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

By Petavue Research

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 LLMs fare across different hosting platforms—such as AnyScale, Bedrock, Anthropic, OpenAI versus self-hosted environment—remains largely absent. This overlooks how hosting platforms can significantly influence model performance, a gap that is crucial for informed deployment and resource optimization.

Moreover, most existing efforts have narrowly focused more on accuracy as one of the key metrics for assessing LLM efficacy. Often they were not interrelated with 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 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 knowledge, our research aims to provide a comprehensive evaluation of LLMs' performance, focusing on 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, Anthropic family of 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	Llama3-70B Llama2-70B Mistral 7B (Version 1) Mixtral8x7B CodeLlama-70B
Amazon Bedrock	Llama3-70B Llama2-70B Mistral-7B (Version 2) Mixtral8x7B Claude3 - Sonnet Claude3 - Haiku
OpenAI	GPT3.5 Turbo GPT4 Turbo GPT4 omni GPT4o Mini
Anthropic	Claude3 - Opus Claude3 - Sonnet Claude3 - Haiku Claude3.5 - Sonnet
Google	Gemini 1.0 Pro
Self-Hosted	DBRX Instruct Llama3-70B CodeLlma-70B CodeLlma-34B Mistral-7B (Version 1) Mistral-7B (Version 2) Mixtral8x7B Sql Coder 7B2 Sql Coder 70B Alpha WizardCoder-33B

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

Results

Our benchmark study focused on 3 metrics: (1) Execution Accuracy (2) Throughput, and (3) Total Cost of running the models (compared token sizes and inference time).

Execution Accuracy (EX)

We used Execution Accuracy (EX) as the metrics for measuring the accuracy of SQL queries generated by LLMs . This approach involves evaluating the result table of LLM generated queries (table 2) against the table of expected SQL queries given by the BIRD dataset (table 1). This means that the script will search for occurrences of each column of table 2 in table 1 with the same order of values. To evaluate this logic, we developed a custom evaluation script by optimizing BIRD's existing script. This optimization aimed to address the limitations of BIRD's script by modifying it to rectify the scenarios where it underperformed.

Overall EX% averaged across 1800 inferences

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 run per model.

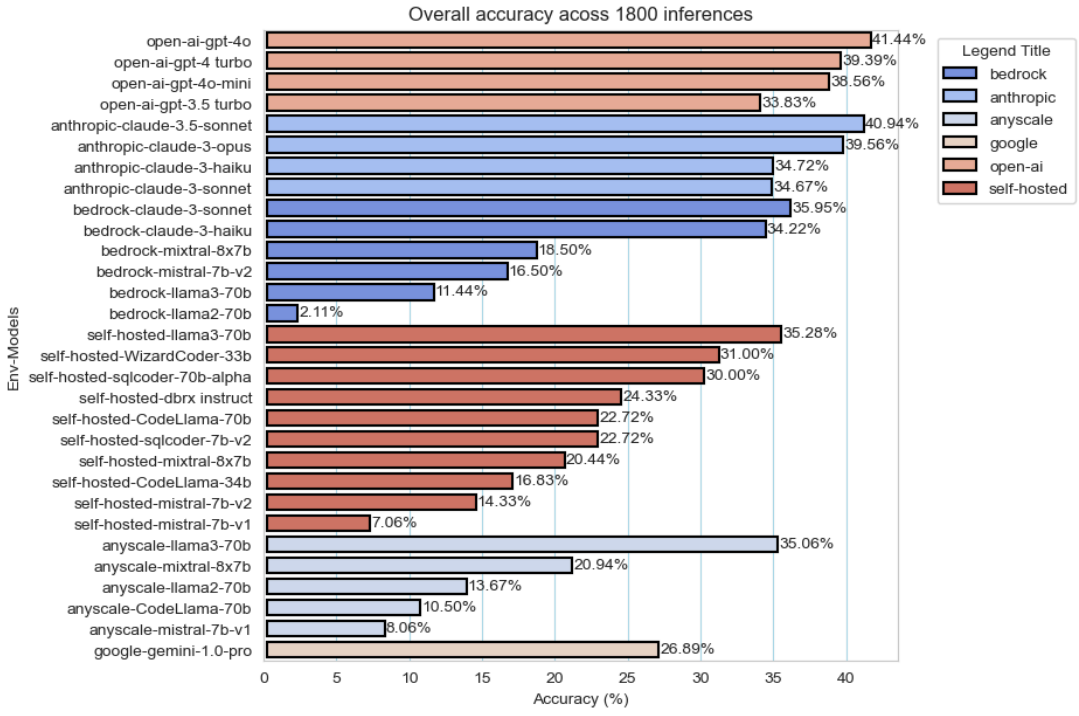

Figure 1. Average EX of each model compared to others.

We found that OpenAI's GPT-4o, Anthropic's Claude-3.5 Sonnet, Claude-3 Opus and OpenAI's GPT 4 Turbo are the top performing models with the highest accuracy of 41.44%, 40.94%, 39.56% and 39.39%

respectively (Figure 1). The latest model released by OpenAI, GPT-4o Mini, performs at 38.56%, just a few percentage points behind the top models. Following this are the Claude3-Sonnet (35.95% 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%). Llama3-70B stood at 35% accuracy in both Self-hosted and Anyscale. Self-hosted models such as Wizardcoder 33B and SQLcoder-70B-alpha achieved accuracy of 31% and 30%.

Comparison of models run in multiple platforms

One of the key goals of this study was to find out how the same model when hosted in different platforms varies in terms of accuracy. Interesting results have emerged from this analysis (Figure 2). First, for Claude-3 (Sonnet and Haiku) models we noticed only marginal differences in accuracy when hosted between Anthropic and Bedrock. Sonnet's accuracy is marginally higher in Bedrock (35.95%) than in Anthropic (34.67%). In Haiku performances are nearly identical in Bedrock (34.22%) and Anthropic (34.72%).

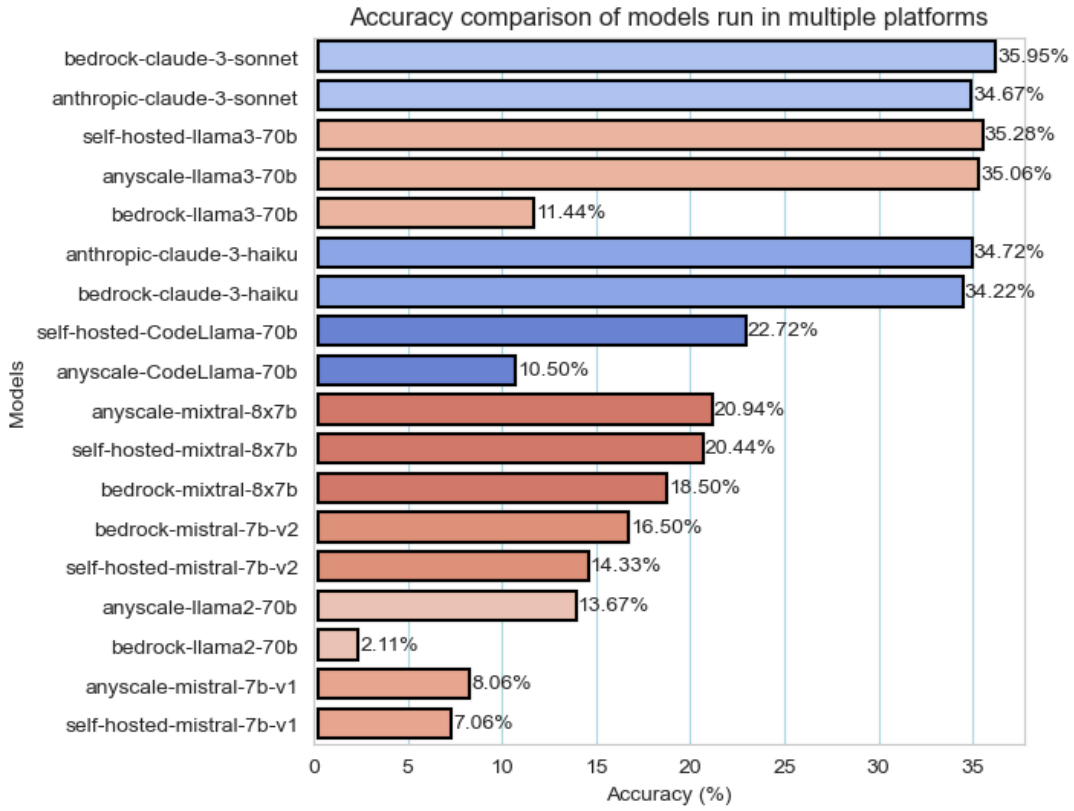

Figure 2. EX % comparison of models between platforms.

But the biggest difference between platforms was observed for Llama3-70B and Llama2-70B. For Llama3-70B in Anyscale and self-hosted we got an accuracy of 35% versus 11.4% in Bedrock. Similarly for Llama2-70B, we got an accuracy of 2.11% on Bedrock versus Anyscale at 13.67%, revealing a nearly

6 times difference. Similarly, a big difference was also observed for Codellama-70B when hosted on Anyscale (10.5%) versus self-hosted (22.72%), more than a 2x times difference. Also, the models of Mistral including Mistal-7B versions and Mixtral-8x7B, exhibited a significant difference mainly between Bedrock and when self-hosted. 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-hosted compared to Bedrock (18.5%).

Impact of question difficulty on accuracy

The sample chosen from the BIRD dataset has questions with three levels of difficulties: simple, moderate and challenging. We compared the accuracy in relation with these levels.

