

Comparison of AI Agents for Creating SQL Queries

Julia Sierpień*, Maria Skublewska-Paszkowska

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

Large language models have been increasingly applied for Text-to-SQL tasks recently, due to the fact that generating correct SQL queries is the most important factor. This study presents a comparison of AI agents based on open-source and closed-source large language models for generating SQL queries. GPT-4o, Claude 3.7 Sonnet, and LLaMA 3 8B were evaluated using two widely known datasets: Spider and WikiSQL. The chosen architectures were compared using Exact Match, F1-score, BERTScore, Execution Accuracy and processing time. Obtained results show that all agents perform well on simple queries, while agents based on closed-source models achieve better performance on complex SQL generation tasks.

Keywords: LLMs; Text-to-SQL; AI agents

*Corresponding author

Email address: s97731@pollub.edu.pl (J. Sierpień)

Published under Creative Common License (CC BY 4.0 Int.)

1. Introduction

In recent years, large language models (LLMs) have demonstrated an ability to understand natural language and generate complex queries. As a result, they became a promising tool for SQL query generation, especially helpful for users without extensive SQL knowledge. This approach can significantly simplify and improve database interactions.

Despite the growing number of LLMs, research comparing their performance in practical database tasks remains limited. These models can be divided into two main subsets: open-source and closed-source. The differences between these two groups may affect the quality, correctness, efficiency, and time required for SQL query generation.

The aim of this paper is to evaluate and compare AI agents built on different large language models (LLMs) in generating SQL queries. This study includes closed-source models such as GPT-4o and Claude 3.7 Sonnet, and an open-source model LLaMA 3 8B. The following hypotheses were defined in the study:

H1: The agent based on the open-source LLaMA 3 8B LLM requires more time to generate SQL queries than the agents based on GPT-4o and Claude 3.7 Sonnet.

H2: Agents based on closed-source LLMs achieve better performance in generating complex SQL queries than the agent based on open-source LLM, as measured by Exact Match, BERTScore, F1-score, and Execution Accuracy.

H3: The agent based on open-source LLaMA 3 8B LLM achieves performance comparable to agents based on closed-source LLMs in generating simple SQL queries, as measured by Exact Match, BERTScore, F1-score, and Execution Accuracy.

2. Related work

This chapter summarizes recent studies on AI-based SQL query generation and Text-to-SQL methods. The

related literature is divided into four sections, which are presented in the following part of the chapter.

2.1. Surveys and Overviews

The evolution of Text-to-SQL systems is dominated by Large Language Models (LLMs). Research identifies four key foundations for Text-to-SQL: natural language understanding, schema linking, semantic parsing, and SQL generation [1]. Surveys outline several recurring challenges for Text-to-SQL systems, such as: generalizing across domains, difficulties with schema understanding, strict syntax and vocabulary gaps [2][3]. The most popular Text-to-SQL datasets, according to recent research, are Spider (a complex cross-domain dataset) and WikiSQL [4]. The performance of systems is primarily evaluated using execution accuracy and exact matching [5].

2.2. LLMs for SQL generation

Comparative studies evaluate the capabilities of various LLMs, often using the Spider and Spider-realistic datasets [6]. Closed-source models: GPT-3.5 and GPT-4 consistently outperform open-source models, such as Dolly, LLaMA, and MistralAI in Text-to-SQL tasks [7][8]. Evaluation of SQL syntax accuracy on the NorthWind database reported an average accuracy of 87% for GPT-3.5 and 80% for Gemini 1.0 Pro [9]. While LLMs handle recognition tasks effectively, they still struggle with understanding semantics [4]. Despite high accuracy, LLMs lack the expertise needed to optimize the queries, meaning programmers cannot be fully replaced.

2.3. Prompt Engineering, RAG and In-Context Learning

Retrieval-Augmented Generation (RAG) and specific prompt design strategies are implied to enhance LLMs' accuracy and to improve their performance [10]. RAG systems consist of a retrieval module and a generative

module. Although RAG is a promising answer to the limitations of the existing Text-to-SQL systems, it has issues with dynamic adaptation to schema changes. ReFSQL is a retrieval-augmented framework that improves robustness by using a structured-enhanced retriever and a generator [11].

