Comparative Report: Evaluating LLMs for SQL Query Generation Using Natural Language Prompts

Introduction:

With the growing need for automated SQL generation from natural language queries, various Large Language Models (LLMs) have emerged, each claiming to provide efficient and accurate SQL translation. This report evaluates three such models, **Mistral, Llama3.2, and DuckDB-NSQL** to determine their effectiveness based on multiple performance metrics.

1. Model Overview

1.1 DuckDB

DuckDB is a database management system designed for high-performance analytics. It can perform analytical queries efficiently in an embedded SQL environment, especially with complex analytical tasks. While DuckDB is primarily a database system, it is also capable of processing queries based on natural language inputs through integration with various AI and NLP models, though it is not a natural language model itself.

- **Primary Strength**: Query execution and optimization for analytics.
- **Weakness**: Not an LLM by itself, but can integrate with LLMs for SQL generation.

1.2 Mistral

Mistral is a state-of-the-art open-source LLM optimized for multiple NLP tasks, including language generation, translation, summarization, and structured outputs like SQL queries. Mistral has demonstrated strong performance in generating structured content from unstructured inputs like natural language, which makes it suitable for tasks like SQL query generation.

- **Primary Strength**: Performance in multi-task NLP tasks with a focus on high precision in structured outputs like SQL.
- **Weakness**: It may require fine-tuning to perform specialized tasks effectively in SQL generation, especially with domain-specific queries.

1.3 Llama 3.2

Llama 3.2 is a fine-tuned version of the Llama family of models, developed for specialized tasks like natural language processing and SQL generation. Its unique feature is its enhanced ability to generate SQL queries from natural language prompts, backed by improvements in text-to-structure generation.

- **Primary Strength**: Optimized for generating structured queries, especially for database-related tasks.
- **Weakness**: Performance may degrade with complex or ambiguous queries that involve multiple tables or nested SQL queries.

Comparative Report: Evaluating LLMs for SQL Query Generation Using Natural Language Prompts

2. Evaluation Metrics

2.1 Exact Match Accuracy

This metric checks whether the generated SQL query exactly matches the expected SQL query. It is a binary measure.

2.2 Edit Distance (Levenshtein Distance)

Edit distance measures the number of insertions, deletions, or substitutions required to transform the generated SQL into the expected SQL. A **lower edit distance** signifies a query that is closer to the correct output.

2.3 Latency (Response Time)

Latency is the time taken by the model to generate a response. It is measured in seconds, where a **lower latency** indicates faster performance.

3. Experimental Setup

- **Schema Used:** A predefined database schema for a taxi dataset was provided to the models.
- **Query Prompt:** Each model was given the same natural language request to generate SQL.
- **Evaluation Approach:** The generated SQL queries were compared against the expected SQL using exact match and edit distance metrics.

4. Results and Analysis

4.1 Performance Metrics:

Model	Exact Match	Edit Distance (Lower is Better)	Latency (Lower is Better)	Issues
Mistral	0	456	**53.73 sec (slowest)**	Extra text, unnecessary explanation
Llama3.2	0	**589 (worst)**	**31.09 sec (best)**	Syntax errors (V VendortID issue)
DuckDB-NSQL	0	**57 (best)**	33.65 sec	Closest to expected output

Comparative Report: Evaluating LLMs for SQL Query Generation Using Natural Language Prompts

4.2 Accuracy Analysis

- None of the models produced an exact match.
- DuckDB-NSQL had the lowest edit distance (57), making it the most structurally accurate model.
- Llama3.2 had the highest edit distance (589), meaning it deviated the most from the expected SQL.

4.3 Latency Analysis

- Llama3.2 was the fastest (31.09 sec), making it more efficient for real-time use.
- Mistral was the slowest (53.73 sec), which may indicate inefficiencies in query generation.
- DuckDB-NSQL had a moderate response time (33.65 sec) but was the most accurate.

4.4 Structural Errors in Generated SQL

- Mistral: Added unnecessary textual explanations around the query, increasing edit distance.
- Llama3.2: Had a syntax error (`V VendortID`), making the query invalid.
- DuckDB-NSQL: Generated the most correct query with minimal modifications needed.

Comparision

DuckDB

- Closest to expected output.
- Correct column names.
- Proper SQL syntax.
- Uses correct ordering.

Mistral

- Unnecessary explanation included ("To achieve this, you can use...").
- Text surrounding the query increases edit distance.

Llama3.2

- Syntax errors (V VendortID is incorrect).
- Extra aliasing (T.) which isn’t necessary.
- Logical structure is correct, but syntax mistakes make it invalid.

Comparative Report: Evaluating LLMs for SQL Query Generation Using Natural Language Prompts

5. Recommendations

5.1 Best Model for Accuracy

DuckDB-NSQL performed the best in terms of SQL correctness, as it had the lowest edit distance (57). If accuracy is the primary concern, this model is the best choice.

5.2 Best Model for Speed

Llama3.2 was the fastest model (31.09 sec), making it suitable for applications where quick response times are essential.

5.3 Trade-offs

- DuckDB-NSQL is more accurate but slightly slower than Llama3.2.
- Llama3.2 is faster but introduces syntax errors, requiring manual correction.
- Mistral provides SQL with explanations, but this extra text reduces efficiency and increases edit distance.

