# Cycling Tour Recommendation System Documentation

Vikrant Singh

November 5, 2024

## Contents

# 1 Introduction

The Cycling Tour Recommendation System is developed to assist users in finding optimal cycling routes tailored to their preferences and requirements. By leveraging Natural Language Processing (NLP), geospatial analysis, and data filtering techniques, the application delivers personalized recommendations for cycling enthusiasts in the Tyrol region.

# 2 Data Loading and Preprocessing

## 2.1 Loading the Data

The system utilizes a GeoJSON file as its primary data source, containing comprehensive details of various cycling tours in the Tyrol area. The data is loaded into a pandas DataFrame, enabling efficient data manipulation and querying.

## 2.2 Extracted JSON Features

Key features extracted from the GeoJSON data include:

- **Name**: The name of the cycling tour.
- **Route Length**: Total length of the route in kilometers.
- **Elevation Gain**: Total elevation gain in meters.
- **Difficulty Level**: Categorized as Easy (0), Medium (1), or Hard (2).
- **Tour Type**: Indicates whether the tour is a Loop, or Single.
- **Round Trip**: Boolean indicating if the route is a round trip.
- **Available Months**: List of months when the tour is accessible.
- **Coordinates**: Start and end latitude and longitude for mapping.

## 2.3 Normalization into DataFrame

The GeoJSON data is parsed and normalized into a pandas DataFrame, consolidating multiple LineStrings into single routes and extracting start and end coordinates for each tour. This structured format facilitates efficient filtering and recommendation processes.

# 3 Data Features and Constraints

## 3.1 JSON Features Utilized

The application leverages various properties from the JSON data to filter and recommend tours:

- **Route Length and Elevation Gain**: Used to filter tours based on user-specified distance and difficulty.
- **Difficulty Level**: Helps in categorizing tours suitable for different user types.
- **Tour Type**: Enables filtering for specific types of tours, such as loops.
- **Availability**: Ensures tours are recommended based on seasonal accessibility.

## 3.2 Applied Constraints

To provide accurate recommendations, the system applies several constraints based on user types and preferences:

- **Route Length Range**: Filters tours within specified kilometer ranges.
- **Elevation Gain Threshold**: Limits tours based on elevation gain requirements.
- **Difficulty Level Alignment**: Ensures the difficulty of the tour matches the user's skill level or preference.
- **Seasonal Availability**: Recommends tours available during the user's preferred months.
- **Proximity to Location**: Suggests tours near a specified city or location within a defined radius.

## 3.3 User Type Specific Constraints

Different user types have tailored constraints to ensure recommendations align with their needs:

- **Family**:
  - **Route Length**:
    - Difficulty Level 0: $\leq$ 10 km
    - Difficulty Level 1: $\leq$ 15 km
  - **Elevation Gain**: $\leq$ 500 meters
- **Beginner**:
  - **Route Length**:
    - Difficulty Level 0: $\leq$ 10 km
  - **Elevation Gain**: $\leq$ 300 meters
- **Other User Types**: Additional constraints can be defined similarly based on specific requirements.

# 4 Natural Language Processing

## 4.1 Named Entity Recognition (NER)

The application employs two NER models to extract entities from user queries:

- **BERT-Based NER**: Utilizes the `dbmdz/bert-large-cased-finetuned-conll03-english` model from Hugging Face for robust entity extraction.
- **spaCy NER**: Acts as a fallback mechanism using spaCy's `en_core_web_sm` model to ensure location entities are accurately captured.

## 4.2 Zero-Shot Classification

For determining the difficulty level based on user descriptions, the system uses a zero-shot classification pipeline:

- **Model**: `facebook/bart-large-mnli` from Hugging Face.
- **Purpose**: Classifies user queries into predefined difficulty levels without requiring labeled training data.

## 4.3 Entity Ruler and Custom Patterns

Using spaCy's `EntityRuler`, custom patterns are defined to recognize specific cities and elevation-related terms, enhancing the NER capabilities to better suit the application's domain.

# 5 Query Handling

## 5.1 Types of Queries Supported

The application is designed to handle a variety of user queries, including:

- **Basic Queries**: Listing all available cycling routes.
- **Seasonal Queries**: Finding tours available in specific seasons or months.
- **Location-Based Queries**: Recommending tours near a particular city or location.
- **Difficulty and User Type Queries**: Suggesting routes based on user skill level or specific user types (e.g., families, beginners).
- **Combined Criteria Queries**: Handling queries that combine multiple filters such as location, route length, and elevation gain.