ASTRES combines AST-based reranking and schema pruning to dynamically gather database-specific information and determine the best few-shot examples for In-Context Learning [12]. Strategies like SL (identifying the relevant tables), CC (clause-by-clause SQL generation), and SL+CC (combination of both) significantly improved the performance of LLMs [13]. To enhance the accuracy of RW-RDBs, a two-step RAG-based method was proposed. First, generating a synthetic dataset of natural language and SQL pairs, then retrieving similar pairs to use as prompts to the LLM [14]. Although EMSL reduces robustness, it creates models vulnerable to typos. The study [15] suggests substituting it with a pre-trained language model as an encoder.

2.4. Synthetic Data, Fine-Tuning and Training Methods

A synthetic data method that blends high-quality samples from large, advanced models with weaker samples generated by smaller models was proposed to improve the generalization in Text-to-SQL models [16]. Research on training LLMs for schema-independent prompting showed that fine-tuned Flan-T5 models achieved an LFA of 0.85, outperforming current state-of-the-art models [17]. LR-SQL is a general dual-model SFT framework that includes both a schema linking model and an SQL generation model. It was proven to use 40% less GPU memory than baseline SFT methods [18]. The study [19] introduces a dynamic loss function that applies distinct weighting to specific token categories, improving the execution accuracy by 20% compared to the standard loss function.

3. Materials and methods

This section describes the datasets, models and evaluation metrics used during the experiments.

3.1. Datasets

The experiments were conducted using two popular datasets for Text-to-SQL tasks: Spider [20] and WikiSQL [21].

Spider is a large-scale, complex, and cross-domain dataset. It consists of natural language questions paired with SQL queries across multiple databases. The queries range from simple SELECT statements to advanced queries involving JOIN operations, aggregations, and nested subqueries. Spider categorizes queries into four difficulty levels: easy, medium, hard, and extra hard. This dataset is widely used to evaluate models' ability to generate semantically correct and structurally accurate queries. In this study, 866 queries were randomly selected from the Spider dataset. The sampled data reflects the original distribution of query difficulty levels.

WikiSQL is a large-scale dataset focused only on single-table queries. Similar to Spider, it provides natural language questions mapped to SQL queries. The SQL statements are limited to simple SELECT statements with optional aggregation and filtering conditions. In this study, 400 queries were randomly selected to evaluate model performance on simple SQL generation tasks.

Using both datasets enables a thorough evaluation of large language models on both simple and complex SQL queries, providing a comparative analysis across different levels of query difficulty. SQL queries were normalized by removing redundant whitespace and standardizing keyword casing to ensure consistent comparison. Natural language questions were kept in their original form. No data augmentation of queries was performed. The final distribution of queries across datasets and difficulty levels is presented in Table 1.

Table 1: Distribution of selected queries across datasets and difficulty levels.

Dataset	Easy	Medium	Hard	Extra Hard	Total
Spider	456	300	100	10	866
WikiSQL	400	-	-	-	400

3.2. Models

Based on the literature overview, the performance of open-source and closed-source LLMs differs significantly. This study evaluates three AI agents based on large language models for SQL query generation: two closed-source models, GPT-4o and Claude 3.7 Sonnet, as well as one open-source model, LLaMA 3 8B. The selected LLMs were used as the foundation for AI agents in order to examine how differences in model architecture influence SQL query generation performance.

3.2.1. GPT-4o

GPT-4o [15] is a large multimodal model developed by OpenAI. It is based on a Transformer architecture and pre-trained using both commonly available and licensed data. GPT-4o was fine-tuned using Reinforcement Learning from Human Feedback (RLHF). GPT-4o outperforms existing large language models on across various NLP (Natural Language Processing) tasks.

3.2.2. Claude 3.7 Sonnet

Claude 3.7 Sonnet [22] is a closed-source large language model developed by Anthropic. It is built on a Transformer-based architecture, designed to support advanced contextual understanding. Claude represents a state-of-the-art alternative to GPT-4o.

3.2.3. LLaMA 3 8B

LLaMA 3 8B [23] is an open-source large language model developed by Meta. It is based on a Transformer architecture. This model is pre-trained on 15.6T tokens and contains 8 billion parameters, supporting a context window of up to 8K tokens.

3.3. Metrics

AI agents' performance was evaluated using multiple metrics to ensure both structural accuracy and semantic validity of generated SQL queries.