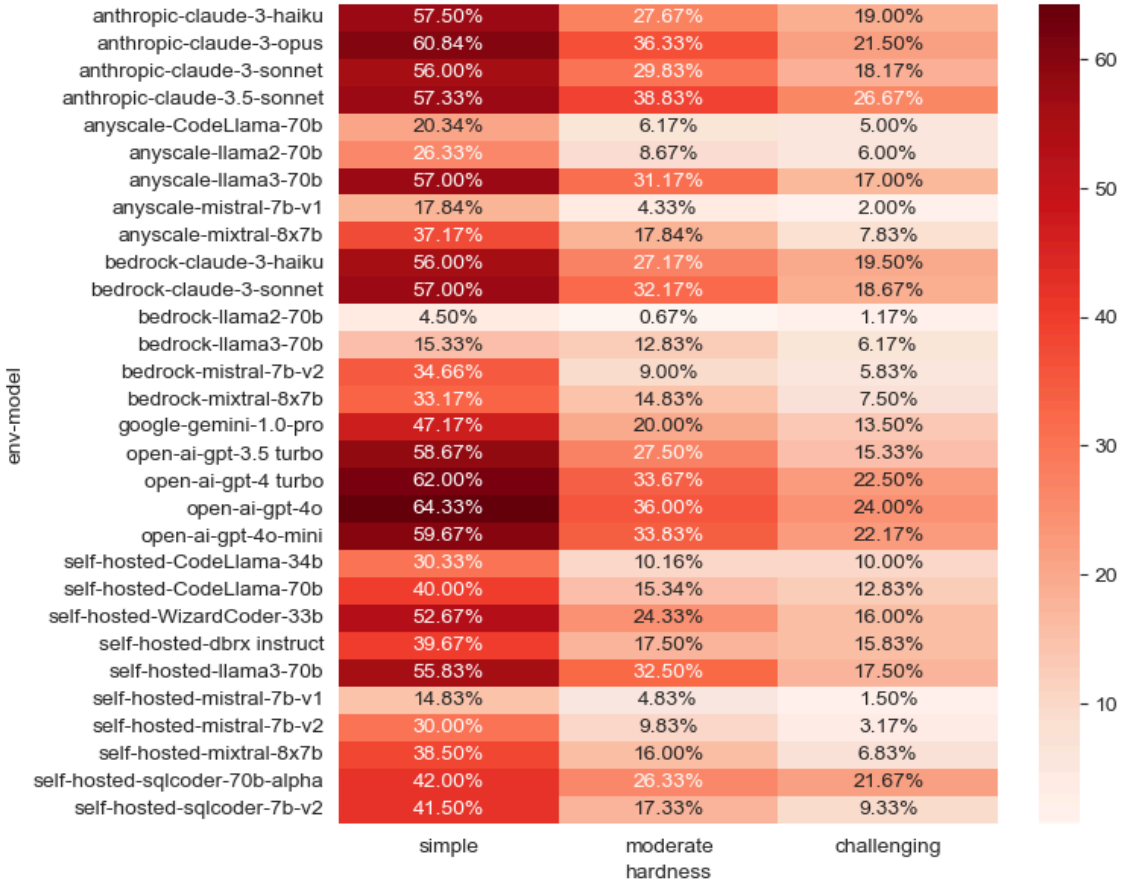

Figure 3. EX % across 3 levels of Question Hardness.

We found that across models, accuracy dropped when question hardness level increased from simple to moderate, and from moderate to challenging (Figure 3).

In simple level NL-to-SQL tasks, GPT-4o was observed to be the top performing model at 64.33%. This is followed by GPT-4 Turbo (62%) and Claude3-Opus (60.84%). Interestingly, we observed that Claude3-Haiku has achieved a higher accuracy (57.5%) compared to Sonnet (56%) for simpler tasks, even though Sonnet is a superior model. The latest models, Claude-3.5 Sonnet and GPT-4o Mini, stand at 57.33% and 59.67%. GPT-4o Mini performed better than GPT-3.5 Turbo (58.67%) on simpler tasks. Amongst those self-hosted, Wizardcoder-33B achieved the highest accuracy of 52.67%, followed by SQLcoder-70B Alpha (42%). However, none of the self-hosted models were able to achieve the level of GPT-4 models or Opus.

When moved to moderate level tasks, we observed accuracy drops for all models. Claude-3.5 Sonnet was the best performing model with accuracy of 38.83%. This is followed by Claude-3 Opus (36.33%) standing at par with GPT-4o (36%), followed by GPT-4o Mini (33.83%) and GPT-4 Turbo (33.67%). Claude3-Sonnet (32.17% in Bedrock & 29.83% in Anthropic) outperformed Haiku (27.17% in Bedrock & 27.67% in Anthropic), contrary to what was observed in simpler tasks. Amongst self-hosted models, Sqlcoder-70B alpha (26.33%) achieved highest accuracy followed by 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.

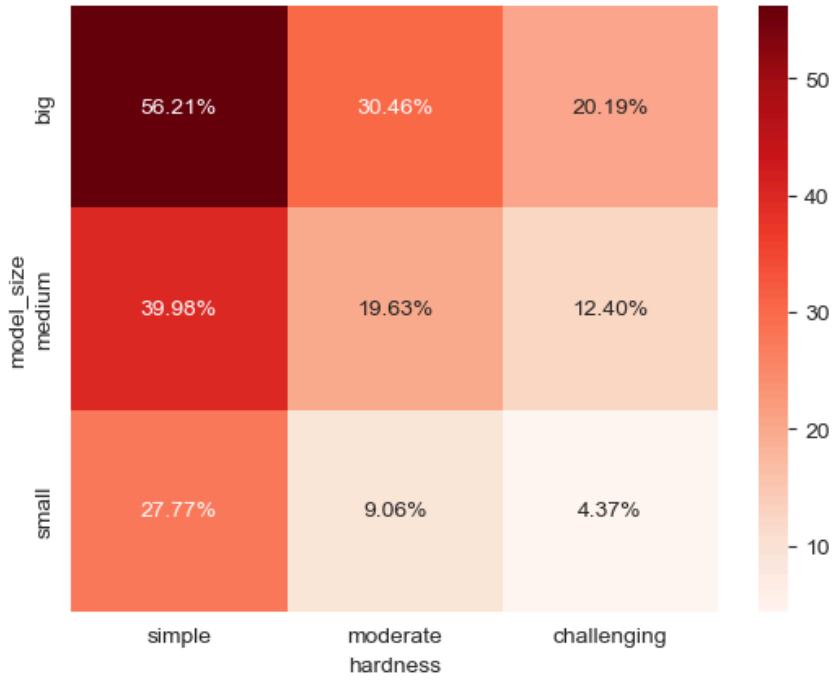

Figure 4. EX % shown in relation between model sizes vs hardness level of queries.

In challenging tasks, Claude-3.5 Sonnet achieved the highest accuracy of 26.67% compared to any other model, followed by GPT-4o (24%). This is followed by GPT-4 Turbo (22.5%), GPT-4o Mini (22.17%),

Opus (21.5%) and SQLcoder-70B alpha in self-hosted (21.67%). The accuracy drop is observed across all the models. Claude 3.5 Sonnet performed better than GPT-4o in challenging tasks.

We examined the correlation between model size and query difficulty levels (Figure 4). Our analysis revealed clear relationships aligning with logical expectations. Across all tasks, larger models consistently outperformed medium and smaller models. Additionally, as query complexity increased, performance declined for models of all sizes.

Impact of number of instructions provided in the prompt on accuracy

At the next level, we analyzed how model accuracy varies when the number of instructions given to the prompt are increased in the set of 0, 5, 7, 9 to 11 (Appendix A). Unlike what was observed with respect to hardness, varying the number of instructions did not have a striking impact on accuracy of models (Figure 5).

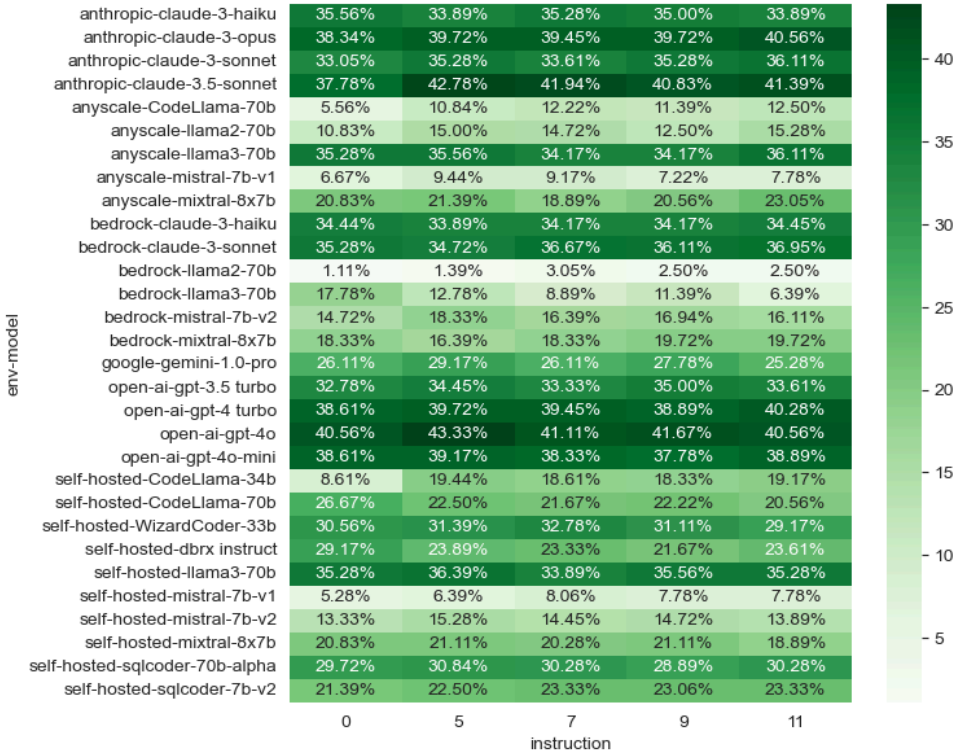

Figure 5. EX % across different sets of instructions provided to the prompt.

