Intent Identification task report (Chetas Parekh)

1. Introduction

Community college students in California frequently encounter challenges when navigating the transfer process to University of California (UC) or California State University (CSU) campuses. Common inquiries pertain to course equivalencies, the development of semester-by-semester academic plans, and various other transfer-related questions. To enhance and automate the provision of answers to these queries, a specialized chatbot system has been developed. This report details the problem-solving methodology employed, describes the system's overall technical framework, elucidates the selection and application of various tools and systems, and offers reflections on the role of Generative AI in facilitating the project's development.

2. Problem-Solving Process

a. Definition and Scope

The primary objective was to address specific categories of student inquiries, which were defined and scoped as follows:

- **Course Equivalencies:** Questions such as, "Does ECON 1A at City College SF count as UC Berkeley ECON 1A?"
- **Academic Planning:** Inquiries like, "What courses should be taken in the third semester at UCLA after transferring all credits?"
- **Other:** Miscellaneous transfer-related questions, for example, "How does one appeal a missed CSU transfer deadline?"

To effectively manage these inquiries, two key tasks were identified:

1. **Intent Classification:** Accurately assigning free-form student questions to one of the three predefined categories.
2. **Efficient Retrieval:** If a similar question has been previously encountered and classified, retrieving its stored label directly rather than re-classifying the query.

b. Data Preparation

A dataset comprising approximately 100 labeled question pairs was utilized, stored in a CSV file named transfer_prompts.csv. Each row in this file contained a Prompt and its corresponding Label. Prior to system integration, the dataset underwent verification to ensure the absence of duplicate entries, consistency in label spelling, and a balanced distribution across the defined intent classes. This dataset was then uploaded to the Qdrant vector database for use in the dense retrieval mechanism.

c. Classification Strategy

A hybrid classification strategy was implemented, combining dense retrieval with zero-shot classification:

- **Dense Retrieval:**
    - Each prompt from the transfer_prompts.csv dataset was embedded into a 384-dimensional vector using the "all-MiniLM-L6-v2" sentence transformer model.
    - These embeddings were subsequently stored in a Qdrant collection designated as intent_identification.
    - During runtime, a user's query is similarly embedded. A search is then performed within the Qdrant collection for the nearest neighbor based on cosine similarity,

with a predefined threshold of ≥0.8. If a neighbor meeting this threshold is identified, its associated label is immediately reused as the predicted intent.

- **Zero-Shot Classification:**
  - If no sufficiently similar neighbor is found through dense retrieval, the system resorts to zero-shot classification.
  - A concise, engineered prompt is constructed and sent to a pre-trained zero-shot classification model, such as "facebook/bart-large-mnli". An example prompt structure is: "You are an academic advisor. Classify the question into: Course Equivalencies, Academic Planning, or Other. Query: '. Respond with only the label."
  - The top-ranked label from the model's output is then selected as the predicted intent.

d. Feedback Loop

A critical component of the system is its feedback mechanism, designed to continuously improve classification accuracy:

- Following a prediction, the system prompts the user with the question: "Is this correct? (y/n):"
- **Correct Prediction (Yes):** If the user confirms the prediction is correct, the prompt and its validated label are stored in the Qdrant database. This reinforces the system's knowledge base and improves future retrieval efficiency for similar queries.
- **Incorrect Prediction (No):** If the user indicates the prediction is incorrect, the system attempts to suggest an alternative intent. The original prompt and the previously incorrect intent are recorded in Qdrant, marked to indicate user disagreement. The system then proposes a new intent, either by identifying a different high-probability label from the classification model (if available) or by falling back to a default alternative from the predefined labels. This newly suggested intent is also stored in Qdrant, marked as an alternate suggestion. This process allows for iterative refinement of the classification.

3. Overall Framework

- **FastAPI:** This serves as the central API layer, managing all communication and data processing.
- **React Frontend:** This constitutes the interactive user interface.
- **Python Backend:** Developed a functional and Comprehensive Backend.

4. Tools and Systems Used

- **FastAPI:** Chosen for the backend API due to its high performance, asynchronous capabilities, and ease of use for building robust web services.
- **React:** Utilized for the frontend user interface for its component-based architecture, which promotes modularity and reusability, and its efficiency in building dynamic and interactive single-page applications.
- **Hugging Face Transformers Library:** This library was fundamental for integrating state-of-the-art Natural Language Processing (NLP) models. It provided access to pre-trained models for:
  - **Sentence Embeddings:** The "sentence-transformers/all-MiniLM-L6-v2" model was used for converting text into dense vector representations, crucial for semantic search.

    - **Zero-Shot Classification:** Models like "facebook/bart-large-mnli" enabled classification of prompts into predefined categories without requiring explicit training on those specific labels, offering flexibility and rapid deployment.
    - **Large Language Models (LLMs):** Models such as "google/flan-t5-small" were incorporated for text-to-text generation tasks, providing an alternative classification mechanism through few-shot prompting. The Hugging Face ecosystem was chosen for its extensive model hub, ease of integration, and active community support.
- **Qdrant:** Selected as the vector database for efficient storage and retrieval of text embeddings. Qdrant's capabilities for high-performance similarity search were essential for the dense retrieval component of the classification strategy, allowing for rapid identification of similar historical queries.
- **Pandas:** Employed for data manipulation, particularly for reading and processing the transfer_prompts.csv dataset, facilitating the initial loading of data into Qdrant.
- **scikit-learn:** Used for generating classification reports, providing standard metrics (precision, recall, F1-score) to assess model performance.
- **TensorFlow:** Served as the underlying deep learning framework for many of the Hugging Face models used, providing the computational backbone for embedding and inference.
- **CORS Middleware (FastAPI):** Enabled cross-origin resource sharing, allowing the frontend application to securely communicate with the backend API.

5. Reflection on Generative AI Assistance

- **Code Generation (CSS and React)**
- **Architectural Guidance**
- **Debugging and Error Resolution**
- **Documentation and Explanation**
- **Prompt Engineering**

6. Conclusion

By combining vector retrieval, zero-shot classification and few-shot generation in a feedback-driven loop, I delivered a conversational chatbot that accurately interprets transfer advising queries. The GUI front end makes this system approachable for students, while the modular architecture ensures extensibility—new intents, alternate backends or richer analytics can be added with minimal effort. Overall, the project showcases how GenAI and modern vector databases empower domain-specific intent solutions on an accelerated timeline.