For evaluating queries, Exact Match, F1-score, Execution Accuracy, and BERTScore were applied. Exact Match [24] measures whether the generated SQL query exactly matches the reference query after normalization. F1-score [25] evaluates the correctness of individual SQL clauses by comparing predicted and reference components. The F1-Score is defined as:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (1)$$

where:

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

and TP is True Positive, FP is False Positive and FN is False Negative. In addition, BERTScore [26] was used to measure the contextual similarity between generated and reference SQL queries. BERTScore is defined as:

$$F1_{BERT} = 2 \cdot \frac{P_{BERT} \cdot R_{BERT}}{P_{BERT} + R_{BERT}} \quad (4)$$

where:

$$P_{BERT} = \frac{1}{|x|} \sum_{x_i \in x} \max_{y_j \in y} x_i^T y_j \quad (5)$$

$$R_{BERT} = \frac{1}{|y|} \sum_{y_j \in y} \max_{x_i \in x} x_i^T y_j \quad (6)$$

and x, y are the candidate and reference texts, x_i, y_j are the token embeddings. Execution Accuracy [24] tests if the generated SQL query is correct and produces the same outcome as the referenced query. Execution Accuracy is defined as:

$$Execution\ Accuracy = \frac{\text{Number of correct executions}}{\text{Total number of queries}} \quad (7)$$

3.4. Experimental setup

All experiments were conducted using pre-trained large language models accessed via their APIs. No additional fine-tuning was performed. For each query, the corresponding AI agent generated a single SQL query which was then evaluated using Exact Match, F1 score, BERTScore and Execution Accuracy. The evaluation was performed on a local machine with the following specifications: Apple MacBook Air with M1 chip, running macOS Thae 26.2.

4. Results

AI agents' evaluation was divided into three parts: simple queries from the WikiSQL dataset, simple queries from the Spider dataset (easy subset) and complex queries from the Spider dataset (medium, hard and extra hard subsets).

4.1. Simple queries from WikiSQL

Evaluation of agents was conducted on 400 queries from the WikiSQL dataset. The performance of the evaluated systems is presented in Table 2. All evaluated agents achieved perfect Execution Accuracy, indicating that all generated queries returned correct results. Differences between the evaluated systems can be observed in Exact Match (EM) and F1-Score. The highest EM was achieved by the GPT-4-based agent (0.09), followed by LLaMA-based agent (0.06), while Claude-based agent obtained the lowest value (0.01). Similar results were observed for F1-Score where GPT-4-based agent and LLaMA-based agent achieved comparable results (0.71), outperforming Claude-based agent (0.66). BERTScore values were consistently high for all evaluated systems (0.93-0.96), suggesting high semantic similarity between generated and reference queries. The shortest processing time was achieved by GPT-4-based agent, followed by Claude-based agent, while LLaMA-based agent was the slowest. These results indicate that all agents are highly effective at generating correct SQL queries for simple, single-table tasks.

Table 2: Performance of evaluated agents on simple queries from the WikiSQL dataset.

Model	Time	F1-score	Bert-Score	Exact Match	Execution Accuracy
LLaMA	13min 42s	0.71 ±0.16	0.96 ±0.03	0.06 ±0.23	1.00 ±0.00
GPT	6min 13s	0.71 ±0.17	0.95 ±0.02	0.09 ±0.28	1.00 ±0.00
Claude	10min 36s	0.66 ±0.17	0.93 ±0.03	0.01 ±0.07	1.00 ±0.00

4.2. Simple queries from Spider

The results for simple queries from the Spider dataset (456 queries) are presented in Table 3. In comparison to WikiSQL, the performance of the evaluated agents on Spider is slightly lower, reflecting the greater variability of queries. GPT-4-based agent achieved the highest Execution Accuracy (0.87), followed by Claude-based agent (0.84) and LLaMA-based agent (0.83). Exact

Match remained low for all evaluated systems, with GPT-4-based agent achieving the highest score (0.21) and Claude-based agent the lowest (0.15). The highest F1-Score was obtained by LLaMA-based agent (0.77), slightly outperforming GPT-4-based agent (0.75) and Claude-based agent (0.72). BERTScore values achieved by GPT-4-based agent and LLaMA-based agent were nearly identical (0.94), indicating comparable semantic

quality of the generated queries. The shortest processing time was again achieved by GPT-4-based agent, while LLaMA-based agent was the slowest. These results indicate that although all evaluated systems performed

well on simple queries, GPT-4-based agent provides slightly better execution correctness, while LLaMA-based agent achieves competitive performance in terms of token-level similarity.