Also, we noticed that the extent of fluctuations in accuracy of models when instruction sizes are increased, changes from model to model and platform to platform. For example, in Anyscale, we observed that for all 4 models accuracy increased when 5 instructions were provided compared to no instructions. 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%) and Mistral-7B-V1 peaked in the trial of 5 instructions (9.44%). Similarly fluctuations were observed in Bedrock, Open AI and Anthropic (Figure 6 & 7). However the extent of fluctuations observed in Mistral or Llama2 family models are more than what is observed in higher sized models such as GPT-4 Turbo, GPT-4o and all Claude-3 models.

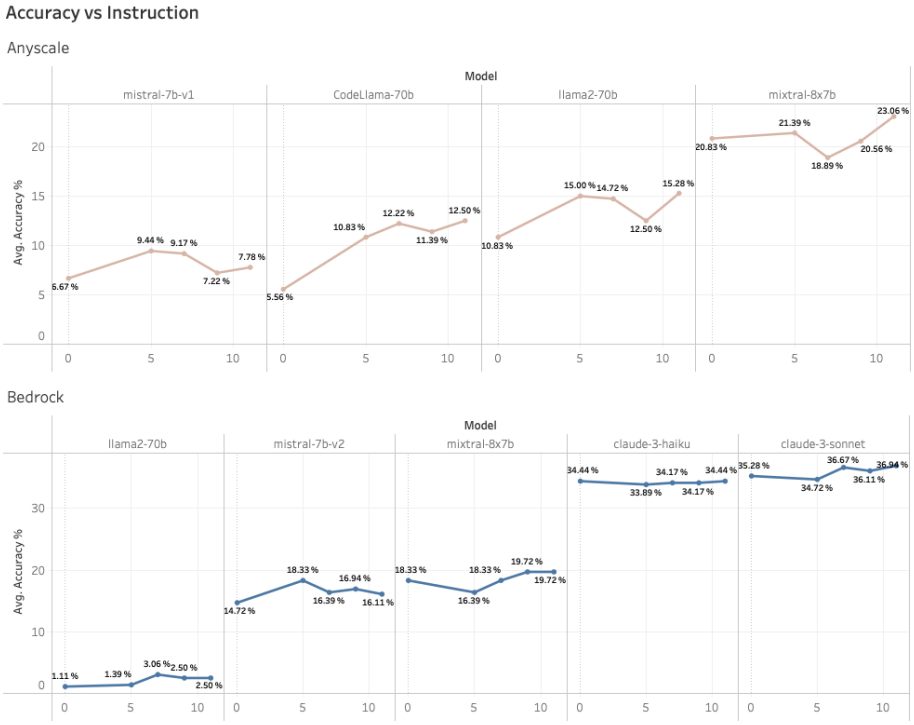

Figure 6. EX % comparison between models within Anyscale and Bedrock. The chart shows how EX changes from 0 to 11 instructions.

Similarly self-hosted models fluctuate a lot when we increase instructions from 0 to 11 (Figure 7). For example, for models such as Wizardcoder, Sqlcoder-70B, Sqlcoder-7B-2, Mixtral 8x7B, Mistral 7B-V1 and V2, the accuracy increased drastically from no instruction to 5 instructions. For DBRX Instruct and Codellama-70B, we noticed a massive drop in accuracy from 0 to 5 instructions, from 26.67% to 22.5% and 29.17% to 23.33%, respectively.

Overall, we observe that each model attained its own peak accuracy irrespective of the number of instructions provided, exhibiting no pattern or consistency in accuracy in relation to number of instructions. However, the extent of fluctuation in accuracy also varied drastically in self-hosting compared to other platforms and higher sized models exhibited relatively lesser fluctuation in accuracy.

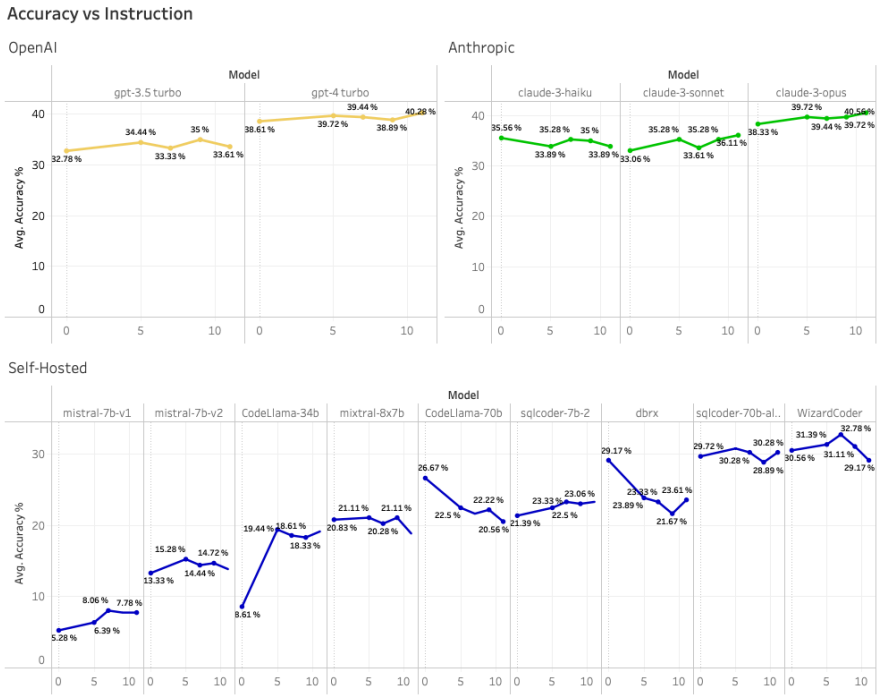

Figure 7. EX % comparison between models within OpenAI, Anthropic and Self-Hosted. The chart shows how EX changes from 0 to 11 instructions.

Cost

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

Total cost incurred for running each model for 1800 inferences

Our analysis revealed that self-hosted models without inference engines incurred substantially higher costs compared to those on other platforms. Codellama70b and DBRx Instruct were the most expensive, costing $94.82 and $88.81 respectively (Figure 8). Mixtral 8x7b and Coellama 34b also showed significant expenses. Among non-self-hosted models, Anthropic's Claude-3 Opus led at $47.3, followed by GPT-4 Turbo at $22.61. Recent releases like GPT-4o ($11.69), Claude-3.5 Sonnet ($10.36 on Anthropic), and GPT-4o Mini ($0.36) demonstrated markedly lower costs than their predecessors. Notably, the GPT-4o model, at $11.69, cost roughly half of GPT-4 Turbo while offering improved accuracy. This trend of more affordable models with higher accuracy is currently a focal point in LLM discussions. In platform comparisons, Anyscale-operated models showed the lowest expenses overall.

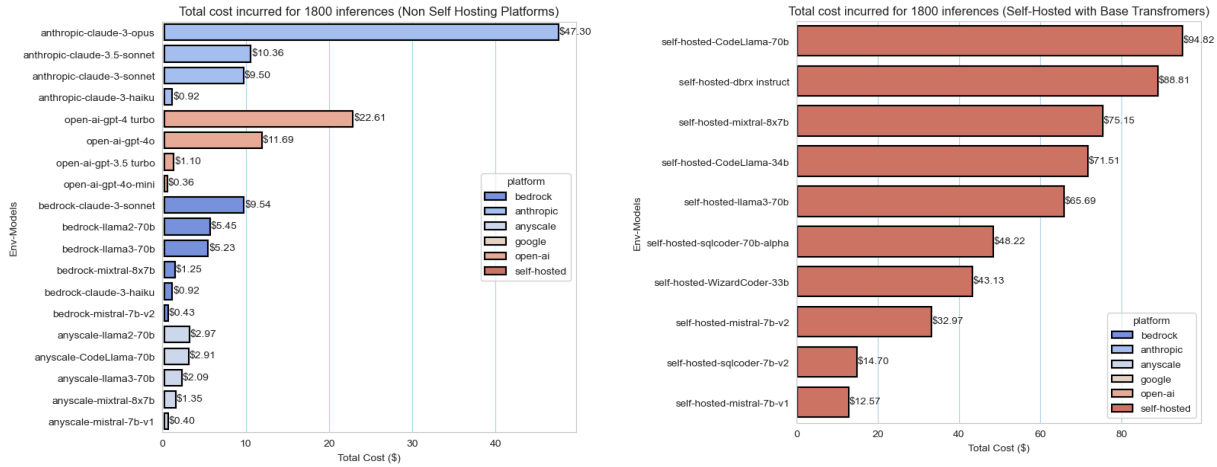

Figure 8. Total cost incurred to run all models for 1800 inferences. On the left is the total cost spent for non-self hosted models. On the right is the total cost spent for self-hosted models with base transformers.

Cost comparison of models run in multiple platforms

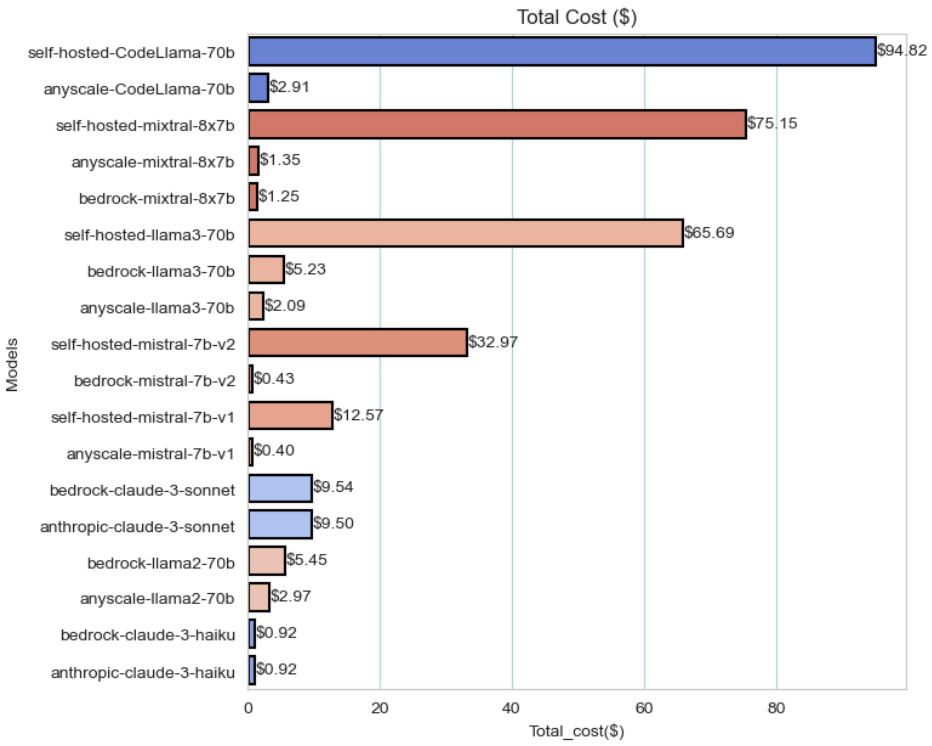

Figure 9. Total cost spent to run the models between platforms (self hosted vs other platforms)

From both Figure 8 and 9, we can observe that there exists a significant disparity in costs between self-hosted and other hosting platforms. For instance, the codelama70b model on a self-hosted platform incurred 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 required $75.15 to run on a self-hosted setup. While the cost differences among Anyscale, Bedrock, and Anthropic platforms are minimal, self-hosting significantly escalated the costs.

