**Overview Of the Code**

The code implements a comprehensive map matching and traffic analysis system designed for analyzing GPS trajectories in Porto, Portugal. At its core, the system consists of two main components: the EnhancedViterbiMatcher for map matching and the RouteAnalyzer for traffic analysis.

The EnhancedViterbiMatcher is responsible for matching GPS trajectories to actual road segments. It uses a probabilistic approach based on the Hidden Markov Model (HMM) with the Viterbi algorithm. When a GPS trajectory is input, the matcher first creates a spatial index using an R-tree structure to efficiently locate nearby road segments. For each GPS point, it finds candidate road segments within an adaptive search radius, starting with a small radius and expanding if needed. The matcher calculates two types of probabilities: emission probabilities (how likely a GPS point belongs to a road segment based on distance) and transition probabilities (how likely a vehicle moved from one road segment to another based on road connectivity and turn angles).

The map matching process is enhanced with several sophisticated features. It handles GPS measurement noise through a configurable sigma parameter and includes distance decay factors to prefer closer road segments. The system also considers road network topology, preventing impossible transitions between unconnected road segments. For longer trajectories, it implements a sequential matching approach, breaking the trajectory into manageable segments with overlap to maintain consistency.

The RouteAnalyzer takes the matched trajectories and performs detailed traffic analysis. It processes the trajectory data by parsing edge IDs and timestamps, converting them into a format suitable for analysis. The analyzer calculates various statistics for each road segment, including:

* How many times each segment was traversed (traverse count)
* Average travel time through each segment
* Speed calculations
* Time taken per 100 meters of road

One of the key features of the RouteAnalyzer is its ability to identify patterns in the traffic data. It can determine the most frequently traversed road segments, which helps identify popular routes and potential bottlenecks. It also identifies the slowest road segments by calculating and comparing travel times, helping to pinpoint areas of congestion. The analyzer includes validation checks to ensure the results are reasonable, such as filtering out segments that are too short or have unrealistic speeds.

The code include visualization capabilities built on the Folium library. It creates interactive web maps that display the analysis results in an intuitive way. Road segments are color-coded based on various metrics (such as traverse frequency or average speed), and each segment includes popup information showing detailed statistics. The visualization includes a legend, custom markers, and background mapping to provide context.

**More Specific Information On The Code:**

**1. Primary Objectives of the Code.**

The primary objectives are to identify the most frequently traversed road segments, determine areas of slow traffic flow, and provide visualizations for analysis purposes.

The solution implements a structured four-stage pipeline that combines probabilistic modeling, spatial indexing, and statistical processing. At its core, the system utilizes theoretical frameworks from graph theory, Hidden Markov Models (HMM), and spatial geometry to deliver robust and actionable insights.

**2. System Architecture and Core Components**

**2.1 Data Processing**

The Data Processing comprises of two essential data sources that work in tandem to enable accurate traffic pattern analysis.

The system processes data through several stages:

1. Initial data validation to ensure coordinate and timestamp integrity
2. Conversion of coordinates to a consistent coordinate reference system
3. Organization of trajectory points into coherent sequences
4. Preparation of data structures for efficient matching operations

**Data Processing - Road Network Processing**

The system begins by acquiring road network data through OpenStreetMap using the OSMNX library. This data is structured as a directed graph G(V,E), representing the complete road network of the target area. In this mathematical representation, V denotes the set of vertices (intersections in the road network), while E represents the edges (road segments connecting these intersections).

Each edge in the graph carries multiple attributes essential for analysis:

* Geometric properties defining the physical shape and length of the road segment
* Unique identifiers (OSMID) for consistent reference
* Topological connections indicating how segments connect to form the complete network

The system initializes with several carefully calibrated parameters that govern its behavior:

# Core system parameters

SIGMA\_Z = 15.0           # GPS measurement noise parameter

MAX\_DISTANCE = 50.0      # Maximum search distance for candidate segments

TURN\_ANGLE\_THRESHOLD = pi / 4    # 45 degrees threshold for turn penalties

MIN\_TRANSITION\_PROB = 1e-5       # Minimum transition probability

These parameters are crucial for the system's operation:

* SIGMA\_Z determines the system's tolerance for GPS measurement noise
* MAX\_DISTANCE sets the boundary for searching candidate road segments
* TURN\_ANGLE\_THRESHOLD helps identify unrealistic turns in trajectories
* MIN\_TRANSITION\_PROB ensures numerical stability in probability calculations

**Data Processing - GPS Trajectory Data Processing**

The second key component of the Data Processing handles GPS trajectory data, which is stored in a structured CSV format. This data contains three essential elements:

1. Coordinate Pairs:
   * Latitude and longitude values defining precise geographical locations
   * Each pair represents a single point in a vehicle's trajectory
   * Coordinates are stored with high precision to ensure accurate mapping
2. Timestamps:
   * Recorded at 15-second intervals for optimal data density
   * This interval balances data resolution with storage efficiency
   * Timestamps enable calculation of travel times and velocities
3. Matched Path Information:
   * Contains pre-processed path matching data
   * Enables validation of matching algorithms
   * Provides ground truth for system accuracy assessment

**2.2 Map Matching Engine**

The Map Matching Engine represents a core component that implements a probabilistic approach through the EnhancedViterbiMatcher class. This engine utilizes Hidden Markov Models (HMM) and the Viterbi algorithm to accurately map GPS trajectories to the road network.

