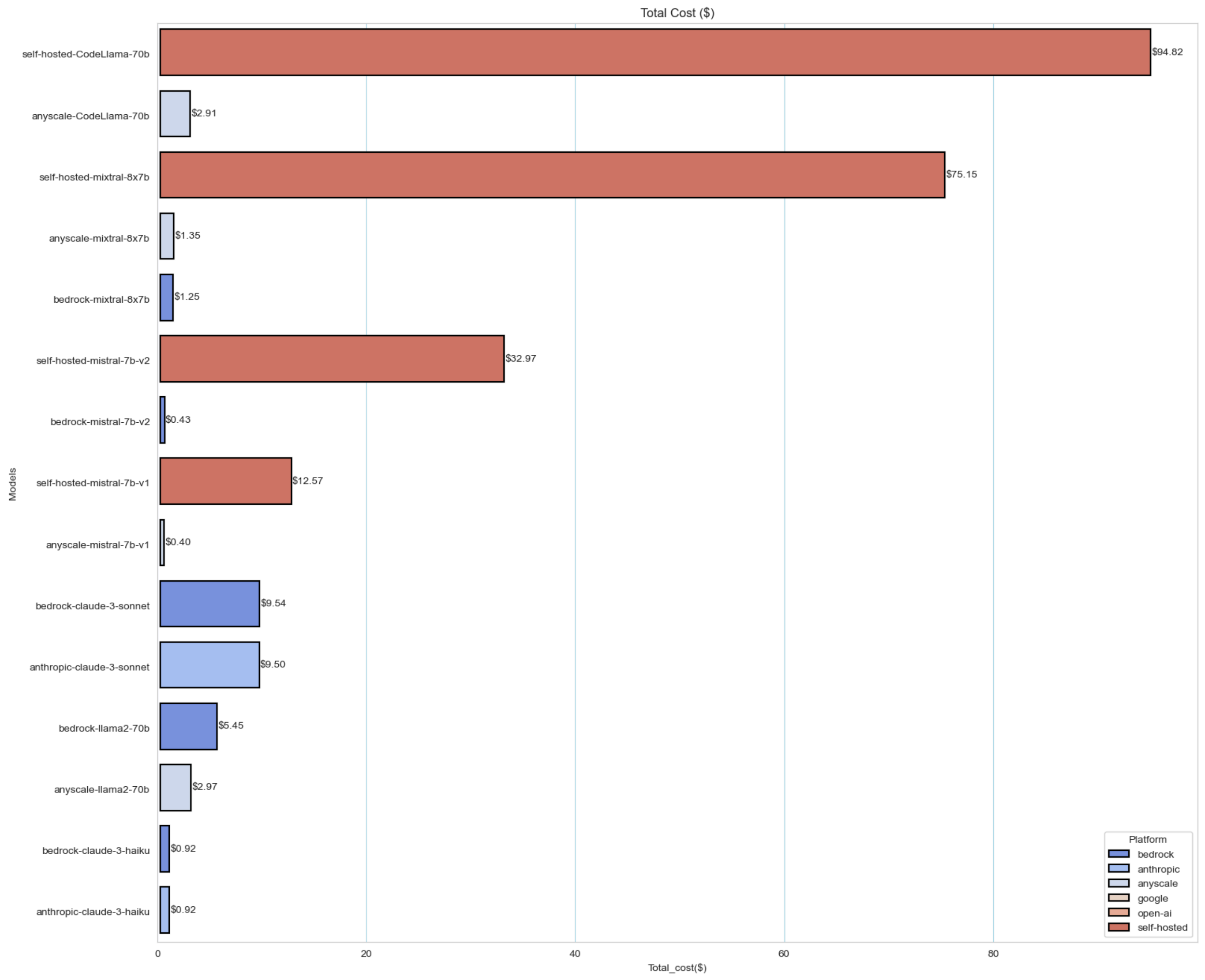


Figure 7. Total cost spent to run the models between platforms (self hosted vs other platforms)

The stark cost disparity between self-hosted and other platforms can be attributed to the differing basis for cost calculation. For non-self-hosted platforms, costs are determined by the number of input and output tokens. Conversely, for self-hosted models, costs are calculated based on latency—the time taken to process inferences.

Self-hosted models displayed higher average latencies (for 1800 inferences), in stark contrast to their counterparts on other platforms. Also there is a direct correlation observed between increased latency and higher incurred costs in self-hosted environments. (Figure 8). Therefore it is important to look into the findings of both separately.