The stark cost disparity between self-hosted and other platforms can be attributed to the differing method of cost calculation. For non-self-hosted platforms, costs are determined by the number of input and output tokens used to run the models. Conversely, for self-hosted models, costs are calculated based on total time taken to run the model on GPU servers. This is otherwise called inference time or network latency - which is the time taken to process the inferences. Because of this, it is important to look into the cost comparison separately between self-hosted and non-self hosted platforms.

Total cost incurred for non self-hosted platforms

From Figure 9, we can see that amongst non self-hosted models, Anthropic's Claude3 Opus turned out to be the most expensive model ($47.3) followed by GPT-4 turbo ($22.61) and GPT-4o ($11.69). For models that are not hosted on different platforms, the cost is calculated by total number of input tokens and output tokens. Below is the formula for cost of a model:

Total Model Costs = {(Total number of input tokens x Cost per input token) **+** (Total number of output tokens x Cost per output token)}

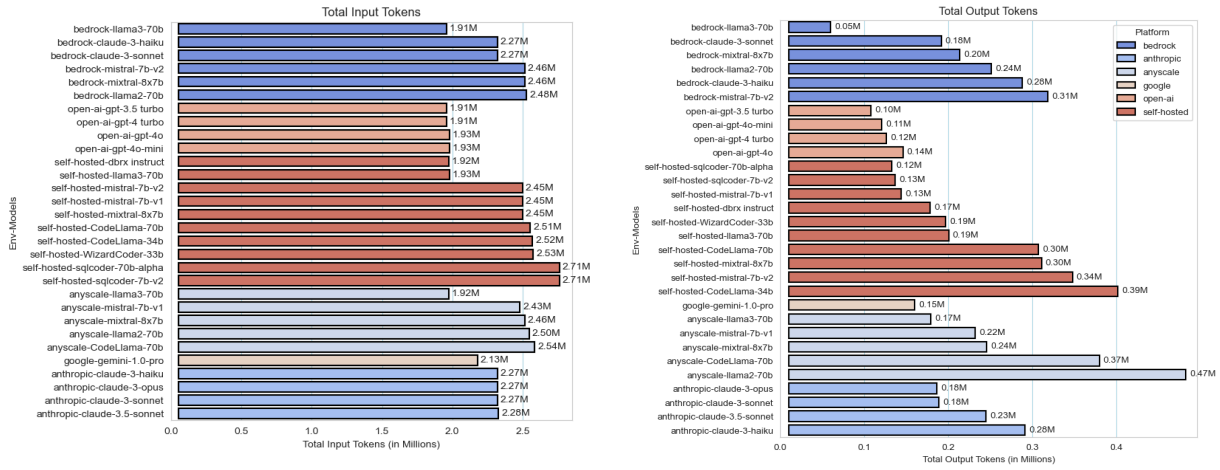

Figure 10. Total number of input tokens and total output tokens used for running all models across 1800 inferences.

From Figure 10, we can note that, for non self-hosted models, the number of input tokens vary from one another as each model uses its own method of tokenization. Similarly, models also vary from one another in terms of number of output tokens generated. As a result this will impact the difference in total cost incurred. In terms of output tokens, we can observe that Open AI models have the least number of output tokens compared to other platforms. In Appendix B, we have provided the cost per million tokens (as on 15th May 2024) for each model in various platforms.

Total cost incurred for self-hosted models

Unlike other platforms, costs of self-hosted models are calculated based on the time taken to run inferences, otherwise called network latency. From Figure 9, we can note that the total cost of running a model in a self-hosted setup is significantly higher than when they are run on a cloud platform such as Bedrock or Anyscale. This is primarily due to higher latency of running the model on self-hosted compared to Bedrock or Anyscale as shown in Figure 11. Observe that CodeLlama-70b and DBRX Instruct have the highest network latency (21.11 sec and 17.78 sec, respectively). As a consequence, the total cost spent is also highest for them (Figure 8).

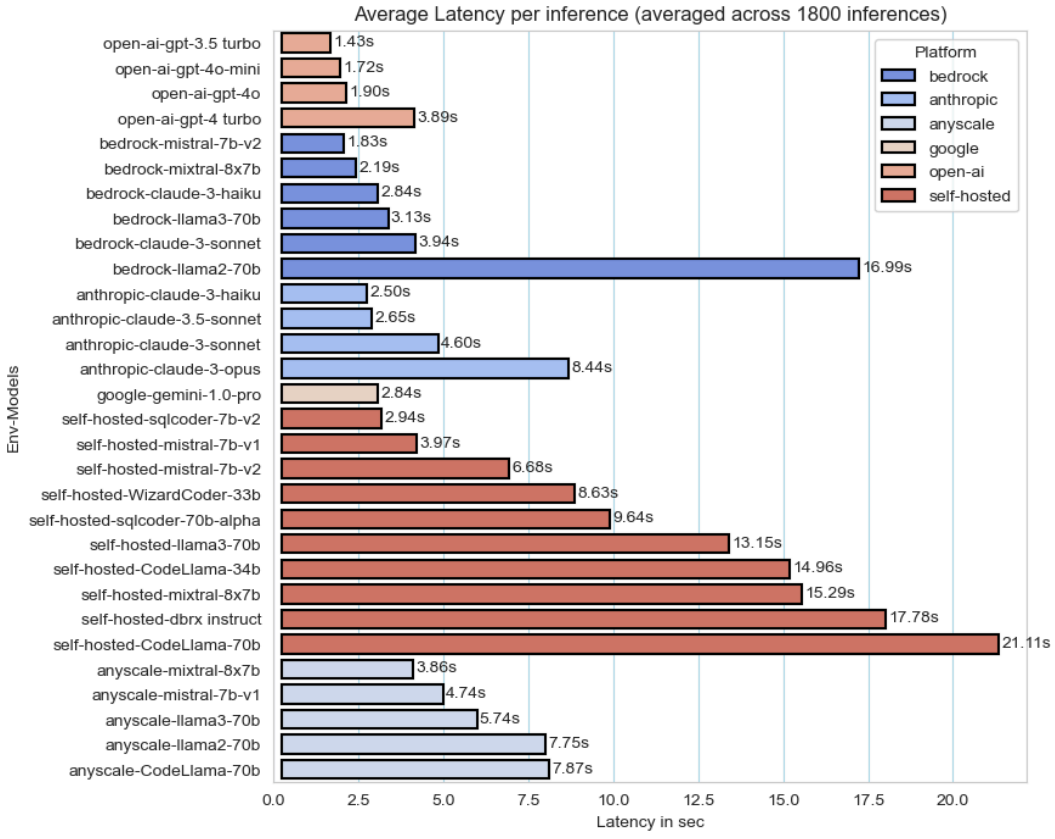

Figure 11. Average latency per inference.

But note that the latency of CodeLlama70b on Anyscale is only 7.87 sec compared to 21.11 sec on self-hosted. This higher latency on self-hosting compared to other platforms could be attributed to the fact that while self-hosting, we only used the inherent versions of the models that come with base transformers and without any inference engines applied above them. On the other hand, a platform such as Anyscale or Bedrock do perform certain optimizations and use inference engines to reduce the network latency and get best possible performance.

