Using DeepSeek to Query TMDb Movie Data in MongoDB

1. Team Details

Group Name: ChatDB 81

Team Size: 1 member

Member Name: Jinyi Wang

The project is being undertaken individually by Jinyi Wang under the team name ChatDB 81.

2. Tech Stack Used

The project utilizes Python for backend processing and query handling, with MongoDB as the primary database. PyMongo is used for database interactions, while DeepSeek API processes natural language queries. TMDb API provides movie data, which is stored in MongoDB. Additional tools include Flask (optional for API integration), Postman (for testing API requests), and Jupyter Notebook/PyCharm for development. A local caching mechanism optimizes API calls, and query sanitization techniques prevent injection attacks.

3. Query Syntax Implementation Plan

The system processes user queries using DeepSeek API, which extracts key parameters such as movie title, genre, release year, director, and rating. These parameters are then structured into a MongoDB query for accurate retrieval.

For example, if a user inputs “Top-rated sci-fi movies after 2020”, the system:

1.Extracts genre = sci-fi, release_year > 2020, rating sorting.

2.Constructs a MongoDB query:

{ "genre": "Science Fiction", "release_year": { "$gt": 2020 } }

3.Queries MongoDB and returns a ranked list of results.

To enhance accuracy, the system applies semantic parsing, query optimization, and synonym mapping, ensuring variations like "best sci-fi films" and "top-rated science fiction movies" yield the same results.

4. Database Selection

The project uses MongoDB due to its flexibility in handling semi-structured movie data retrieved from TMDb API. The document-based structure allows for efficient storage of metadata such as genres, cast details, and ratings without requiring complex table relationships. MongoDB’s aggregation framework supports advanced filtering and ranking, aligning with the DeepSeek API’s dynamic query generation. Cloud-based MongoDB Atlas is considered for scalability and remote access.

5. Planned Implementation

The project follows a structured approach, beginning with database setup and data ingestion from the TMDb API into MongoDB, followed by the integration of DeepSeek API for natural language query processing. The initial phase focused on

retrieving movie data, structuring it into collections such as movies, actors, and reviews, and optimizing query performance through indexing. The next phase involved developing a command-line interface (CLI) where users can input natural language queries, which are processed by DeepSeek and converted into MongoDB find() and aggregation queries.

A key deviation from the original proposal is the extended timeline for complex query execution. While basic queries such as retrieving movies by genre and release year were implemented on schedule, handling multi-collection queries using $lookup and refining DeepSeek's query parsing accuracy required additional time. To address API rate limits and improve efficiency, local caching mechanisms were introduced to store frequently queried results. Moving forward, the focus is on enhancing data modification capabilities (insert, update, delete) and refining query translations for ambiguous user inputs.

6. Project Status

So far, the project has progressed well, with database setup, schema exploration, and initial query execution successfully implemented. The MongoDB database has been populated with movie data from TMDb API, and the system can process basic natural language queries using DeepSeek API. Users can search for movies based on criteria such as genre, release year, and ratings, with queries being accurately converted into MongoDB operations.

However, some challenges have led to slight deviations from the original plan. Handling multi-collection joins ($lookup) and aggregation queries ($match, $group, $sort) required additional refinement, and a local caching system was introduced to optimize API calls and reduce redundant queries. The next steps include completing data modification features (insert, update, delete) and focusing on query optimization, testing, and debugging to ensure accurate and efficient execution. Final refinements will improve query translation for ambiguous user inputs and enhance overall system performance before the final demonstration.

7. Challenges Faced

One of the main challenges encountered was handling complex query translation from natural language to MongoDB syntax. While basic queries such as filtering by genre or release year worked as expected, queries requiring aggregation and multi-collection joins ($lookup) were more difficult to process accurately. To address this, additional parsing logic was implemented to refine how DeepSeek API interprets user input, ensuring correct mapping to MongoDB query structures.

Another challenge was API rate limits and performance optimization. Frequent calls to DeepSeek API for query translation introduced delays and increased dependency on external processing. To mitigate this, a local caching system was implemented to store frequently queried results, reducing redundant API requests and

improving response times. Additionally, ensuring consistent query interpretation for varying user input required refining synonym mapping and entity recognition, allowing the system to handle similar queries with different phrasing more effectively. Testing and debugging efforts continue to further improve accuracy and efficiency.

8. Timeline

Week	Dates	Tasks
Week 4	Mar 2 - Mar 8	Improve NLP model accuracy for handling complex queries
Week 5	Mar 9 - Mar 15	Implement filtering and ranking features (e.g., sorting by rating, genre)
Week 6	Mar 16 - Mar 22	Conduct initial testing and debugging
Week 7	Mar 23 - Mar 29	Optimize system performance and security
Week 8	Mar 30 - Apr 5	Final refinements and optimizations
Week 9	Apr 6 - Apr 12	Prepare for final project demonstration
Week 10	Apr 13 - Apr 19	Final project demonstration preparation
Week 11	Apr 20 - Apr 26	Buffer time for unexpected issues
Week 12	Apr 27 - May 3	Final documentation and report writing
Week 13	May 4 - May 9	Submit final project report