Prolog European Countries Analysis System

Semen Sazonov 20221689

Introduction

This report documents the design and implementation of a Prolog-based system for analyzing data on European countries across various domains: geography, culture, economy, and politics. The goal of the system is to leverage logical programming to handle complex relationships, offering an efficient way to query and understand interconnections within European data.

1. System Architecture

1.1 Database Structure

The knowledge base is organized into four main domains, with each containing various attributes that define the relationships and characteristics of European countries:

1. **Geographic Information**
    - Major cities and capitals
    - Neighboring countries
    - Regional classifications
    - Geographical relationships
2. **Cultural Information**
    - Official languages
    - Traditional customs and national foods
    - Popular sports
    - Cultural markers and national icons
3. **Economic Information**
    - National currencies
    - Key exports
    - Prominent inventions
    - Trade relationships
4. **Political Information**
    - EU membership status
    - Government systems
    - Climate types
    - International relations

1.2 Predicate Organization

The system is built upon 18 core predicates, grouped into four categories that provide various functionalities:

1. **Basic Queries**
    - `major_city_in_countries/2`: Identifies major cities by country.
    - `region_capitals/2`: Lists capitals by region.
    - `country_pride/2`: Displays icons or inventions that symbolize national pride.
2. **Regional Analysis**
    - `regional_cuisine_specialties/3`: Maps regional foods to countries.
    - `regional_sport_popularity/2`: Highlights sports popularity by region.
    - `country_by_region_and_city/3`: Finds countries based on region and city.
3. **Comparative Analysis**
    - `common_language/3`: Identifies shared languages between countries.
    - `shared_cultural_neighbors/3`: Finds common cultural influences in neighboring countries.
    - `similar_political_system/2`: Identifies countries with similar political systems.
    - `similar_climate/2`: Groups countries by climate type.
    - `common_export/3`: Shows common exports between countries.
    - `eu_non_eu_comparison/3`: Compares exports between EU and non-EU countries.
4. **Complex Analysis**
    - `common_neighbors_with_count/4`: Counts and lists shared neighbors between countries.
    - `all_common_neighbors_with_count/4`: Generates pairs of countries based on shared neighbors.
    - `best_to_do_in_country/4`: Combines attractions, activities, and cuisine for tourism insights.
    - `cross_border_tradition/3`: Lists traditions shared by neighboring countries.

2. Implementation Details

2.1 Base Facts

The system employs a robust collection of base facts, defining the fundamental attributes and relationships for each country. Examples of these base facts include:

```
capital(Country, Capital).
neighbor(Country1, Country2).
major_city(Country, City).
region(Country, Region).
currency(Country, Currency).
```

2.2 Complex Predicates

The system includes several multi-functional predicates, which conduct advanced analyses by combining multiple base facts and relationships:

Common Language Analysis

```
common_language(Country1, Country2, Language) :-
      official_language(Country1, Language),
      official_language(Country2, Language),
      Country1 \= Country2.
```

This predicate identifies when two different countries share an official language, facilitating comparative linguistic analysis.

Shared Cultural Neighbors

```
shared_cultural_neighbors(Country1, Country2, Influence) :-
      neighbor(Country1, Country2),
      cultural_fingerprint(Country1, Influence),
      cultural_fingerprint(Country2, Influence),
      Country1 \= Country2.
```

Finds shared cultural influences between neighboring countries, useful for analyzing cultural regions.

Similar Political Systems

```
similar_political_system(Country1, Country2) :-
      political_system(Country1, System),
      political_system(Country2, System),
      Country1 \= Country2.
```

Identifies countries with matching political systems for political structure comparison.

Similar Climate Types

```
similar_climate(Country1, Country2) :-
      climate_type(Country1, Climate),
      climate_type(Country2, Climate),
      Country1 \= Country2.
```

Groups countries with similar climates, supporting environmental and tourism analysis.

2.2.2 Regional and Cuisine Analysis Predicates

Regional Cuisine Specialties

```
regional_cuisine_specialties(Region, Country, Dish) :-
      region(Country, Region),
      national_food(Country, Dish).
```

Associates regions with notable dishes, linking cuisine to geographic areas.

Common Exports

```
common_export(Country1, Country2, Export) :-
      export(Country1, Export),
      export(Country2, Export),
      Country1 \= Country2.
```

Identifies shared exports between countries, useful for economic and trade analysis.

Regional Sport Popularity

```
regional_sport_popularity(Region, Sport) :-
      region(Country, Region),
      popular_sport(Country, Sport).
```

Maps sports popularity by region, showing patterns in athletic culture.

2.2.3 EU and Economic Analysis Predicates

EU-NonEU Export Comparison

```
eu_non_eu_comparison(EUCountry, NonEUCountry, Export) :-
      eu_membership(EUCountry, yes),
      eu_membership(NonEUCountry, no),
      export(EUCountry, Export),
      export(NonEUCountry, Export).
```

Compares export activities between EU and non-EU countries.