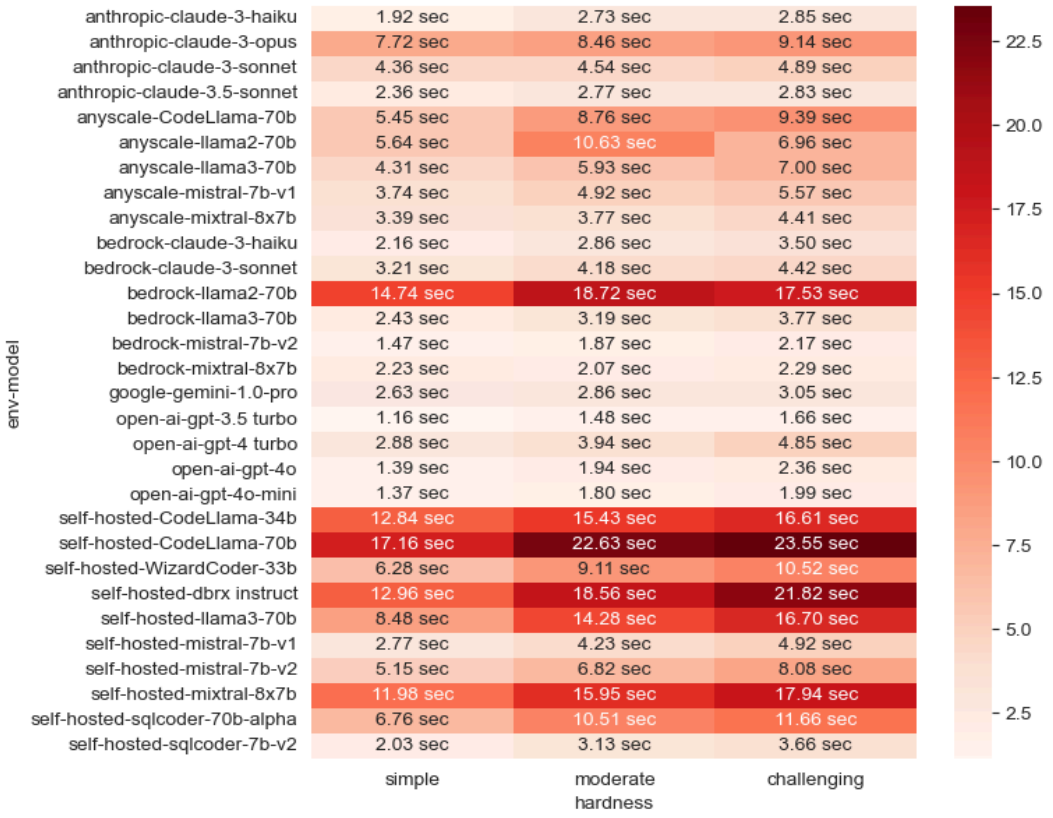

Figure 12. Average latency per inference based vs hardness level of questions

Impact of question hardness and number of instructions on total cost spent

Query hardness is found to have a significant impact on cost in the self-hosted models as shown in Figure 13. This is primarily due to the fact that harder questions took more time to process with more latency. Higher latency impacted the cost directly. Also amongst non self-hosted models it is observed that questions of moderate level hardness cost marginally more than complex questions, as the total number of input tokens were found to be higher for moderate questions compared to the complex ones.

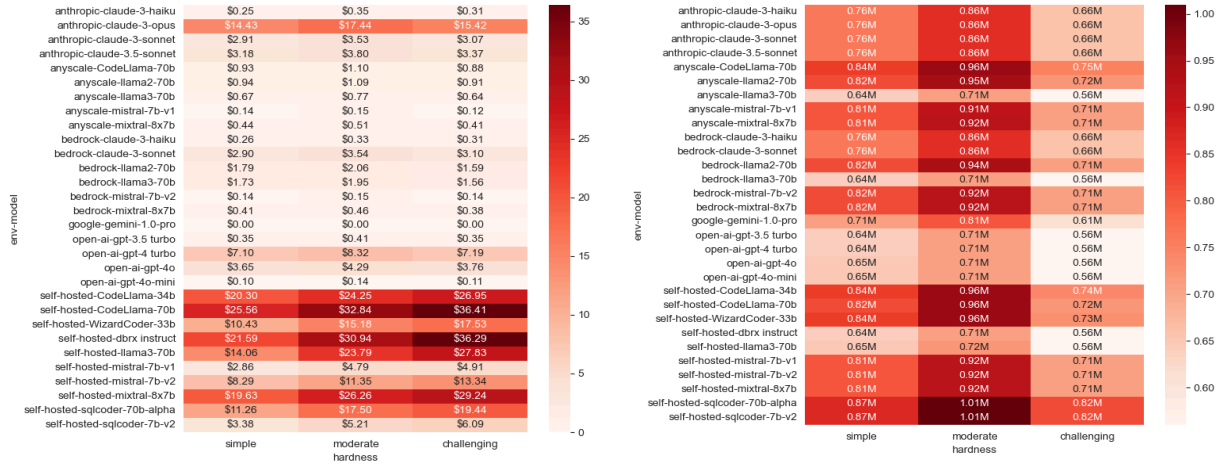

Figure 13. Total cost incurred for all models and total number of input tokens across different levels of query hardness.

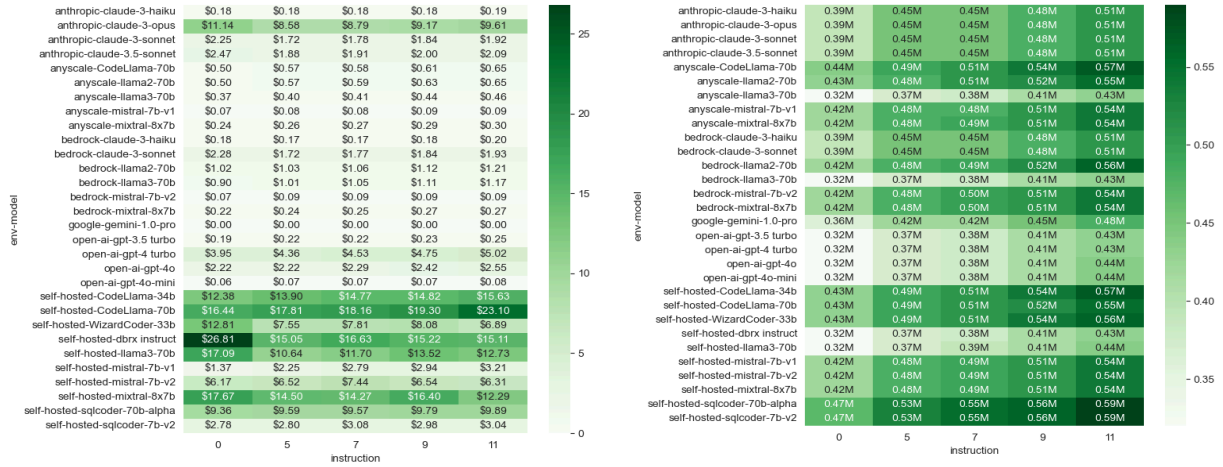

Figure 14. Total cost incurred for all models and total number of input tokens across different levels of instructions.

The number of instructions also had a marginal impact on the total cost across all models. For all non self-hosted models, increasing the number of instructions increased the cost spent. This is because the total number of input tokens is directly proportional to the number of instructions provided in the prompt as shown in Figure 14. However, in self-hosted models the total cost was not directly impacted by the size of the instructions and the extent of fluctuation in cost between instruction sets also varied from model to model.

Model accuracy versus total cost spent

When evaluating LLMs based on their accuracy relative to the total cost incurred, it becomes evident that a higher financial investment does not necessarily translate to superior accuracy. For instance, substantial sums were allocated to self-hosted models. Yet their accuracy was lower in comparison to larger models from companies like OpenAI or Anthropic (especially when self-hosted without inference engines). A striking example is the contrast between Codellama70b, which cost $94.82, and Claude3 Haiku, which cost a mere $0.92, but achieved significantly higher accuracy than Codellama70b (Figure 15). This observation underscores the pivotal role played by the platform on which the models are run, as it can profoundly impact their performance. However note that, even while self-hosting the cost and latency could significantly get lowered when inference engines are applied. However the extent of reduction is to be experimented in the future research.

Furthermore, models in the top-left quadrant of Figure 15, including GPT-4o, GPT-4o Mini, Claude-3.5 Sonnet, Claude-3 Sonnet, and Haiku, offer the best balance of accuracy and cost. These models provide higher accuracy at lower costs compared to GPT-4 Turbo and Claude-3 Opus. Among these, GPT-4o and Claude-3.5 Sonnet stand out as the most optimal choices, surpassing even Opus in accuracy while maintaining lower costs.

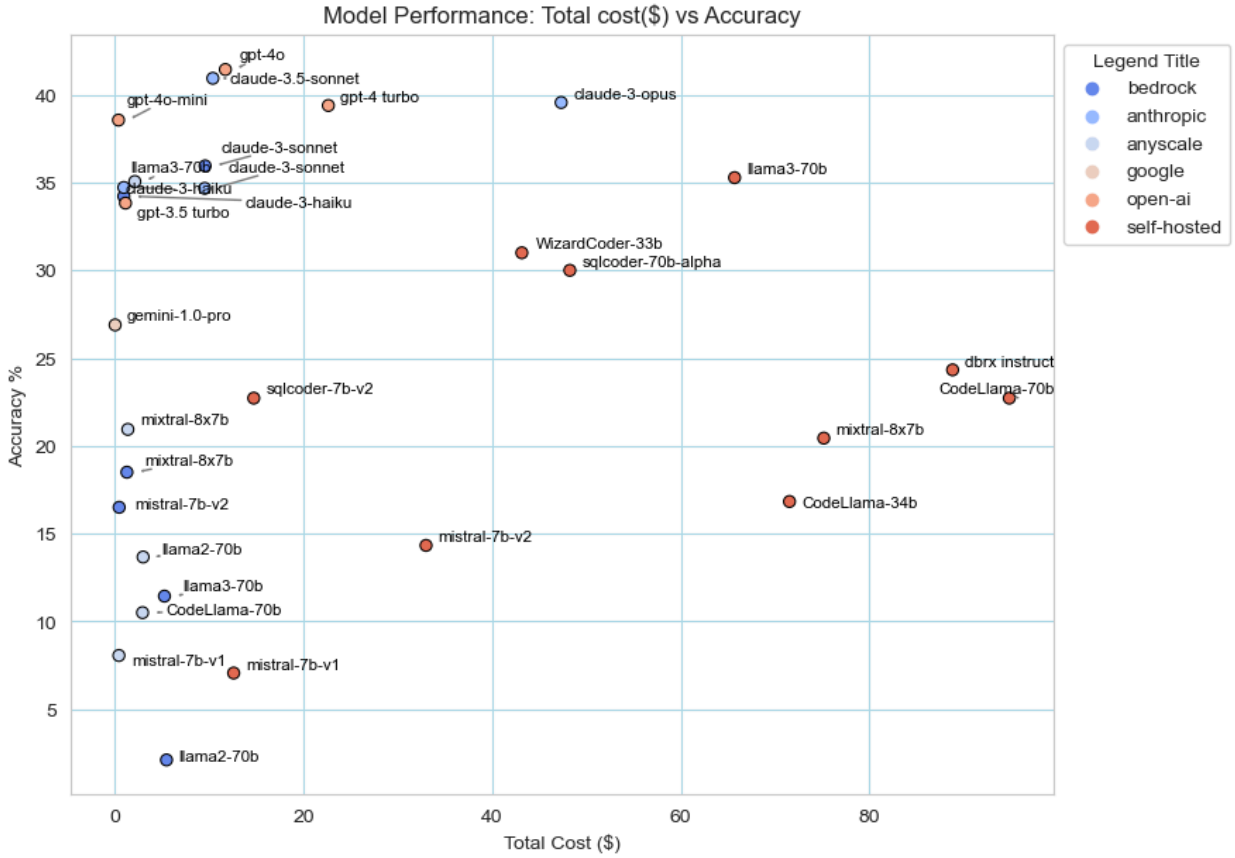

Figure 15. Total cost incurred for all models in relation to the model accuracy.