## 5.2 Parsing Queries into Criteria

The core of the query handling mechanism lies in parsing natural language inputs into structured criteria. This involves:

- **Capitalizing City Names**: Ensures accurate entity recognition by standardizing city name formats.
- **Removing Location Prefixes**: Strips unnecessary prefixes like "City" or "Town" to focus on the core location names.
- **Extracting Numerical Constraints**: Uses regular expressions to identify and parse numerical values and their corresponding units for route length and elevation gain.
- **Identifying User Types**: Detects specific user types (e.g., family, beginner) to apply relevant constraints.
- **Mapping Difficulty Levels**: Aligns extracted difficulty levels with predefined categories to filter tours appropriately.

## 5.3 Mapping Queries to Factors

Once parsed, the criteria are mapped to various factors used in filtering:

- **Route Length and Elevation Gain**: Applied as minimum or maximum thresholds based on user input.
- **Difficulty Levels**: Filters tours that match the user's skill level or preference.
- **Seasonal Availability**: Ensures tours are recommended only during specified months.
- **Proximity Constraints**: Filters tours based on their distance from a specified location.

# 6 Proximity Search

## 6.1 Geocoding with Caching

To determine the geographical coordinates of user-specified locations, the application uses the `geopy` library with the Nominatim geocoding service. To optimize performance and reduce API calls:

- **Caching Mechanism**: Implemented using Python's `shelve` module to store and retrieve previously geocoded locations.
- **Rate Limiting**: Ensures compliance with geocoding service usage policies by implementing a rate limiter.

### 6.2 Distance Calculations

The application calculates the distance between the user's location and each cycling tour's coordinates using geodesic calculations:

- **Geopy's Geodesic Function**: Determines the distance in kilometers between two geographic points.
- **Radius Filtering**: Selects tours within a specified radius (e.g., 50 km) from the user's location.

## 7 Filtering and Ranking

### 7.1 Applying Constraints

Based on the parsed criteria, the system sequentially filters the DataFrame:

- **Location Proximity**: Filters tours near the specified location.
- **Route Length**: Applies minimum and/or maximum route length constraints.
- **Elevation Gain**: Applies elevation gain constraints.
- **Difficulty Level**: Filters tours matching the desired difficulty levels.
- **Seasonal Availability**: Ensures tours are available in the preferred months.
- **Tour Type**: Filters based on tour type (e.g., loop).

### 7.2 Ranking Logic

After filtering, tours are ranked and the top N recommendations are selected based on:

- **Proximity**: Tours closer to the user's location are prioritized.
- **Route Length**: Longer routes may be preferred based on user preferences.
- **Elevation Gain**: Higher elevation gains can indicate more challenging tours.

## 8 Map Visualization

### 8.1 Using Folium for Interactive Maps

The application visualizes recommended tours on an interactive map using the `folium` library:

- **Map Centering**: Centers the map based on the user's location.
- **Markers**: Plots start and end points of each tour with customized icons to enhance readability and prevent overlap.
- **Route Paths**: Draws the actual path of the cycling tour using polylines for better visual representation.

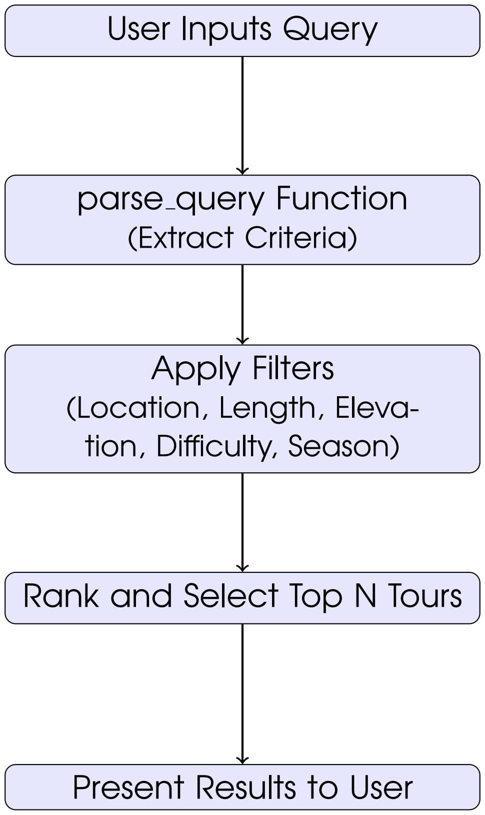
### 8.2 Customized Icons

To address overlapping issues:

- **Icon Sizes**: Adjusted the size of play and stop icons for start and end points.
- **Custom Icons**: Utilized `folium.CustomIcon` for precise control over icon dimensions.