Cross-Border Traditions

```
cross_border_tradition(Country1, Country2, Tradition) :-
      neighbor(Country1, Country2),
      tradition(Country1, Tradition),
      tradition(Country2, Tradition).
```

Highlights traditions shared by neighboring countries.

Best Activities in Country

```
best_to_do_in_country(Country, Place, Activity, Food) :-
      tourist_place(Country, Place),
      tradition(Country, Activity),
      national_food(Country, Food).
```

Provides comprehensive tourism information for a country, combining places, activities, and cuisine.

2.2.4 Geographic and Political Analysis Predicates

Non-Euro EU Countries

```
non_euro_eu_country(Country, Currency) :-
      eu_membership(Country, yes),
      currency(Country, Currency),
      Currency \= euro.
```

Identifies EU countries not using the Euro currency.

Common Neighbors

```
common_neighbor(Country1, Country2, CommonNeighbor) :-
      neighbor(Country1, CommonNeighbor),
      neighbor(Country2, CommonNeighbor),
      Country1 \= Country2,
      Country1 \= CommonNeighbor,
      Country2 \= CommonNeighbor.
```

Lists countries that share common neighboring countries.

Region Capitals

```
region_capitals(Region, Capital) :-
      region(Country, Region),
      capital(Country, Capital).
```

Lists capital cities within a specific region.

Major Cities

```
major_city_in_countries(City, Country) :-
major_city(Country, City).
```

Maps major cities to their respective countries.

Country by Region and City

```
country_by_region_and_city(Region, City, Country) :-
      region(Country, Region),
      major_city(Country, City).
```

Combines regional and urban data for identifying countries.

Country Pride

```
country_pride(Country, Pride) :-
      ( icon(Country, Pride);invention(Country, Pride) ).
```

Lists notable icons and inventions that represent national achievements.

Common Neighbors Count

```
common_neighbors_with_count(Country1, Country2, Count,
CommonNeighbors) :-
      dif(Country1, Country2),
      setof(Neighbor, (neighbor(Country1, Neighbor),
      neighbor(Country2, Neighbor)), Neighbors),
      length(Neighbors, Count),
      CommonNeighbors = Neighbors.
```

Counts and lists shared neighbors between countries.

All Common Neighbors Count

```
all_common_neighbors_with_count(Count, Country1, Country2,
CommonNeighbors) :-
    distinct((Country1, Country2, CommonNeighbors), (
           neighbor(Country1, _),
           neighbor(Country2, _),
           Country1 @< Country2,
           common_neighbors_with_count(Country1, Country2, Count,
CommonNeighbors)).
```

3. Diagrams Analysis

3.1 Prolog Relationships and Tree Diagrams

The relationship and tree diagrams provide a comprehensive visualization of how various components within the European countries database interconnect. At the core is the Country entity, which branches into multiple attributes, such as Capital, Currency, Region, Political System, Climate Type, and EU Membership. These primary relationships form the foundational structure of the knowledge base, while secondary relationships—through Cultural and Social connections (National Food, Official Language, Icons, Traditions, Popular

Sports, and Cultural Fingerprints) as well as Economic and Tourism relationships (Exports, Tourist Sites, Major Cities, and Inventions)—add further layers of depth. This diagram effectively shows how queries navigate both direct and multi-step relationships to extract nuanced information, enabling sophisticated analysis of each country's unique characteristics.

3.2 Prolog Project Structure

The project structure diagram reveals the organized, hierarchical arrangement of the knowledge base, divided into four primary categories: Geographic Info, Cultural Info, Economic Info, and Political Info. Each category encompasses specific facts and rules, creating a modular, organized codebase that enhances both readability and maintainability. This clear separation of domains enables efficient query processing and logical grouping of related information. The structure allows for the easy expansion of the knowledge base and addition of new predicates without impacting existing functionalities.

3.3 Prolog Predicates Structure

The predicate structure diagram outlines the system's two main components: Base Facts and Complex Queries. Base Facts contain fundamental predicates, such as `capital/2`, `neighbor/2`, and `major_city/2`, while Complex Queries are higher-level predicates that build upon these base facts to answer more sophisticated questions. The diagram clarifies how predicates are organized by their arity and dependencies, with arrows indicating how complex queries evolve from simpler ones. This visualization provides insight into the query resolution process and the hierarchical relationships between predicates in the system.

3.4 Predicate Dependencies

The predicate dependencies diagram maps out the logical flow between various predicates, showing how Core Facts Dependencies feed into Simple Queries and Complex Queries. Arrows indicate the flow of data and logical dependencies, illustrating how complex queries are processed through the breakdown into simpler components. This visual aid is invaluable for identifying potential bottlenecks, optimizing query processing, and maintaining or extending the system's capabilities.