GPT-4o Mini and Claude-3 Haiku emerge as even more cost-effective alternatives. GPT-4o Mini outperforms Claude-3 Haiku with better accuracy at one-third the cost. Moreover, GPT-4o Mini offers a substantial cost advantage over GPT-3.5 Turbo and Haiku, with only a marginal difference in accuracy.

Throughput

In the context of LLMs, throughput is a metric that refers to the amount of text or the number of tokens that a model can process within a given time frame. It measures a model's efficiency and performance in handling text input and generating output. It is measured by the unit called - tokens per second (tokens/sec). A higher throughput means that the model can process more tokens in a shorter amount of time, which is desirable for applications and end users.

To evaluate the throughput of the LLMs in our study, we employed the following formula:

Throughput = Total Output Tokens / Elapsed Time (Including Network Latency)

Specifically, we divided the total number of tokens generated as output by the LLM by the total time taken for the model to process the input, perform inference, and produce the output, including the time required for data transmission over the network. This approach allowed us to quantify the throughput in terms of tokens per second, taking into account the impact of network latency, which is a crucial factor when assessing the performance of LLMs deployed in a platform server.

Average throughput per inference of all models across 1800 inferences

The study showed that the Mistral models on bedrock, specifically Mistral 7b v2 and Mixtral 8x7b, achieved the highest throughputs of 87.11 tok/sec and 55.31 tok/sec, respectively. Additionally, the Claude3-Haiku model on Anthropic exceeded the performance of Mixtral 8x22b with a throughput of 62.27 tok/sec (Figure 16). Amongst Open AI models, GPT-4o has higher throughput (40.32 tok/sec), even compared to GPT-3.5 turbo (36.24 tok/sec) and GPT-4o Mini (35.46 tok/sec).

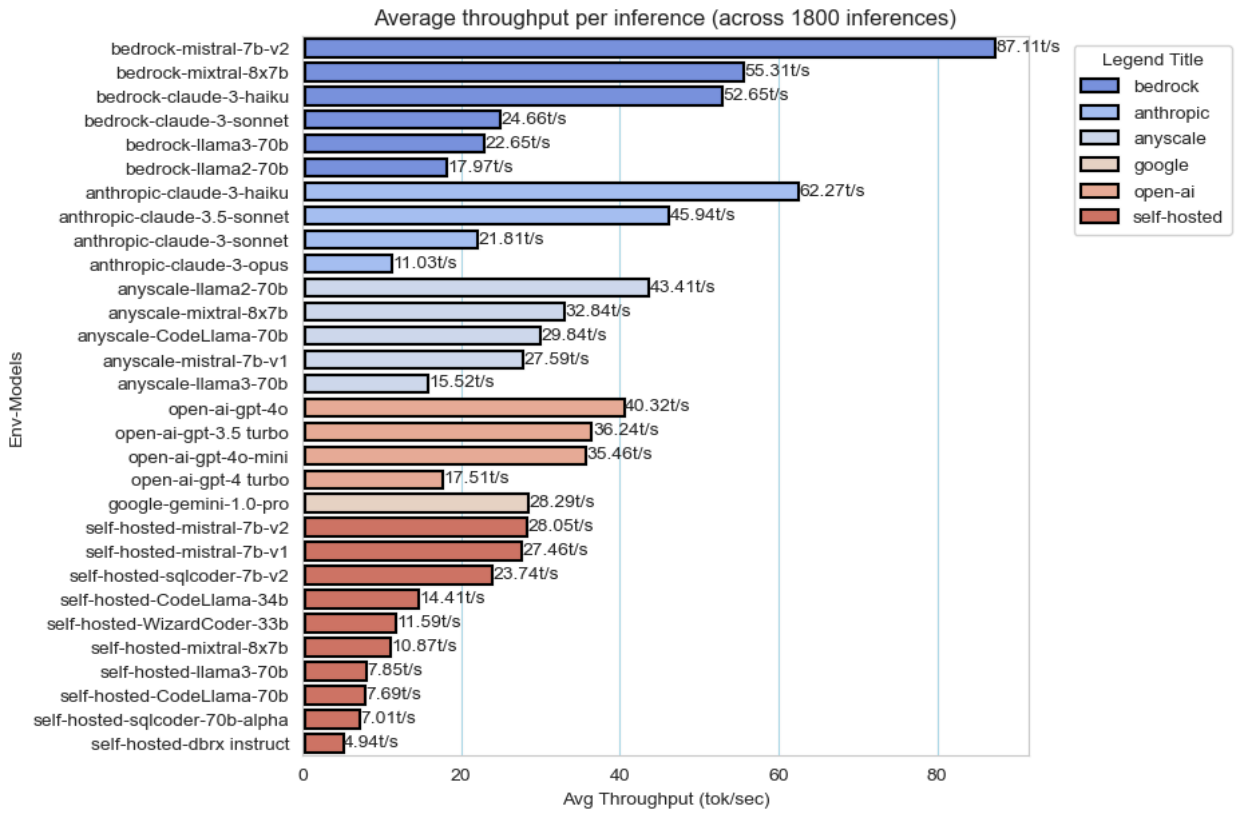

Figure 16. Throughput of all models across different platforms.

Throughput of models run in different platforms

We also observed variations in throughput for the same model across different platforms (Figure 17). For instance, the throughput of the Haiku model on Anthropic is 62.27 tok/sec compared to 52.65 tok/sec on Bedrock. Additionally, there is a significant difference in the throughput of the Mixtral 8x7b model, which stands at 55.31 tok/sec on bedrock, 32.84 tok/sec on Anyscale, and only 10.87 tok/sec when self-hosted. The lower throughput observed in self-hosted models can be attributed to the higher latency noted in previous sections, which considerably extends the processing and output production time.

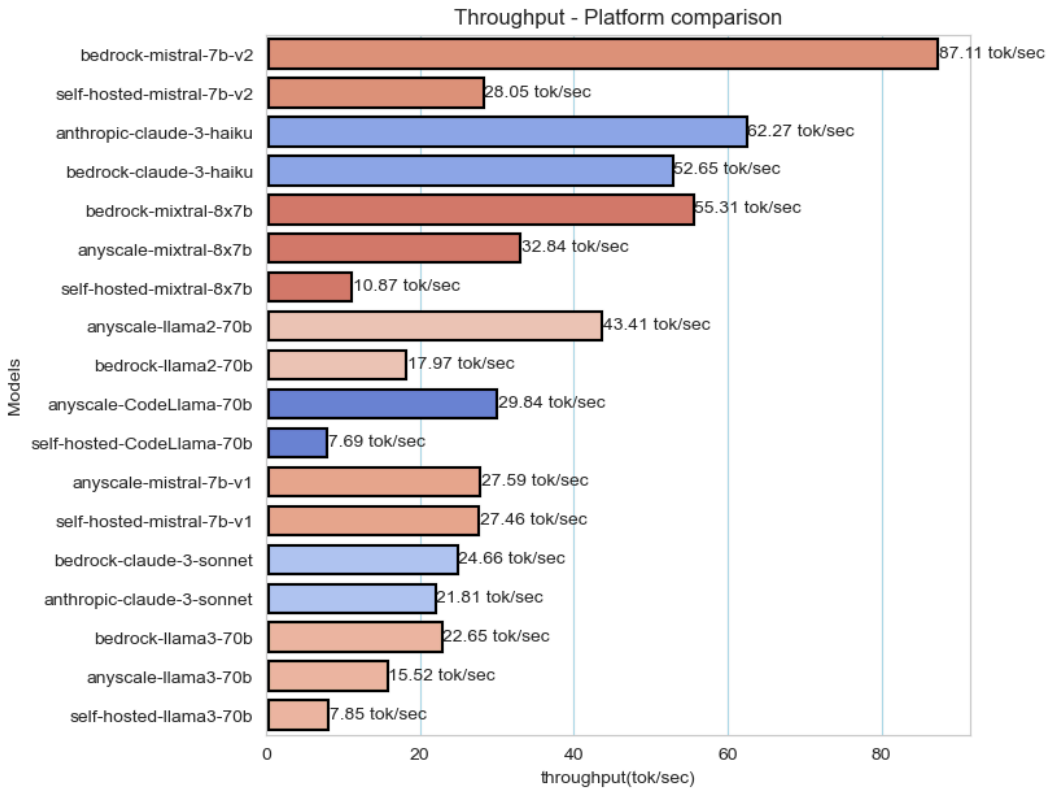

Figure 17. Throughput of all models across different platforms.

Variation of throughput across model sizes

From Figure 18, notice that throughput is lower for models of higher sizes. For example, this can be clearly observed in GPT-4 turbo (17.51 tok/sec), Claude3 Opus (11.03 tok/sec) and DBRX Instruct (4.94 tok/sec). Interestingly GPT-4o despite being a higher model has throughput (40.32 tok/sec) that is at par with models of medium sizes. Amongst medium sized models, Claude3 Haiku on Anthropic tops the list with 62.27 tok/sec.