**Theoretical Foundation**

The map matching process relies on two fundamental probabilistic components that work together to ensure accurate trajectory mapping: the emission probability and the transition probability.

**Emission Probability**

The emission probability, denoted as P(o|s), determines how likely a GPS observation point corresponds to a particular road segment. This probability is calculated using a compound function:

P(o|s) = exp(-d²/2σ²) × exp(-d × α)

Where:

* d represents the perpendicular distance from the GPS point to the road segment
* σ is the GPS measurement noise parameter, calibrated to 15.0 meters based on empirical testing
* α is the distance decay factor, set to 0.85 to model the decreasing probability of matches at greater distances

This formulation combines a Gaussian distribution component (first term) with an exponential decay component (second term) to create a robust probability model that accounts for both GPS measurement noise and physical constraints.

**Transition Probability**

The transition probability, P(s₁|s₂), evaluates the likelihood of transitioning between consecutive road segments. It is computed as:

P(s₁|s₂) = connectivity\_score × angle\_score

Where:

* The connectivity\_score is binary: 1.0 for connected segments and 0.3 for disconnected segments
* The angle\_score is calculated as: 1 - (turn\_angle/max\_angle) × penalty\_max
* turn\_angle represents the angle between consecutive segments
* max\_angle is set to π/2 (90 degrees)
* penalty\_max determines the maximum penalty for sharp turns

**Implementation Architecture**

The matcher implementation incorporates three other features that work in concert to ensure both accuracy and efficiency:

**Spatial Indexing System**

The spatial indexing system utilizes an R-tree data structure to optimize spatial queries. This implementation:

* Organizes road segments in a hierarchical tree structure
* Enables efficient spatial searches with O(log n) complexity
* Significantly reduces the computational overhead of candidate segment identification
* Maintains a balanced tree structure for consistent performance

**Adaptive Candidate Search**

The candidate search mechanism implements an adaptive radius approach that:

1. Initiates searches with a 30.0-meter radius
2. Dynamically expands the search area up to 50.0 meters when necessary
3. Employs a ranking system for candidate segments based on:
   * Perpendicular distance to the GPS point
   * Segment orientation relative to the trajectory
   * Historical matching probabilities
4. Returns a prioritized list of candidates, limited to the most probable matches

**Sequential Matching Process**

For handling extended trajectories, the system implements a sequential matching process. The sequential matching process is implemented in the EnhancedViterbiMatcher class, specifically within the \_sequential\_matching method. This method handles the breakdown of long trajectories into smaller segments, processes each independently, and ensures smooth transitions and continuity. The key features that are carried out in the method are:

1. Divides long trajectories into manageable 30-point segments
2. Maintains a 10-point overlap between adjacent segments to ensure continuity
3. Processes each segment independently while preserving global context
4. Implements a merging strategy that resolves conflicts in overlap regions to ensures smooth transitions between segments and maintains trajectory consistency
   * The overlap region is resolved by removing the last overlapping points from the previous segment before appending the current segment's path.

The combination of these three features enables the Map Matching Engine to process large volumes of trajectory data efficiently while maintaining high accuracy. The system achieves this by:

* Minimizing the search space through efficient indexing
* Adapting to varying GPS quality and road network density
* Maintaining continuity in trajectory matching

**Map Matching Configuration Parameters:**

**1. max\_candidates: 20**

**Purpose**: Limits the number of candidate road segments considered for each GPS point.

**Reasoning**:

* Prevents computational explosion in dense road networks
* Focuses on the most likely matches
* Balance between accuracy and performance
* 20 candidates typically sufficient for urban environments

**2. max\_distance: 1000.0 (meters)**

**Purpose**: Defines the maximum search radius when looking for candidate road segments.

**Significance**:

* Sets outer boundary for candidate search
* Handles GPS points far from road network
* Prevents considering irrelevant distant segments
* 1000m provides generous coverage for GPS errors

**3. sigma\_z: 50.0**

**Purpose**: GPS measurement noise parameter in the emission probability calculation.

**Role**:

* Models GPS measurement uncertainty
* Larger values are more tolerant of GPS errors
* Smaller values require closer matches
* 50.0 meters accommodates typical urban GPS error

**4. beta: 2.0**

**Purpose**: Transition scoring parameter that influences route connectivity preferences.

**Impact**:

* Controls weight of topological connectivity
* Higher values favor connected segments
* Lower values allow more flexible matching
* 2.0 provides balanced connectivity preference

**5. min\_prob\_norm: 1e-7**

**Purpose**: Minimum probability threshold to prevent numerical underflow.

**Importance**:

* Prevents computational instability
* Handles extremely unlikely matches
* Maintains numerical stability in HMM
* Small enough to not affect valid probabilities