3.5 Search Trees and Execution Flow

The search tree diagrams depict the execution patterns of different queries within the system. They show how Prolog traverses the knowledge base to resolve queries, highlighting backtracking points, alternative paths, success and failure nodes, variable bindings, and the sequential goal-processing order. This clear visualization of the execution path and decision points is essential for debugging and optimizing query performance, especially for complex queries involving multiple predicates and relationships.

4. Application Examples

4.1 Cultural Analysis Examples

```
% Example 1: Finding countries with shared languages
?- common_language(belgium, switzerland, Lang).
Lang = french ;
Lang = german.

% Example 2: Identifying shared cultural influences
?- shared_cultural_neighbors(greece, turkey, Influence).
Influence = ottoman_influence ;
Influence = mediterranean_culture.

% Example 3: Analyzing cross-border traditions
?- cross_border_tradition(serbia, montenegro, Tradition).
Tradition = orthodox_christianity ;
Tradition = folk_music.

% Example 4: Finding activities, places, and food in a country
?- best_to_do_in_country(italy, Place, Activity, Food).
Place = venice_canals,
Activity = renaissance_art,
Food = "Pizza" ;

Place = rome,
Activity = catholicism,
Food = "Pizza".
```

4.2 Economic Analysis Examples

```
% Example 1: Finding common exports between countries
?- common_export(germany, france, Export).
Export = machinery ;
Export = motor_vehicles ;
Export = electronics.

% Example 2: Comparing EU and non-EU exports
?- eu_non_eu_comparison(germany, switzerland, Export).
Export = machinery ;
Export = precision_instruments.

% Example 3: Analyzing regional economic patterns
```

```
?- regional_cuisine_specialties(scandinavian, Country, Dish).
Country = sweden, Dish = meatballs ;
Country = norway, Dish = lutefisk.
```

4.3 Geographic Analysis Examples

```
% Example 1: Finding countries with common neighbors
?- common_neighbors_with_count(austria, germany, Count, Neighbors).
Count = 3,
Neighbors = [czech_republic, switzerland, liechtenstein].

% Example 2: Analyzing regional capitals
?- region_capitals(balkan, Capital).
Capital = tirana ;
Capital = sarajevo ;
Capital = sofia ;
Capital = zagreb.

% Example 3: Complex regional queries
?- country_by_region_and_city(central_europe, vienna, Country).
Country = austria.

% Example 4: Finding all pairs with a specific common neighbor count
?- all_common_neighbors_with_count(3, Country1, Country2,
CommonNeighbors).
Country1 = austria,
Country2 = germany,
CommonNeighbors = [czech_republic, switzerland, liechtenstein] ;

Country1 = france,
Country2 = germany,
CommonNeighbors = [belgium, luxembourg, switzerland].
```

4.4 Political Analysis Examples

```
% Example 1: Finding similar political systems
?- similar_political_system(france, russia).
true.

% Example 2: Non-Euro EU members
?- non_euro_eu_country(Country, Currency).
Country = denmark, Currency = danish_krone ;
Country = sweden, Currency = swedish_krona.
```

```
% Example 3: Climate similarity analysis
?- similar_climate(spain, italy).
true.

% Example 4: Major city lookup in different countries
?- major_city_in_countries(berlin, Country).
Country = germany.

?- major_city_in_countries(City, france).
City = paris ;
City = marseille ;
City = lyon ;
City = toulouse.
```

4.5 Cultural Icons and Achievements Examples

```
% Example 1: Finding cultural icons
?- country_pride(germany, Pride).
Pride = albert_einstein ;
Pride = beethoven ;
Pride = automobile.

% Example 2: Technological achievements
?- country_pride(finland, Pride).
Pride = nokia ;
Pride = "Linux operating system".
```

5. Data Collection and Preparation

The data for this project was meticulously gathered from reliable and reputable sources to ensure accuracy and depth in the Prolog knowledge base. Primary sources included **Wikipedia** for general information on European countries, the **EU Official Website** for details on political affiliations and EU membership, **UNESCO's World Heritage List** for cultural and historical sites, and **World Bank Data** for economic indicators such as exports and GDP. Additionally, I used **OpenAI** resources to refine complex queries and understand relationships between data points. After compiling the data, I created an organized **Excel file** that served as a structured repository of facts and figures. This spreadsheet allowed me to systematically define rules and relationships, which I then translated into Prolog predicates.

Conclusion

This Prolog system exemplifies a robust analytical tool that integrates geographical, cultural, political, and economic data into a flexible and scalable platform. By demonstrating logical programming’s efficacy in modeling complex real-world relationships, the system stands as a powerful tool for European countries' analysis.