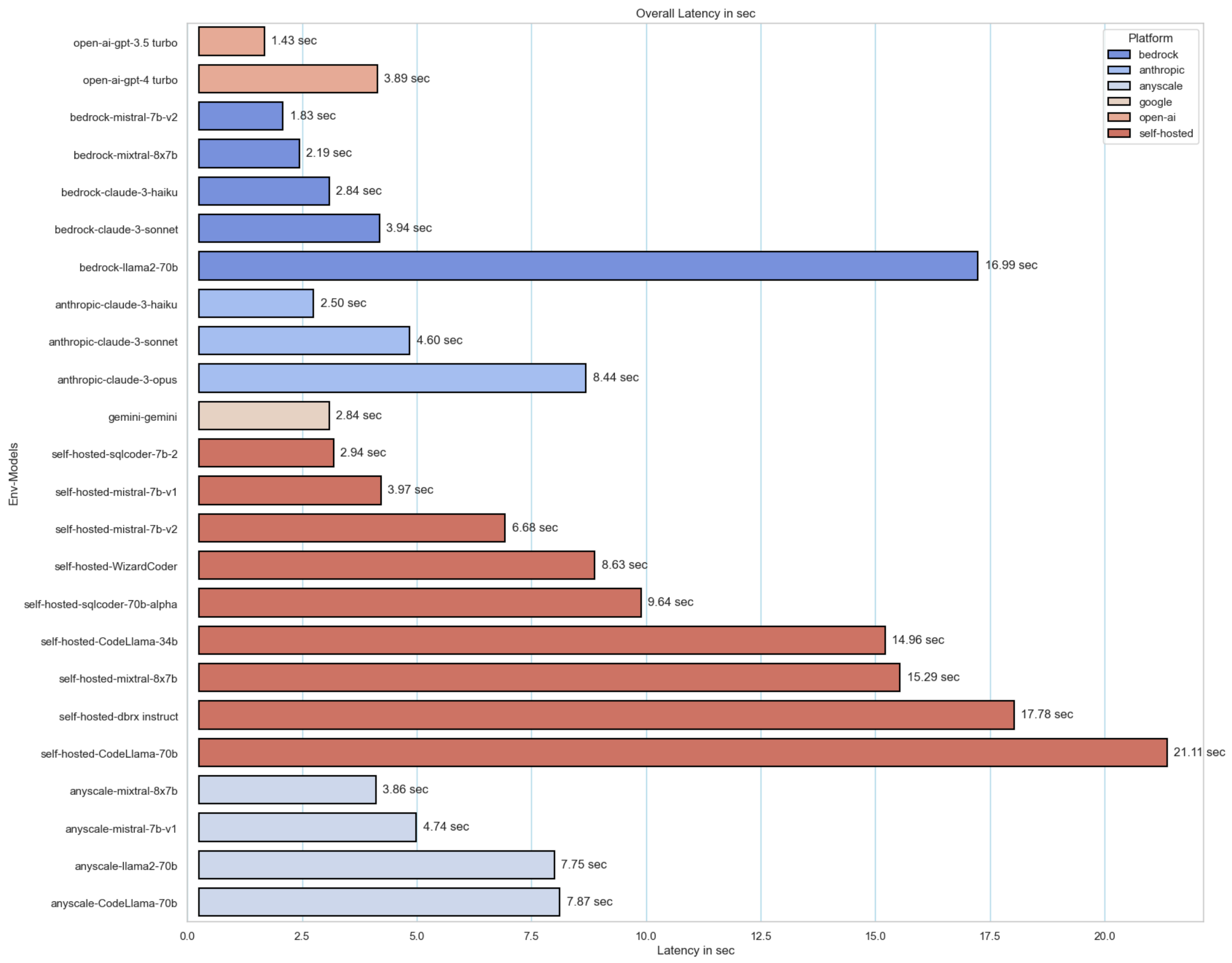


Figure 8. Average latency per inference (averaged for 1800 inferences) for all models across platforms.

Discussion

Execution Accuracy (EX)

The comparative analysis of Execution Accuracy (EX), Throughput and Cost, across various LLMs when applied to SQL query generation offers a fascinating glimpse into the current capabilities and limitations of these models in generating structured query language. Our observations draw attention to several key aspects, ranging from overall performance to model behavior across different platforms and prompted with different sets of instructions.