**6. angle\_tolerance: np.pi/2 (90 degrees)**

**Purpose**: Maximum allowed turn angle between consecutive segments.

**Significance**:

* Controls realistic vehicle movement patterns
* Penalizes unrealistic sharp turns
* 90 degrees allows normal urban turning patterns
* Helps filter implausible matches

**7. distance\_decay: 0.85**

**Purpose**: Factor controlling how quickly probability decreases with distance.

**Effects**:

* Shapes the distance-probability relationship
* Higher values cause faster probability decay
* Lower values are more tolerant of distance
* 0.85 provides gradual but significant decay

**Parameters’ Relationships and Trade-offs:**

1. **Accuracy vs. Performance**
   * max\_candidates and max\_distance affect search space
   * Higher values increase accuracy but reduce performance
2. **Flexibility vs. Constraint**
   * sigma\_z and angle\_tolerance control matching flexibility
   * More flexible parameters may reduce precision
3. **Stability vs. Sensitivity**
   * min\_prob\_norm and beta affect algorithm stability
   * Balance needed between stability and sensitivity

**2.3 Traffic Analysis Engine**

The Traffic Analysis Engine, implemented through the RouteAnalyzer class, serves as the analytical core of the system, processing matched trajectories to extract meaningful traffic patterns and insights. This component uses statistical methods and validation criteria to ensure the reliability of its analysis.

**Statistical Analysis Framework**

The engine implements two primary analytical components that work in tandem to provide comprehensive traffic insights:

**Traverse Count Analysis**

The system maintains a detailed count of trajectories passing through each road segment. For every segment s in the network, the traverse count is calculated as the sum of all trajectories that include that segment:

count(s) = Σ trajectories containing s

This calculation is implemented through an efficient aggregation process that:

1. Maintains a running count for each segment
2. Updates counts in real-time as trajectories are processed
3. Handles partial trajectories appropriately
4. Accounts for trajectory direction and segment orientation

**Time-Based Calculations**

The engine performs three critical time-based calculations for each road segment:

1. Average Travel Time:

avg\_time(s) = total\_time(s) / count(s)

This calculation provides the mean time vehicles spend traversing each

segment.

1. Normalized Time per 100 meters:

time\_per\_100m(s) = (avg\_time(s) / length(s)) × 100

This normalization enables fair comparison between segments of different

lengths.

1. Average Speed:

speed(s) = length(s) / avg\_time(s)

This calculation provides insights into traffic flow efficiency.

**Validation Framework**

The Traffic Analysis Engine implements a comprehensive validation framework to ensure data quality and reliability. This framework consists of several key criteria:

**Length Validation**

* Enforces a minimum segment length of 50 meters
* Ensures meaningful analysis of traffic patterns
* Eliminates noise from very short segments
* Helps maintain statistical significance

**Speed Validation**

The system enforces realistic speed bounds:

* Minimum speed: 0.1 meters per second (0.36 km/h)
  + Accounts for extreme congestion
  + Filters out potential GPS errors
  + Maintains data integrity
* Maximum speed: 33.3 meters per second (120 km/h)
  + Aligns with urban speed limits
  + Filters out unrealistic measurements
  + Ensures data reliability

**Count Validation**

The system requires:

* Positive traverse counts for all segments
* Minimum threshold for statistical significance
* Consistent counting methodology
* Error checking for count aggregation

**2.4 Visual Output**

**Overview**

The code generates two separate interactive HTML maps that visualize traffic patterns in Porto, Portugal. Each map can be opened in a web browser and provides different insights into traffic patterns.

**Output Format**

* Two separate HTML files:
  1. most\_traversed\_segments.html
  2. slowest\_segments.html

**Map 1: Most Traversed Segments Map**

**Visual Appearance**

1. **Base Map**
   * Shows Porto's street layout in light gray
   * Centered at Porto's coordinates (41.1579, -8.6291)
   * Zoom level set to show city overview
2. **Highlighted Segments**
   * Top 10 most frequently traversed road segments
   * Color gradient from light cream to deep red:
     + Light cream (#fff7ec): Lower traffic
     + Orange tones: Medium traffic
     + Deep red (#990000): Highest traffic
   * Segments appear as thick lines (6 pixels wide)
   * Each segment numbered (#1 through #10)
3. **Information Display** When clicking a segment, a popup shows:
   * Rank (1-10)
   * Edge ID (road identifier)
   * Count (number of times traversed)
   * Length of segment in meters

**Map 2: Slowest Segments Map**

**Visual Appearance**

1. **Base Map**
   * Identical base layout to Map 1
   * Same light gray street network background
2. **Highlighted Segments**
   * Top 10 slowest road segments
   * Color gradient from light blue to deep purple:
     + Light blue (#f7fcfd): Faster segments
     + Medium blue tones: Moderate speed
     + Deep purple (#6e016b): Slowest segments
   * Same line thickness as Map 1
   * Numbered segments (#1 through #10)
3. **Information Display** Clicking a segment shows:
   * Rank (1-10)
   * Edge ID
   * Time per 100 meters in seconds
   * Segment length in meters
   * Average speed in meters per second