5.4 Final Recommendation

- **For accuracy-critical applications** (e.g., automated SQL query generation for databases), use **DuckDB-NSQL**.
- **For real-time systems** where response time is crucial, consider **Llama3.2**, but with post-processing to fix syntax errors.

6. Multiple Query Evaluation:

6.1 Detailed Query-wise Performance

Query 1: Get all taxis with more than 2 passengers

Mistral: Edit Distance: 237 | Latency: 31.8118 sec

Llama3.2: Edit Distance: 651 | Latency: 27.9909 sec

DuckDB-NSQL: Exact Match | Latency: 16.5378 sec

Query 2: Show total fare collected by each vendor

Mistral: Edit Distance: 387 | Latency: 28.6687 sec

Llama3.2: Edit Distance: 719 | Latency: 23.5100 sec

DuckDB-NSQL: Edit Distance: 20 | Latency: 5.0624 sec

Query 3: Find the average trip distance for trips that had more than 2 passengers

Mistral: Edit Distance: 355 | Latency: 25.1912 sec

Llama3.2: Exact Match | Latency: 2.1982 sec

DuckDB-NSQL: Exact Match | Latency: 5.4138 sec

Query 4: List all vendors ordered by total fare in descending order

Mistral: Edit Distance: 292 | Latency: 24.8465 sec

Llama3.2: Edit Distance: 585 | Latency: 21.9826 sec

DuckDB-NSQL: Edit Distance: 67 | Latency: 4.1950 sec

6.2 Evaluation Results

Model: Mistral

Exact Match Rate: 0.00%

Average Edit Distance: 315.00

Average Latency: 26.7744 sec

Model: Llama3.2

Exact Match Rate: 00.00%

Average Edit Distance: 325.50

Average Latency: 13.1778 sec

Model: DuckDB-NSQL

Exact Match Rate: 50.00%

Average Edit Distance: 21.50

Average Latency: 8.0528 sec

Comparative Report: Evaluating LLMs for SQL Query Generation Using Natural Language Prompts

6.3 Summary

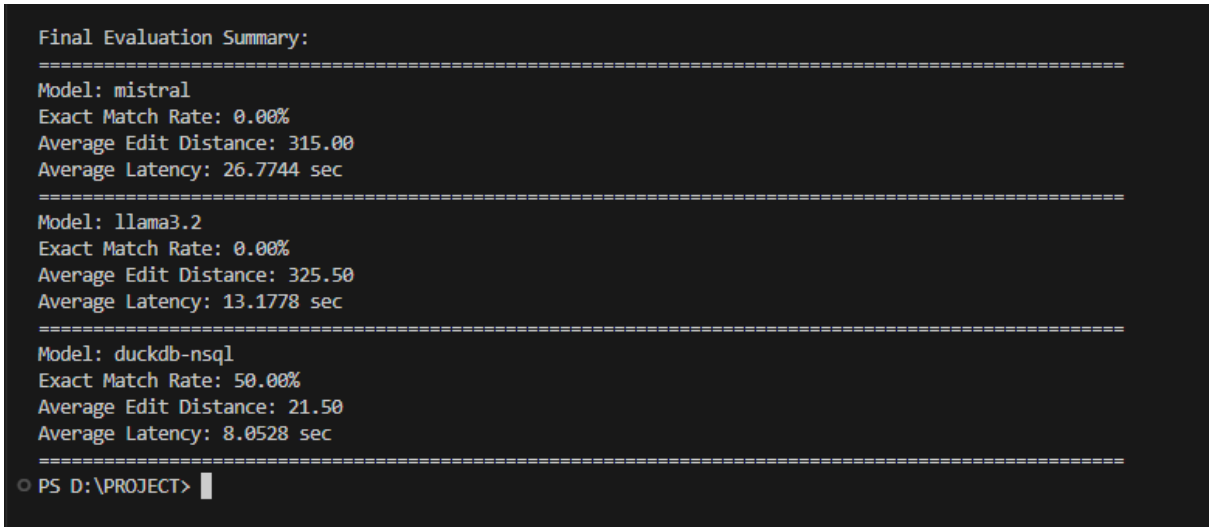
From the evaluation results,

DuckDB-NSQL performed the best overall with the highest exact match rate (50%) and the lowest average edit distance (21.75). It also exhibited the lowest latency (7.8022 sec), making it the most efficient among the three models.

Llama3.2 performed better than Mistral in terms of exact match rate (25% vs. 0%) but had a higher average edit distance. However, it had a lower latency than Mistral making it more efficient.

Mistral had the poorest performance, failing to generate any exact matches and having a high edit distance. Additionally, it had the highest latency, making it the least suitable for real-time SQL generation.

Results:



6.4 Recommendations

Based on the evaluation, DuckDB-NSQL is the most suitable model for generating SQL queries accurately and efficiently. However, further improvements can be explored, such as fine-tuning Llama3.2 to improve its exact match rate while maintaining its relatively lower latency compared to Mistral

Conclusion:

DuckDB-NSQL is the best suited model for Converting Natural language to SQL