**Case Study: Natural Language Query Processing for Database Retrieval**

**1. Introduction**

**1.1 Background**

Querying databases using natural language allows users to interact with databases without needing to know complex query languages like SQL. This project combines natural language processing (NLP) with discrete mathematics for logical representation and Prolog for logical reasoning, aiming to build a system that translates natural language queries into formal database queries and retrieves the relevant information.

**1.2 Objective**

The objective of this project is to develop a system that processes natural language queries, translates them into formal queries, and retrieves the corresponding information from a database. This involves using Prolog and principles of discrete mathematics to facilitate the logical reasoning necessary for accurate data retrieval.

**1.3 Significance**

This project is significant because it simplifies database querying, making it accessible to users without technical expertise in query languages like SQL. By enabling natural language queries, the system enhances user experience and broadens the accessibility of database systems. This approach has potential applications in various domains, including customer service, business intelligence, and academic research.

**2. Problem Statement**

The project addresses the problem of enabling natural language queries for database retrieval. Specifically, it focuses on translating user-friendly natural language inputs into formal database queries that can be executed to retrieve the desired information. Constraints include ensuring the accuracy of the translation process, handling diverse query structures, and maintaining efficient query execution.

**3. Methodology**

**3.1 Approach**

The approach involves several key steps:

1. **Data Collection and Database Preparation**: Creating a sample database with structured data.
2. **Preprocessing**: Using NLP techniques to tokenize and parse natural language queries, and extract entities and their relationships.
3. **Formal Representation**: Translating parsed queries into formal logical expressions.
4. **Prolog Encoding**: Encoding the database schema and data as Prolog facts.
5. **Logical Reasoning with Prolog**: Implementing rules for querying the Prolog database based on logical expressions.
6. **Query Execution and Result Retrieval**: Executing translated Prolog queries against the encoded database and retrieving results.

**3.2 Tools and Techniques**

* **Python**: Used for NLP preprocessing and system integration.
* **Prolog (SWI-Prolog)**: Used for encoding the database and logical reasoning.
* **spaCy**: Used for text preprocessing and query parsing.
* **Discrete Mathematics**: Used for formal logical representation of the queries

**4. Implementation**

**4.1 System Architecture**

The system architecture consists of the following components:

* **NLP Preprocessor**: Tokenizes and parses natural language queries using spaCy.
* **Entity Extractor**: Identifies relevant entities and relationships from the parsed query.
* **Logical Translator**: Converts the extracted information into a formal logical representation.
* **Prolog Engine**: Encodes the database schema and data as Prolog facts and executes queries.
* **Query Executor**: Interfaces with the Prolog engine to retrieve results and display them.

**4.2 Code Examples:**

% Facts about books

book('Harry Potter and the Philosopher\'s Stone', 'J.K. Rowling', 1997).

book('Harry Potter and the Chamber of Secrets', 'J.K. Rowling', 1998).

book('The Hobbit', 'J.R.R. Tolkien', 1937).

% Rule to find books by a specific author

books\_by\_author(Author, BookList) :-

findall(Book, book(Book, Author, \_), BookList).

**5. Results**

**5.1 Test Cases**

**Test Case 1**:

* **Input**: ?- books\_by\_author('J.K. Rowling', BookList).
* **Expected Output**: BookList = ['Harry Potter and the Philosopher\'s Stone', 'Harry Potter and the Chamber of Secrets'].
* **Actual Output**: BookList = ['Harry Potter and the Philosopher\'s Stone', 'Harry Potter and the Chamber of Secrets'].

**Test Case 2**:

* **Input:** ?- books\_by\_author('J.R.R. Tolkien', BookList).
* **Expected Output:** BookList = ['The Hobbit'].
* **Actual Output**: BookList= ["The Hobbit"]

**Test Case 3**:

* **Input**: ?- books\_by\_author('George Orwell', BookList).
* **Expected Output**: BookList = [].
* **Actual Output**: BookList = [].

**5.2 Analysis**

The results indicate that the system correctly processes natural language queries and retrieves the appropriate data from the database. The implementation meets the objectives of translating natural language queries into formal Prolog queries and efficiently retrieving relevant information.

**6. Discussion**

**6.1 Challenges**

Challenges encountered during the project included:

* Handling the variability in natural language queries, such as different phrasings and structures.
* Ensuring accurate entity extraction and logical translation.
* Integrating Python and Prolog for seamless query execution.

**6.2 Future Work**

Future improvements could include:

* Expanding the NLP capabilities to handle more complex queries and different domains.
* Enhancing the Prolog rules to support a broader range of query types.
* Developing a more user-friendly interface, such as a web-based application.

## 7. Conclusion

This project successfully developed a system that processes natural language queries, translates them into formal queries, and retrieves the corresponding information from a database using Prolog and NLP techniques. The system demonstrates the potential for making database querying more accessible and user-friendly. Future work could further enhance the system's capabilities and extend its applicability to various domains.

**4.2 Code Examples**