## 9 Flow Diagram

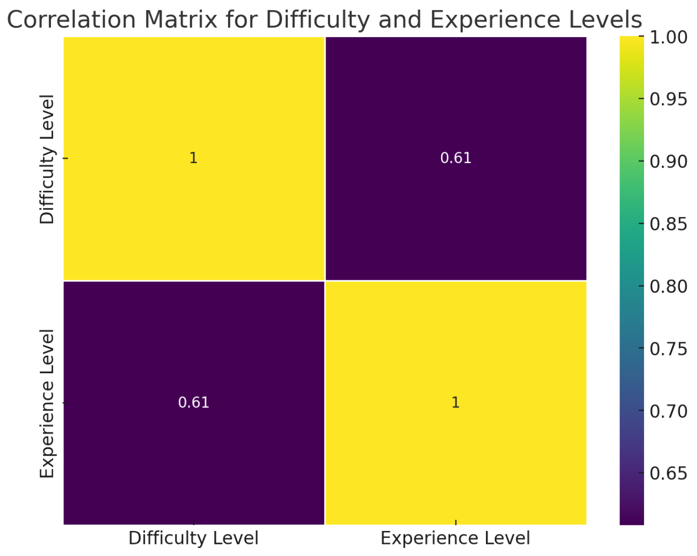
### 9.1 Query Processing Workflow

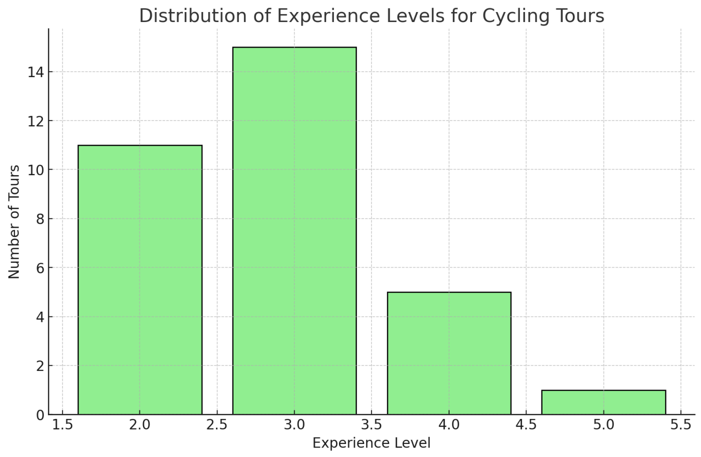

User Inputs Query
parse_query Function (Extract Criteria)
Apply Filters (Location, Length, Elevation, Difficulty, Season)
Rank and Select Top N Tours
Present Results to User

**Sample Output:**

| **Tour Name** | **Route Length (km)** | **Elevation Gain (m)** |
|---|---|---|
| Innradweg Tirol | 224 | 3429 |
| 13 - Glockner-Tour | 298.9 | 4040 |
| ... | ... | ... |

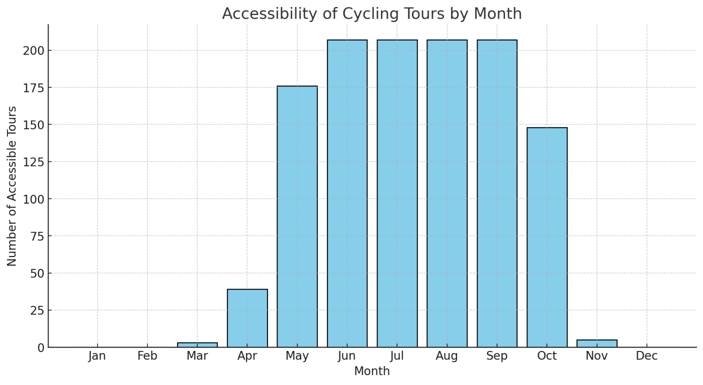
Figure 1: Flow Diagram of Query Processing