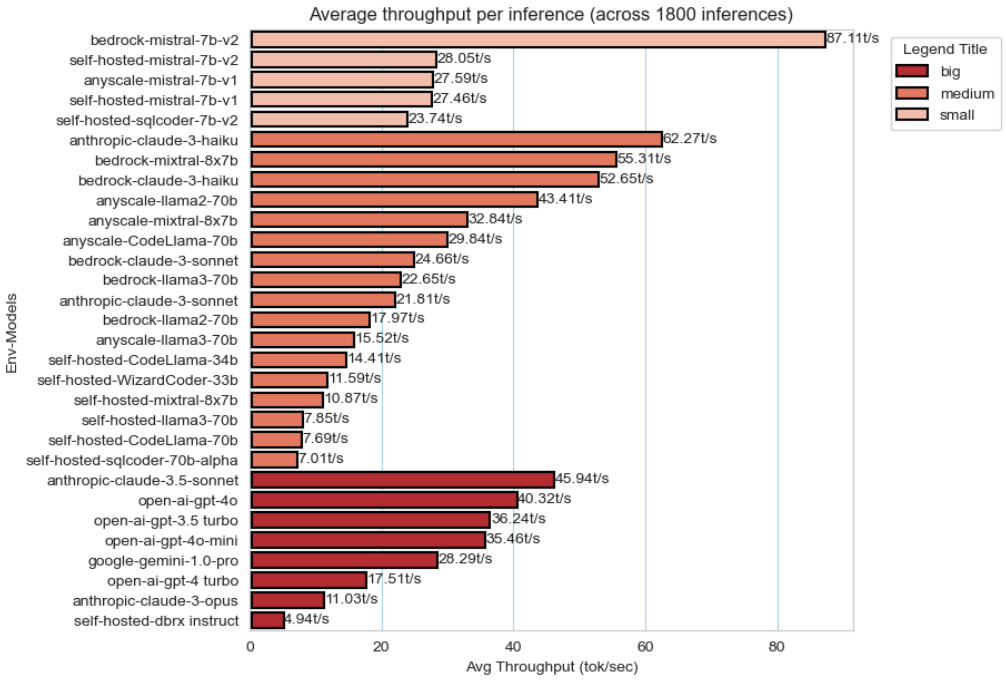

Figure 18. Throughput of all models across different platforms grouped by model size.

Impact of question hardness and number of instructions on throughput

Query hardness did have a marginal impact on the throughput of the models. Note that the throughput increases for all the models when the complexity of the query increases (Figure 19). This is primarily due to the increase in number of output tokens produced for more complex queries when we move from simple to harder questions. However, it was observed that the number of instructions provided did not impact throughput as shown in Figure 19.

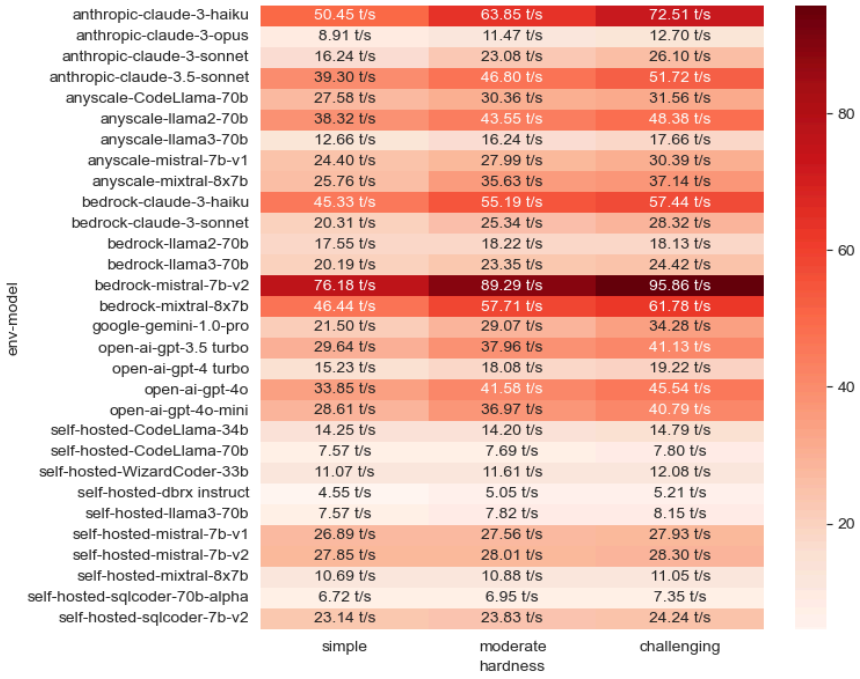

Figure 19. Throughput of all models across different query hardness.

Figure 20. Throughput of all models across different levels of instructions.

Model accuracy versus throughput

Analyzing the relationship between throughput and accuracy provides crucial insights. An ideal model for a user is expected to produce output of high accuracy, yet with high throughput. According to Figure 21, the Claude3 Haiku model on both Bedrock and Anthropic platforms stands out as an ideal choice, demonstrating superior throughput and accuracy compared to other models. However, if one is willing to accept a slight reduction in throughput for even greater accuracy, the Claude-3.5 Sonnet and GPT-4o emerges as the optimal choice compared to any other model.

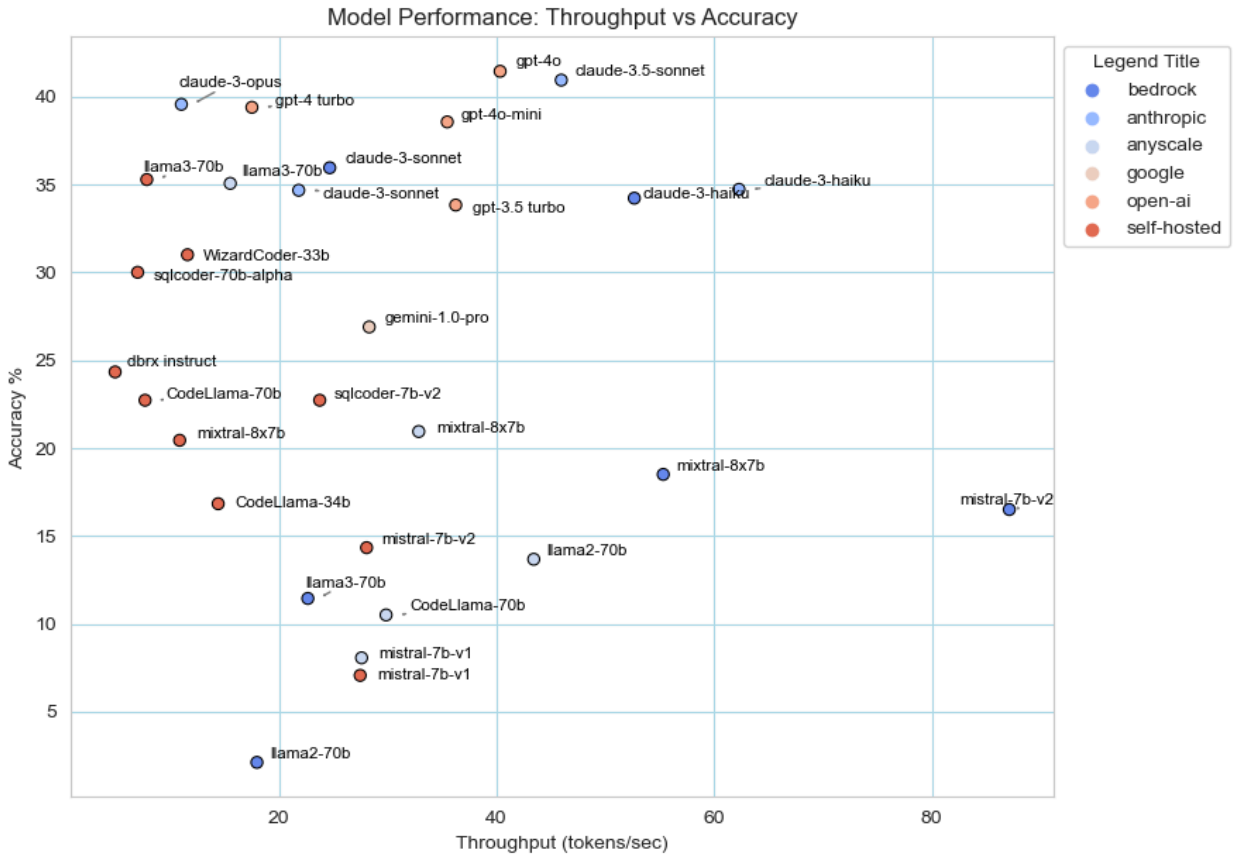

Figure 21. Average throughput per inference in relation to the model accuracy.

Total cost versus throughput

When selecting models based on higher throughput and cost-effectiveness, we observed that models from the Mixtral family, specifically Mistral 7b v2 and Mixtral 8x7b, offer the highest throughput at a lower cost. Additionally, the Claude3-Haiku model also presents itself as an ideal option, combining high throughput with affordability (Figure 22). From OpenAI, the GPT-4o stands out as a top model in performance while being more economical. This makes it a particularly attractive choice for those looking to balance cost with computational efficiency, making it suitable for a variety of applications where budget constraints are a consideration without sacrificing performance.