Table 3: Performance of evaluated agents on simple queries from the Spider dataset.

Model	Time	F1-Score	Bert-Score	Exact Match	Execution Accuracy
LLaMA	16min 24s	0.77 $\pm$ 0.20	0.94 $\pm$ 0.04	0.19 $\pm$ 0.39	0.83 $\pm$ 0.38
GPT	6min 21s	0.75 $\pm$ 0.22	0.94 $\pm$ 0.04	0.21 $\pm$ 0.41	0.87 $\pm$ 0.34
Claude	11min 53s	0.72 $\pm$ 0.23	0.92 $\pm$ 0.04	0.15 $\pm$ 0.36	0.84 $\pm$ 0.37

4.3. Complex queries from Spider

The performance of the evaluated agents on complex queries from the Spider dataset (medium, hard and very hard queries) is presented in Table 4. The results show a significant decrease in performance across all agents compared to simple queries. The highest Execution Accuracy was achieved by GPT-4-based agent (0.60), followed by Claude-based agent (0.57) and LLaMA-based agent (0.51). Exact Match scores were extremely low for all evaluated systems, with GPT-4-based agent achieving 0.03, LLaMA-based agent reaching 0.02, and Claude-based agent failing to generate any exact match-

es (0.00). The highest F1-score was achieved by LLaMA-based agent (0.58), followed by GPT-4-based agent (0.52) and Claude-based agent (0.50). BERTScore values were significantly lower than in simple tasks (0.88-0.90), although still relatively high. These results indicate that the evaluated agents preserved semantic meaning despite structural inaccuracies. This suggests that query complexity significantly impacts agents' performance. Evaluated systems struggled with generating structurally correct SQL, especially in cases involving multiple joins, aggregations, and more complex query structures.

Table 4: Performance of evaluated agents on complex queries from the Spider dataset.

Model	Time	F1-score	Bert-Score	Exact Match	Execution Accuracy
LLaMA	14min 13s	0.58 $\pm$ 0.19	0.90 $\pm$ 0.04	0.02 $\pm$ 0.15	0.51 $\pm$ 0.50
GPT	7min 36s	0.52 $\pm$ 0.15	0.90 $\pm$ 0.03	0.03 $\pm$ 0.16	0.60 $\pm$ 0.49
Claude	10min 36s	0.50 $\pm$ 0.14	0.88 $\pm$ 0.02	0.00 $\pm$ 0.00	0.56 $\pm$ 0.50

5. Discussion

5.1. Performance on simple queries

The results for simple queries indicate that all evaluated agents obtained high performance, especially in terms of Execution Accuracy. For WikiSQL, all agents reached a perfect score, while for Spider, the values remained high, with GPT-4-based agent achieving the best score. Although the Execution Accuracy was similar for all evaluated agents, differences can be observed in Exact Match and F1-score. The highest Exact Match was obtained by GPT-4-based agent which indicates better structural correctness of generated queries. However, LLaMA-based agent achieved similar F1-score and BERTScore values, which suggests that the queries it generated were semantically similar.

These results confirm hypothesis H3, which states that the agent based on open-source model achieves performance comparable to agents based on closed-source models in generating simple queries. Although LLaMA-based agent achieved worse results than GPT-4-based agent, its comparable results in F1-score and BERTScore show that it remains competitive for less complex queries.

5.2. Performance on complex queries

A significant decrease in performance was observed for complex queries. All agents obtained lower Execution

Accuracy, Exact Match, F1-score and BERTScore in comparison to simple queries. Exact match values were extremely low for all evaluated agents, which indicates difficulties in generating fully correct SQL queries. The best Execution Accuracy was obtained by GPT-4-based agent, followed by Claude-based agent and LLaMA-based agent.

These results confirm hypothesis H2, which assumes that agents based on closed-source models outperform the agent based on open-source models in generating complex SQL queries.

5.3. Processing time analysis

The analysis of processing time shows that GPT-4-based agent was consistently the fastest, followed by Claude-based agent, while LLaMA-based agent required the longest time to generate queries.

These results confirm hypothesis H1, which assumes that the agent based on open-source model would require the longest time to generate queries than agents based on closed-source models.

The observed differences in processing time may be related to variations in model size and infrastructure efficiency between model providers.