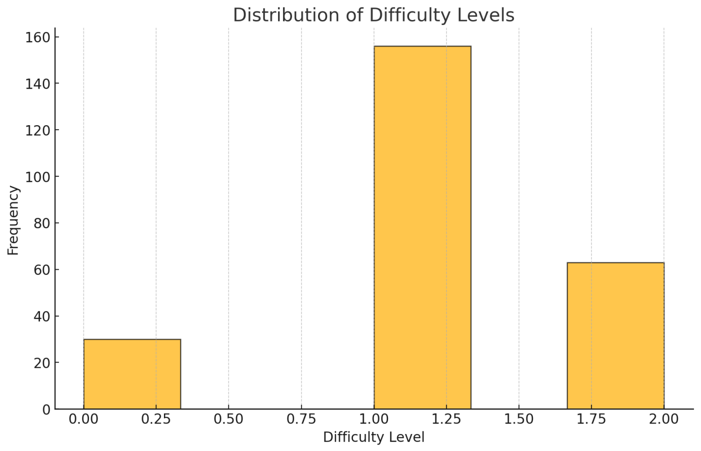


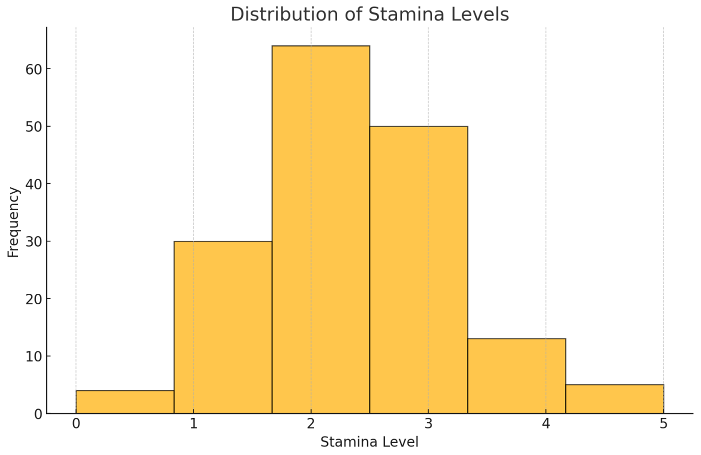


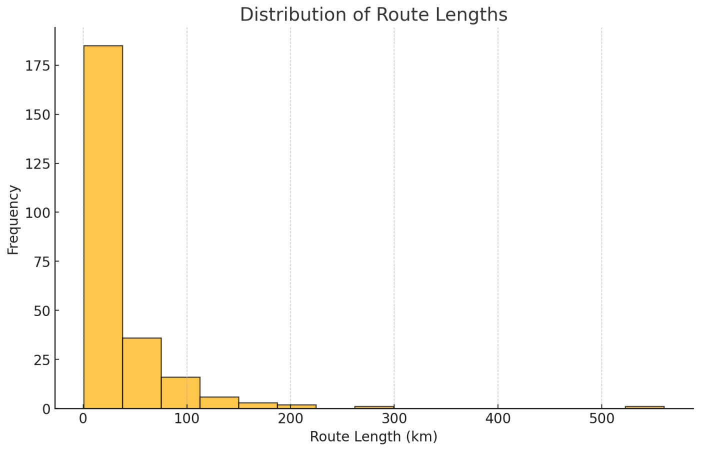
## 10 Conclusion

The Cycling Tour Recommendation System effectively integrates data processing, NLP, and geospatial analysis to deliver personalized cycling route suggestions. By refining query parsing and applying robust filtering mechanisms, the system ensures accurate and user-centric recommendations. Future enhancements may include expanding the range of supported queries, integrating additional data sources, and improving the user interface for an even more seamless experience.

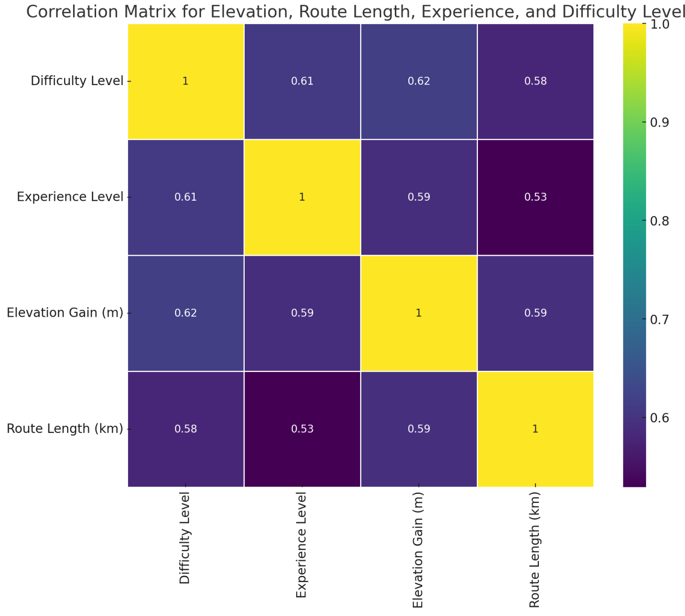

Accessibility of Cycling Tours by Month
Number of Accessible Tours
0
25
50
75
100
125
150
175
200
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Month


Distribution of Difficulty Levels
Frequency
0
20
40
60
80
100
120
140
160
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Difficulty Level


Distribution of Stamina Levels
Frequency
0
10
20
30
40
50
60
0
1
2
3
4
5
Stamina Level


Distribution of Route Lengths
Frequency
0
25
50
75
100
125
150
175
0
100
200
300
400
500
Route Length (km)

Correlation Matrix for Elevation, Route Length, Experience, and Difficulty Levels


Difficulty Level
Experience Level
Elevation Gain (m)
Route Length (km)