KHALIL SHAIKH

Vishwakarma University

LangChain AI/ML Assignment:

1. Abstract

This project aims to transform unstructured narrative data into structured JSON format by extracting key details about characters from stories. Using LangChain and MistralAI, we developed a system that leverages embeddings for information retrieval and processing. The solution involves creating a vector database to store story embeddings and a command-line interface (CLI) for querying character information. This report outlines the objectives, methodology, implementation, and results of the project.

2. Introduction

Objective:

The primary goal is to extract structured character details from a dataset of stories, outputting these details in JSON format. The process includes computing embeddings for the stories and using them to retrieve relevant information accurately.

Scope:

This project demonstrates how to process unstructured narrative data into a structured format using tools like LangChain and open-source vector databases. It also emphasizes edge case handling and a seamless user experience via CLI commands.

3. Requirements and Setup

Tools and Technologies:

Programming Language: Python

Framework: LangChain

AI Model: MistralAI

Vector Database: ChromaDB (or any free/open-source alternative)

Dataset:

The dataset comprises multiple files, each representing a story. These files are processed to compute embeddings and extract character-specific details.

4. Methodology

Embedding Computation:

The compute-embeddings command processes all story files, computes embeddings using MistralAI, and stores them in a vector database. This enables efficient search and retrieval of story details.

Character Information Retrieval:

The get-character-info command takes a character name as input, searches the vector database for relevant embeddings, and outputs the following JSON structure:

{name: "<Character Name>",storyTitle: "<Story Title>",summary: "<Character Summary>",relations: [{ "name": "<Relation Name>", "relation": "<Relation Type>" }],characterType: "<Character Role>"}.

5. Implementation

Command 1: compute-embeddings

Input: All story files in the dataset.

Process: Parse the files, compute embeddings using MistralAI, and store them in the vector database.

Output: Persisted embeddings for future queries.

Command 2: get-character-info

Input: Character name.

Process: Query the vector database to find relevant embeddings and extract structured details.

Output: JSON object containing character details.

Edge Case Handling:

If the character name is not found, return a JSON response with a "Character not found" message.

Handle malformed or missing input gracefully.

6. Results

Example JSON Output:

{name: "Jon Snow",storyTitle: "A Song of Ice and Fire",summary: "Jon Snow is a brave and honorable leader who serves as the Lord Commander of the Night's Watch and later unites the Free Folk and Westeros against the threat of the White Walkers.",relations: [{ "name": "Arya Stark", "relation": "Sister" },{ "name": "Eddard Stark", "relation": "Father" }],characterType: "Protagonist"}

Performance:

The embeddings were computed efficiently, and queries returned results within seconds.

Edge cases were tested, and the system handled invalid inputs as expected.

7. Conclusion

The project successfully demonstrated the use of LangChain and MistralAI for processing unstructured narrative data. By leveraging embeddings and vector databases, we achieved accurate and efficient character detail extraction. Future improvements could include:

Supporting additional data formats.

Enhancing the summarization capability with more advanced models.

Expanding the JSON schema for richer detail representation.

8. Appendix

GitHub Repository:

The repository contains:

Source code for embedding computation and character retrieval.

A detailed README with setup instructions and a demo walkthrough.

References:

LangChain Documentation

MistralAI

Vector Database Options