Overall Performance Insights

First of considering EX, the study identifies Anthropic's Claude3-Opus and OpenAI's GPT-4 Turbo as frontrunners, demonstrating the pinnacle of current LLM capabilities with accuracies of 39.55% and 39.39%, respectively. This is the maximum result that could be achieved by any vanilla model purely by their inherent capabilities. This close competition underscores the rapid catch-up of Anthropic Claude over OpenAI's GPT, showcasing remarkable proficiency in understanding and generating SQL queries. The next tier of performance includes Claude3-Sonnet, Claude3-Haiku, and GPT-3.5 Turbo, indicating a significant grouping of models with competitive capabilities just below the leading duo.

Notably, models such as Google Gemini 1.0 Pro, alongside Llama and Mistral versions, present a lag in performance, with Mistral V2 showing a slight edge over V1. This highlights the diversity in effectiveness of LLMs with certain models still finding their footing in NL-to-SQL tasks.

Among self-hosted models, Wizardcoder-33B and SQLcoder-70B Alpha demonstrated commendable performance, engaging in a tight race and underscoring the potential of using open source models (pre trained and fine tuned) with self-hosted solutions. We also used Databricks' DBRX Instruct into this benchmark as it was recently released with much anticipation surrounding its performance in NL-to-code tasks. It provided a compelling case for our evaluation. Interestingly our findings reveal that the vanilla version of DBRX Instruct trails behind its peers in accuracy. From this we could surmise that the model may potentially showcase its strengths in other of NLU apart from NL-to-SQL tasks. This also requires further investigation on it.

Model Size Impacting Accuracy

Contrary to the often-assumed correlation between model size and performance, we observe that Claude3-Haiku, despite being lighter than Sonnet, performs comparably well in NL-to-SQL tasks (and sometimes better). This finding challenges the usual notion that larger models universally produce higher accuracy. This observation shows that this cannot always be necessarily true.

Instruction Sensitivity

We also found an interesting result from this study which has challenged and contradicted a natural expectation and assumption. We found that more instructions in the form of prompts do not always reflect higher accuracy. We found that accuracy generally improves with a moderate amount of instructions but does not consistently increase with additional input. Notably, after five instructions, we observed either a plateau or a decline in performance, and with a potential increase again at eleven instructions for some models.

Most importantly, we observed that larger models, such as those from Claude3 and OpenAI, exhibited a stable performance across increasing instruction levels. This nonlinear trend underscores that these models have been inherently pre trained or optimized with NL-to-SQL pertinent instructions and rules to efficiently process a given prompt regardless of the instructions' volume. In contrast, smaller and self-hosted models, like the Llama series and DBRX Instruct, showed greater sensitivity to changes in instructions, with some models even performing lesser when transitioning from zero to five instructions.

This nuanced interaction between the number of instructions and model performance highlights the complexity of LLM processing capabilities and suggests an optimal level of instructional detail exists. There is a dire need for further research on it emphasizing the importance of understanding and tailoring the instructional input to enhance model effectiveness.

Platform-dependent Performance

In our analysis comparing model performance across different platforms, we observed distinct patterns. Claude models showed minimal performance difference between Amazon Bedrock and Anthropic platforms, suggesting these models perform consistently regardless of the hosting environment. This indicates a robustness in Claude models and its hosting framework, with significant sync between Bedrock and Anthropic in hosting the Claude models.

The Mistral family models displayed slight variations across platforms, pointing to a mild sensitivity to their hosting environment but with generally stable performance. This suggests that while platform differences exist, they do not drastically affect Mistral models' outcomes.

In contrast, Llama models exhibited significant performance disparities between self-hosted environments and those on Anyscale or Bedrock. This stark difference highlights the importance of platform-specific optimizations and changes in inference engines, particularly for Llama models (Codellama and

Llama-70B). We could surmise that Anyscale and Bedrock might have done specialized optimization to improve other aspects of model performances such as latency, throughput, cost, etc., that has compromised on accuracy. This could have had a larger impact specifically on Llama models.
These observations underline the role of the hosting environment in model performances, with some models being more affected by their operational platform than others. This emphasizes the need for careful platform selection, especially for models sensitive to their hosting conditions.

Hardness Level Considerations

Our analysis of Execution Accuracy (EX) across varying levels of query difficulty in the BIRD dataset revealed a clear trend: as the complexity of the questions increased, the overall accuracy of the models decreased. This trend underscores the intuitive understanding that more complex queries pose greater challenges for Large Language Models (LLMs) in generating accurate SQL queries.

