US Road Network Analysis

A Graph Data Science Approach using Neo4j

Patrick Bikomagu Josue Ukuri sincere Gabin mbishinzemungu

Course: MSDA9215: Big Data Analytics
Instructor: Temitope Oguntade

January 2026

Agenda

Project Overview

Goal: Analyze the connectivity of the US Road Network to identify critical infrastructure and optimize routing.

The Challenge:

- 87,000+ Intersections.
- Relational DBs struggle with pathfinding.
- Need to find "Bottlenecks" (Centrality).

The Solution:

- **Neo4j:** Native Graph Database.
- **GDS Library:** For Dijkstra & Centrality.
- **Python/Plotly:** For interactive Dashboards.

Data Pre-processing (ETL)

Problem: The raw dataset provided (x, y) coordinates but **missing edge weights** (distances).

Solution: We calculated Euclidean Distance in Python before ingestion.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Why this matters?

Without this, "Shortest Path" would only count the *number of roads* (hops), not the *actual distance* traveled.

Data Loading (Cypher)

We ingested the CSVs and indexed the IDs for $O(logn)$ retrieval speed.

```
// Load Roads with Calculated Distance
LOAD CSV WITH HEADERS FROM 'file:///roads.csv' AS row
MATCH (source:Intersection {id: toInteger(row.source)})
MATCH (target:Intersection {id: toInteger(row.target)})
CREATE (source)-[:ROAD {distance: toFloat(row.distance)}]->(target)
    ;

// Create Index
CREATE INDEX intersection_id FOR (i:Intersection) ON (i.id);
```

Task 1: Network Scale

We analyzed the topology of the full continental network.

- **Total Intersections:** 87,575
- **Total Roads:** 121,961
- **Average Degree:** ≈ 2.8 (Very Sparse)

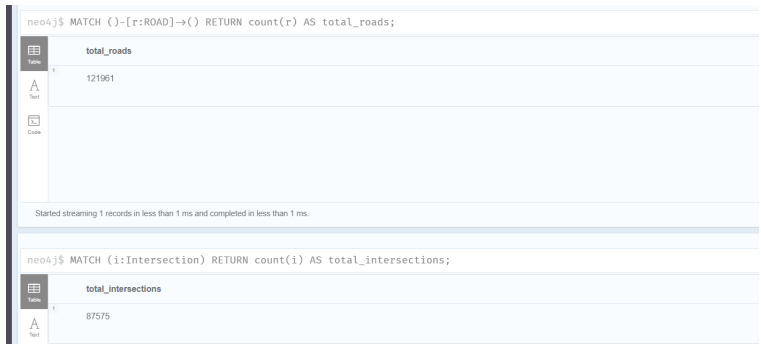

Figure: count

Neo4j Result: 87k Nodes / 121k Edges

Task 2: Pathfinding (Dijkstra)

We used the **Graph Data Science (GDS)** library to project the graph into memory.

```
CALL gds.graph.project(
    'road-network',
    'Intersection',
    {
        ROAD: {
            orientation: 'UNDIRECTED',
            properties: 'distance'
        }
    }
);
```

Result: Dijkstra finds the physically shortest route, not just the fewest turns.

Task 2: Pathfinding Visualization

path	total_distance	number_of_roads
[0, 18, 34, 33, 27, 26, 31, 25, 22, 21, 24, 35, 41, 47, 48, 49, 52, 63, 64, 66, 69, 67, 54, 59, 53, 90, 134, 135, 132, 115, 100]	361.2	30

Figure: Enter Caption

Figure: Neo4j Browser Visualization of the Route

Task 4: Betweenness Centrality

Objective: Identify network "Bridges" and vulnerabilities.

- **Concept:** Nodes that appear on many shortest paths between other nodes.
- **Finding:** High Centrality $\neq$ High Degree. A small intersection connecting two huge clusters is critical.

intersection_id	score
30524	682533356.1824204
30645	636094005.7223902
30625	630815144.5084169
30643	630706197.0096567
30621	630679549.6384052
70035	625307761.2129544
30611	623683761.5709125

Dashboard Architecture

We built an interactive tool to visualize Tasks 5-9.

- **Backend:** Neo4j (Cypher Queries)
- **Frontend:** Python + Plotly
- **Metric:** Connectivity Distribution

Visualizing Connectivity (Task 6 & 8)

Degree Distribution

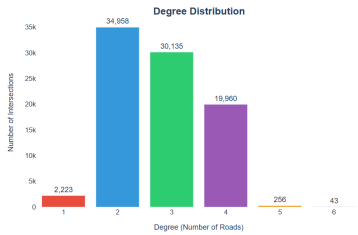

Figure: Enter Caption

Categories

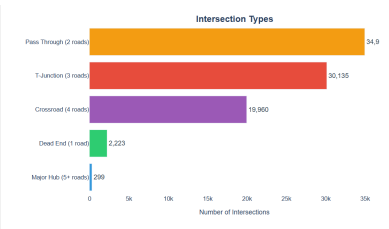

Figure: Enter Caption

Dead Ends vs. Hubs

Conclusion

1. **Success:** Modeled 87k nodes and solved the "missing distance" problem via ETL.
2. **Performance:** GDS library allows for sub-second pathfinding across the US.
3. **Insight:** The US network is sparse; connectivity relies on specific "Bridge" nodes (High Betweenness).
4. **Value:** The Plotly dashboard provides immediate visual insight into network topology.

Thank You!

Questions?