5.4. Comparison with existing literature

These obtained results in this study are consistent with existing research on Text-to-SQL systems. The litera-

ture review shows that closed-source models outperform open-source models in complex queries, which was confirmed by the results of this study [7][8]. The relatively strong performance of the LLaMA-based agent in simple queries supports previous studies suggesting that open-source models are becoming more competitive [23]. Recent findings highlight the difficulty of generating syntactically perfect SQL queries, which can be observed in the low Exact Match scores across all datasets [2][4]. The results indicate the need to improve structural correctness through approaches such as prompt engineering or schema-aware techniques [10][11][12].

6. Conclusions

This study evaluated the performance of agents based on open-source and closed-source LLMs in Text-to-SQL tasks on two datasets with different complexity. The results show that all evaluated agents perform well on simple queries, obtaining high Execution Accuracy as well as strong semantic similarity. In this context, the agent based on the open-source LLaMA 3 model achieved comparable results to agents based on closed-source models as assumed in hypothesis H3. However, the performance of the agent based on the open-source LLaMA 3 model decreases significantly on complex queries. The best results were obtained by closed-source models, particularly GPT-4-based agent. The agent based on open-source model remains competitive in terms of partial correctness, which can be observed in F1-score and BERTScore. The GPT-4-based agent proved to be the fastest, while LLaMA-based agent was the slowest. These findings suggest that open-source models are becoming a possible alternative for simple tasks. However, closed-source models still provide superior performance in more complex queries. Future works should focus on improving structural accuracy through advanced techniques such as prompt optimization or evaluating performance in real-world database environments.

References

- [1] A. Singh, A. Shetty, A. Ehtesham, S. Kumar, T. T. Khoei, A survey of large language model-based generative AI for text-to-SQL: Benchmarks, applications, use cases, and challenges, In 2025 IEEE 15th Annual Computing and Communication Workshop and Conference (CCWC) (2025) 15–21, <https://doi.org/10.1109/CCWC62904.2025.10903689>
- [2] A. Mohammadjafari, A. S. Maida, R. Gottumukkala, From natural language to SQL: Review of LLM-based text-to-SQL systems, arXiv preprint arXiv:2410.01066 (2025), <https://doi.org/10.48550/arXiv.2410.01066>.
- [3] G. Katsogiannis-Meimarakis, G. Koutrika, A survey on deep learning approaches for text-to-SQL, The VLDB Journal 32(4) (2023) 905–936, <https://doi.org/10.1007/s00778-022-00776-8>.
- [4] A. B. Kanburoğlu, F. B. Tek, Text-to-SQL: A methodical review of challenges and models, Turkish Journal of Electrical Engineering and Computer Sciences 32(3) (2024) 403–419, <https://doi.org/10.55730/1300-0632.4077>.
- [5] Z. Hong, Z. Yuan, Q. Zhang, H. Chen, J. Dong, F. Huang, X. Huang, Next-generation database interfaces: A survey of LLM-based text-to-SQL, arXiv preprint arXiv:2406.08426 (2024), <https://doi.org/10.48550/arXiv.2406.08426>.
- [6] D. Gao, H. Wang, Y. Li, X. Sun, Y. Qian, B. Ding, J. Zhou, Text-to-SQL empowered by large language models: A benchmark evaluation, arXiv preprint arXiv:2308.15363 (2023), <https://doi.org/10.48550/arXiv.2308.15363>.
- [7] S. Sun, Y. Zhang, J. Yan, Y. Gao, D. Ong, B. Chen, J. Su, Battle of the large language models: Dolly vs LLaMA vs Vicuna vs Guanaco vs Bard vs ChatGPT – A text-to-SQL parsing comparison, In Findings of the Association for Computational Linguistics: EMNLP 2023 (2023) 11225–11238, <https://doi.org/10.18653/v1/2023.findings-emnlp.750>
- [8] R. Roberson, G. Kaki, A. Trivedi, Analyzing the effectiveness of large language models on text-to-SQL synthesis, arXiv preprint arXiv:2401.12379 (2024), <https://doi.org/10.48550/arXiv.2401.12379>.
- [9] C.-M. Rosca, A. Stancu, Quality assessment of GPT-3.5 and Gemini 1.0 Pro for SQL syntax, Computer Standards & Interfaces 95 (2026) 104041, <https://doi.org/10.1016/j.csi.2025.104041>.
- [10] L. Nan, Y. Zhao, W. Zou, N. Ri, J. Tae, E. Zhang, A. Cohan, D. Radev, Enhancing text-to-SQL capabilities of large language models: A study on prompt design strategies, In Findings of the Association for Computational Linguistics: EMNLP 2023 (2023) 14935–14956, <https://doi.org/10.18653/v1/2023.findings-emnlp.996>.
- [11] K. Zhang, X. Lin, Y. Wang, X. Zhang, F. Sun, C. Jianhe, H. Tan, X. Jiang, H. Shen, ReFSQL: A retrieval-augmentation framework for text-to-SQL generation, In Findings of the Association for Computational Linguistics: EMNLP 2023 (2023) 664–673, <https://doi.org/10.18653/v1/2023.findings-emnlp.48>.
- [12] Z. Shen, P. Vougiouklis, C. Diao, K. Vyas, Y. Ji, J. Z. Pan, Improving retrieval-augmented text-to-SQL with AST-based ranking and schema pruning, arXiv preprint arXiv:2407.03227 (2024), <https://doi.org/10.48550/arXiv.2407.03227>.
- [13] Z. Tan, X. Liu, Q. Shu, X. Li, C. Wan, D. Liu, Q. Wan, G. Liao, Enhancing text-to-SQL capabilities of large language models through tailored promptings, Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024) (2024) 6091–6109, <https://aclanthology.org/2024.lrec-main.539/>.
- [14] G. M. C. Coelho, E. R. S. Nascimento, Y. T. Izquierdo, G. M. García, L. Feijó, M. Lemos, R. L. S. Garcia, A. R. de Oliveira, J. P. Pinheiro, M. A. Casanova, Improving the accuracy of text-to-SQL tools based on large language models for real-world relational databases, Database and Expert Systems Applications: 35th International Conference, DEXA 2024 (2024) 93–107, https://doi.org/10.1007/978-3-031-68309-1_8.