Among the models we evaluated, Anthropic's Claude3-Opus and OpenAI's GPT-4 stood out for their superior performance across all levels of question difficulty, highlighting their advanced capabilities in dealing with complex natural language to SQL conversion tasks. Remarkably, the Claude3-Haiku model, which is less expensive compared to Opus and Sonnet, demonstrated a performance on par with these top performers, especially in handling challenging tasks. This suggests that despite being a lighter model, Haiku offers an impressive return on investment for NL-to-SQL tasks, achieving high accuracy without the need for the extensive spending required for its counterparts.

In the realm of self-hosted models, SQLcoder-70B Alpha and Wizardcoder 33B emerged as top performers, showing a commendable ability to maintain higher accuracy even as question difficulty increased. The decline in accuracy with increased query complexity was lesser for these models compared to even Claude and OpenAI models. However, given these are fine-tuned models, there's a possibility that their exceptional accuracy could also stem from overfitting. This needs further investigation to distinguish between genuine model robustness and potential overfitting in NL-to-SQL tasks.

Cost ($)

The findings from our analysis reveal significant cost implications for hosting choices in deploying large language models (LLMs). Particularly, self-hosted environments demonstrate substantially higher operational costs compared to external platforms such as Anyscale, Bedrock, and Anthropic. This distinction largely stems from the difference in cost-calculation methodologies between the two hosting types.

Influence of Hosting Environment on Cost Efficiency:
The data underscores that self-hosted platforms, while offering potentially greater control over model operations, pose a heavier financial burden. This is principally due to their reliance on latency-based costing, where costs increase with the time taken to process inferences. Such a model contrasts sharply with external platforms, where costs are tied to the volume of data processed (i.e., the number of input and output tokens), which tends to be more predictable and easier to manage.

Comparative Analysis of Cost Across Different Models:
The cost variations observed among different LLMs—such as Codellama70b, DBRx Instruct, and Mixtral 8x7b—highlight the impact of model size and complexity on operational expenses. Larger and more complex models require more resources, which in turn increases the cost, especially in a self-hosted setup where latency can significantly affect pricing.

For organizations considering the deployment of LLMs, these findings suggest that a strategic evaluation of hosting environments is crucial. Organizations may find that external hosting solutions offer a more cost-effective approach, particularly when deploying models that are expected to handle large volumes of queries or require high availability. The trade-off between control and cost must be carefully weighed to align with the organization's operational priorities and budget constraints.

Recommendations for Future Deployments:
It is advisable for entities looking to utilize LLMs to conduct a thorough analysis of anticipated workloads and latency profiles under different hosting scenarios. Such insights could inform better decision-making regarding whether to opt for self-hosting or to leverage the economies of scale offered by external platforms.

This study work also opens avenues for further research into optimizing cost and operational efficiency in LLM deployments. Future studies could explore the viability of hybrid hosting solutions, where critical operations are managed in self-hosted environments and less sensitive tasks are offloaded to cost-effective external platforms. Additionally, advancements in model efficiency and reduction in latency through technological improvements could further alter the cost dynamics explored in this report.

Conclusion

This analysis of Execution Accuracy (EX), Throughput, and Cost across various Large Language Models (LLMs) applied to SQL query generation has shed light on the capabilities and limitations of these technologies. Key findings highlight that Anthropic's Claude3-Opus and OpenAI's GPT-4 Turbo lead in accuracy, demonstrating the advanced potential of LLMs in handling complex tasks.

The study reveals significant cost differences, with self-hosted models incurring higher expenses due to latency-based pricing, whereas external platforms offer more predictable costs tied to data throughput. This emphasizes the need for strategic hosting choices to balance operational control and cost efficiency.

Future research should explore hybrid hosting solutions and model optimizations to improve cost and performance. Entities using LLMs should consider their specific application requirements and operational sustainability when selecting models and platforms.

In summary, while leading models show high proficiency in SQL query generation, the diverse performance and cost landscape indicates the importance of careful model and platform selection to leverage the full potential of LLM technologies in practical applications.

References

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. Advances in neural information processing systems, 33, 1877-1901.

Iyer, S., Konstas, I., Cheung, A., & Zettlemoyer, L. (2018). Mapping language to code in programmatic context. arXiv preprint arXiv:1808.09588.

Li, J., Hui, B., Qu, G., Yang, J., Li, B., Li, B., ... & Li, Y. (2024). Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. Advances in Neural Information Processing Systems, 36.

Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., ... & Radev, D. (2018). Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. arXiv preprint arXiv:1809.08887.