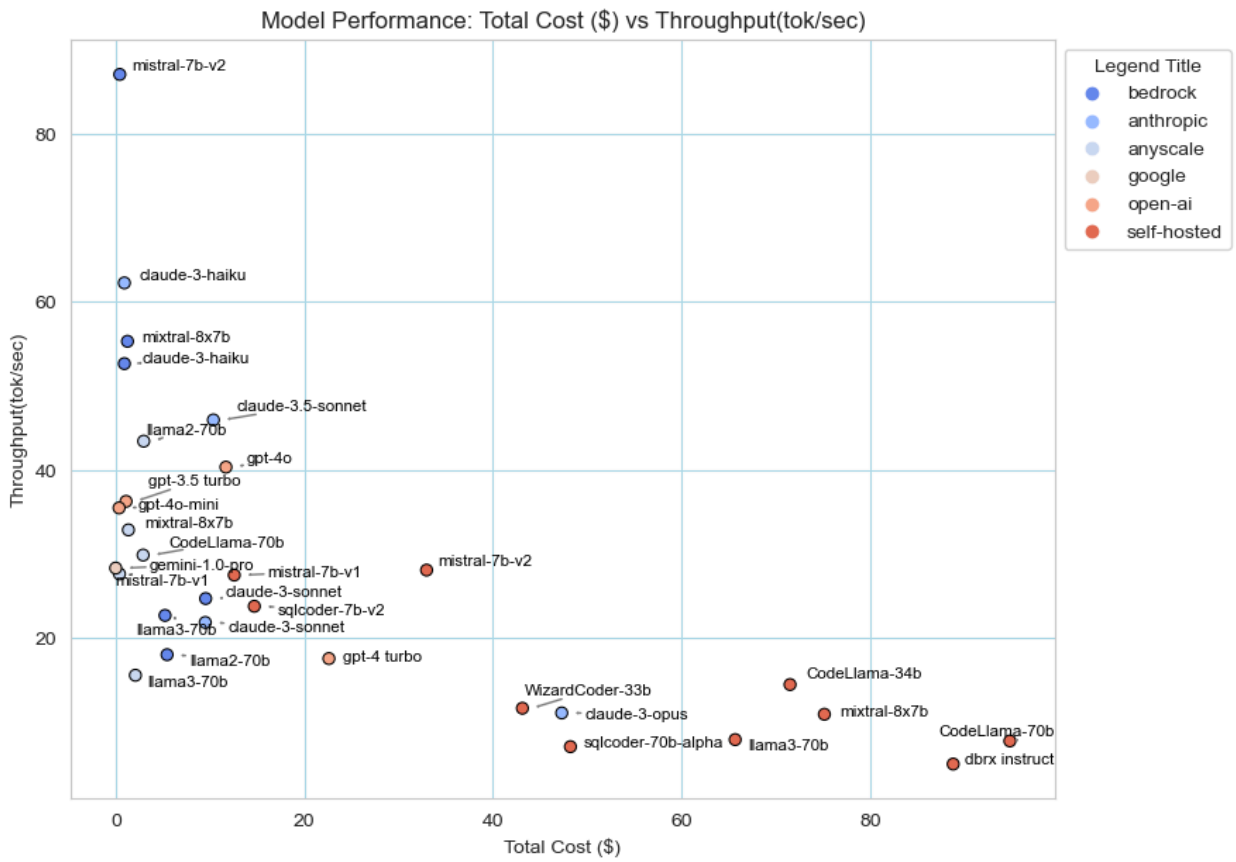

Figure 22. Average throughput per inference in relation to the total cost.

Discussion and Key Takeaways

This study revealed detailed results on performances of LLMs in the context of NL-to-SQL translation tasks, focusing on three crucial performance metrics: execution accuracy, throughput, and total cost. The study evaluated various LLMs across different hosting platforms, offering insights into the interplay between model performance and hosting platform factors. Here, we have elucidated the nuanced findings from our comprehensive evaluation, discussed the implications of these results, and identified potential areas for future research and optimization.

Model Performance and Platform Influence

Our comprehensive study on LLM performance in NL-to-SQL translation tasks has unveiled crucial insights into the intricate relationship between model architecture, hosting platform, and performance metrics. The results clearly demonstrate that model accuracy is not solely a function of architecture but is significantly influenced by the choice of hosting platform. OpenAI's GPT-4o and Anthropic's Claude-3.5 Sonnet consistently outshone other models, achieving impressive execution accuracies of 41.44% and 40.94% respectively. This exceptional performance across diverse hosting environments positions Claude-3.5 Sonnet as an ideal candidate for complex NL-to-SQL tasks.

The study also revealed intriguing platform-specific performance variations. While Claude3 models exhibited remarkable consistency across different hosting platforms, Llama2-70B and Llama3-70B models showed substantial performance fluctuations. For instance, Llama3-70B achieved 35% accuracy on Anyscale and self-hosted setups, but only 11.4% on Bedrock. These disparities highlight the significant impact of platform-specific optimizations and underlying hardware/software configurations on model performance.

Query Complexity and Model Size

Our analysis uncovered a clear correlation between query complexity and model accuracy. As queries progressed from simple to moderate to challenging, accuracy generally decreased across all models. However, larger models consistently outperformed their smaller counterparts across all difficulty levels. Notably, Claude-3.5 Sonnet demonstrated exceptional prowess in handling challenging tasks, even surpassing GPT-4o in this category.

Throughput Efficiency

Throughput emerged as a critical factor in our evaluation, measuring the models' ability to process inputs and generate outputs quickly. The Mistral family models, particularly Mistral 7b v2 and Mixtral 8x7b, showcased the highest throughput, especially when hosted on Bedrock platforms. This finding underscores the potential for platform-specific optimizations to significantly enhance model efficiency.

Conversely, self-hosted platforms generally exhibited lower throughput due to higher latencies, emphasizing the importance of considering hosting options when optimizing for speed.

Cost Considerations

Our study revealed significant cost implications associated with different hosting arrangements, a crucial factor for organizations considering the adoption of LLM technologies. Self-hosted models, particularly those without inference engines, incurred substantially higher costs due to increased network latency and longer inference times. In contrast, cloud-hosted solutions like Anyscale proved to be more cost-effective, potentially lowering the economic barriers to leveraging advanced LLMs for a wider range of applications.

Interestingly, recent releases such as GPT-4o, Claude-3.5 Sonnet, and GPT-4o Mini demonstrated lower costs than their predecessors while maintaining high accuracy. This trend towards more affordable, high-accuracy models is currently a key focus in the LLM landscape.

Impact of Latency

Latency emerged as a critical factor affecting both throughput and cost. Self-hosted environments faced significantly higher latency, directly translating to increased costs and reduced throughput. Cloud platforms like Anyscale demonstrated notably lower latency compared to self-hosted setups, highlighting the importance of carefully considering hosting environments in LLM deployment strategies

Ideal Model Selection

Our findings suggest that the ideal model selection involves a delicate balance of accuracy, throughput, and cost. The landscape has evolved significantly with recent model releases:

- Intense competition between OpenAI and Anthropic models since the emergence of GPT-4o, particularly in terms of cost-effectiveness.
- Claude-3.5 Sonnet currently leads as the best overall option in terms of performance:
    - At par with GPT-4o in performance. But surpasses GPT-4o in complex tasks.
    - Offers high accuracy with relatively low costs.
- Significant shift in the next performance tier:
    - Previously, Claude-3 Haiku dominated over GPT-3.5 Turbo
    - GPT-4o Mini has now disrupted this hierarchy
- GPT-4o Mini's impressive performance:
    - Performs only slightly below top-tier models
    - Dramatically more cost-effective
    - Effectively replaced Haiku as the go-to model for cost-effective performance
- GPT-4o Mini is the most economical option among high-performing models.
- This shift demonstrates:
    - Rapid evolution in the LLM space

- Newer models improve in accuracy while becoming significantly more cost-effective.

This evolving landscape underscores the importance of regularly reassessing model choices to optimize for both performance and cost-effectiveness in LLM deployments.

Conclusions and Future Directions

As LLM technologies continue to evolve, our study points to several promising avenues for future research:

1. Optimization techniques for self-hosted models: Investigating advanced inference engines and other optimizations to enhance cost-effectiveness, reduce latency, and improve throughput.
2. Platform-specific performance analysis: Further exploration of platform variability to develop more robust models that deliver consistent performance across different hosting conditions.
3. Retrieval-Augmented Generation (RAG) and fine-tuning: Examining how these techniques can influence accuracy, throughput, and cost metrics, potentially improving model responsiveness and efficiency for complex queries.

By focusing on these areas, we can continue to refine LLM models and their deployment strategies, enhancing their practical utility and making sophisticated language understanding technologies more accessible and effective across a variety of real-world applications. This ongoing research will not only optimize existing models but also pave the way for innovative approaches to LLM deployment in diverse settings.

Appendix A

List of instructions provided in the prompt

1. The answer generated must only be an SQL query ending with delimiter ";"
2. Dedicate time to understand the database schema fully, identifying the relevant tables and columns that align with the query's objectives.
3. Utilize only the data from the specified tables in the provided database schema.
4. Pay attention to case sensitivity in data and ensure the extraction of required information aligns precisely with the specified columns and tables.
5. Analyze the query's requirements to determine the appropriate use of GROUP BY, HAVING, and UNION clauses, ensuring they contribute to the accurate aggregation and segmentation of data.
6. Pay careful attention to the primary keys, foreign keys present in the database schema to determine appropriate columns for JOIN operations.
7. Apply WHERE clause conditions accurately to filter the dataset based on specified criteria.
8. Apply WHERE clause conditions accurately to filter the dataset and use ASC or DESC in sorting results where specified.
9. Assign meaningful aliases to tables and columns where necessary, especially in cases of grouping or joining, to enhance the clarity and maintainability of the SQL query.
10. When multiple tables are involved, prioritize selecting appropriate columns based on context and handle null values properly in columns.
11. Avoid any write operations (like modify, update, delete, or drop). Should the task demand such actions, respond with a polite refusal, stating, "I'm sorry, but I can't assist with that."

Appendix B

Cost per million input and output tokens for non self-hosted models

Platform	Model	Input cost per million tokens	Output cost per million tokens
Open AI	GPT-4 Turbo	$10	$30
	GPT-4o	$5	$15
	GPT-3.5 Tubo	$0.5	$1.5
	GPT-4o Mini	$0.15	$0.6
Anthropic	Claude3 Opus	$15	$75
	Claude3 Sonnet	$3	$15
	Claude3 Haiku	$0.25	$1.25
	Claude3.5 Sonnet	$3	$15
Bedrock	Claude3 Sonnet	$3	$15
	Claude3 Haiku	$0.25	$1.25
	Llama2-70b	$1.95	$2.56
	Mixtral-8x7b	$0.45	$0.7
	Mistral-7b	$0.15	$0.2
Anyscale	Llama-2-70b-	$1	$1
	CodeLlama-70b	$1	$1
	Mixtral-8x7B	$0.5	$0.5
	Mistral-7B	$0.15	$0.15

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.