- [15] OpenAI, J. Achiam, S. Adler, S. Agarwal et al., GPT-4 technical report, arXiv preprint arXiv:2303.08774 (2024), <https://doi.org/10.48550/arXiv.2303.08774>.
- [16] J. Yang, B. Hui, M. Yang, J. Yang, J. Lin, C. Zhou, Synthesizing text-to-SQL data from weak and strong LLMs, arXiv preprint arXiv:2408.03256 (2024), <https://doi.org/10.48550/arXiv.2408.03256>.
- [17] Y. Mellah, V. Kocaman, H. U. Haq, D. Talby, Efficient schema-less text-to-SQL conversion using large language models, *Artificial Intelligence in Health* 1(2) (2024) 96–106, <https://doi.org/10.36922/aih.2661>.
- [18] W. Wen, Y. Zhang, S. Pan, Y. Sun, P. Lu, C. Ding, LR-SQL: A supervised fine-tuning method for Text2SQL tasks under low-resource scenarios, *Electronics* 14(17) (2025) 3489, <https://doi.org/10.3390/electronics14173489>.
- [19] I. Azurmendi, E. Zulueta, G. García, N. Uriarte-Arrazola, J. M. Lopez-Guede, Beyond standard losses: Redefining text-to-SQL with task-specific optimization, *Mathematics* 13(14) (2025) 2315, <https://doi.org/10.3390/math13142315>.
- [20] Spider dataset, <https://yale-lily.github.io/spider>, [05.06.2026].
- [21] WikiSQL dataset, <https://github.com/salesforce/WikiSQL>, [05.06.2026].
- [22] E. Kasireddy, C. Chow, J. Collet, M.-M. Pourrahmat, M. S. Fazeli, Evaluating the performance of Claude 3.7 Sonnet in data extraction automation for systematic literature reviews, *Value in Health Regional Issues* 53 (2026) 101539, <https://doi.org/10.1016/j.vhri.2025.101539>.
- [23] A. Grattafiori et al., The Llama 3 herd of models, arXiv preprint arXiv:2407.21783 (2024), <https://doi.org/10.48550/arXiv.2407.21783>.
- [24] Exact Match, Execution Accuracy, <https://deepwiki.com/taoyds/spider/3.1-evaluation-metrics>, [05.06.2026].
- [25] F1 Score, <https://www.v7labs.com/blog/f1-score-guide>, [05.06.2026].
- [26] BERTScore, <https://www.analyticsvidhya.com/blog/2025/04/bertscore-a-contextual-metric-for-llm-evaluation/>, [05.06.2026].