Notebook

November 26, 2024

[2]:

```python
try:
    # Import necessary libraries
    import os
    import requests
    import logging
    from typing import Dict, List, Optional, Set, Tuple
    from collections import defaultdict
    from IPython.display import display
    import requests
    import networkx as nx
    import matplotlib.pyplot as plt
    import folium
    from geopy.distance import geodesic
    from dotenv import load_dotenv

    # Configure logging
    logging.basicConfig(level=logging.INFO, format='%(message)s')
    logger = logging.getLogger(__name__)

    load_dotenv('../Docs/.env')
    app_key = os.getenv('app_key')

    if not app_key:
        logger.error("No TfL API key found")

except Exception as e:
    print(f"Error : {e}")
```

[3]:

```python
base_url = 'https://api.tfl.gov.uk'

# Get all stations and their information
# Official TfL line colors
line_colors = {
    'bakerloo': '#B36305',
    'central': '#E32017',
    'circle': '#FFD300',
    'district': '#00782A',
    'hammersmith-city': '#F3A9BB',
    'jubilee': '#A0A5A9',
    'metropolitan': '#9B0056',
    'northern': '#000000',
    'piccadilly': '#003688',
    'victoria': '#0098D4',
    'waterloo-city': '#95CDBA'
}

# Station colors
station_colors = {
    'regular': 'yellow',
    'interchange': 'red',
    'edge': 'black'
}
```

[4]:

```
# Fetch stations
try:
    response = requests.get(
        'https://api.tfl.gov.uk/StopPoint/Mode/tube',
        params={'app_key': app_key}
    )
    response.raise_for_status()
    stations = response.json()
except requests.RequestException as e:
    logger.error(f"Error fetching station data: {e}")
    stations = None

# Fetch lines
try:
    response = requests.get(
        'https://api.tfl.gov.uk/Line/Mode/tube',
        params={'app_key': app_key}
    )
    response.raise_for_status()
    lines = response.json()
except requests.RequestException as e:
    logger.error(f"Error fetching line data: {e}")
    lines = None
```

[5]:

```
# Initialize data structures
station_lines = defaultdict(set)
line_stations = defaultdict(set)

# Process stations
valid_stations = {}
for station in stations['stopPoints']:
```

```python
    if station['stopType'] == 'NaptanMetroStation' and 'tube' in station['modes']:
        station_id = station['naptanId']
        valid_stations[station_id] = {
            'name': station['commonName'],
            'lat': station['lat'],
            'lon': station['lon']
        }

# Process lines and their stations
for line in lines:
    line_id = line['id']
    try:
        response = requests.get(
            f'https://api.tfl.gov.uk/Line/{line_id}/Route/Sequence/all',
            params={'app_key': app_key}
        )
        response.raise_for_status()
        line_data = response.json()

        if 'orderedLineRoutes' in line_data:
            for route in line_data['orderedLineRoutes']:
                for station_id in route['naptanIds']:
                    if station_id in valid_stations:
                        station_lines[station_id].add(line_id)
                        line_stations[line_id].add(station_id)
    except requests.RequestException as e:
        logger.warning(f"Error fetching data for line {line_id}: {e}")

# Filter lines with at least 5 stations
valid_lines = {
    line_id: stations
    for line_id, stations in line_stations.items()
    if len(stations) >= 5
}

# Select top 5 lines with most stations
if len(valid_lines) >= 5:
    selected_lines = dict(
        sorted(valid_lines.items(), key=lambda x: len(x[1]), reverse=True)[:5]
    )

    # Get stations for selected lines
    selected_stations = set()
    for stations_set in selected_lines.values():
        selected_stations.update(stations_set)
    analysis = {
        'stations': {
            station_id: {
                **valid_stations[station_id],
                'lines': station_lines[station_id]
            }
            for station_id in selected_stations
        },
        'lines': selected_lines,
        'stats': {
            'num_stations': len(selected_stations),
            'num_lines': len(selected_lines),
            'lines_info': {
                line: len(stations)
                for line, stations in selected_lines.items()
            }
        }
    }
else:
    logger.error("Insufficient data meeting requirements")
    analysis = None
```

[6]:

```
# Initialize the graph
G = nx.Graph()

# Add stations to the graph
for station_id, station_data in analysis['stations'].items():
    G.add_node(
        station_id,
        name=station_data['name'],
        pos=(station_data['lon'], station_data['lat']),
        lines=station_data['lines']
    )

# Add connections between stations
for line_id in analysis['lines'].keys():
    try:
        response = requests.get(
            f'https://api.tfl.gov.uk/Line/{line_id}/Route/Sequence/all',
            params={'app_key': app_key}
        )
        response.raise_for_status()
        line_data = response.json()

        if 'orderedLineRoutes' in line_data:
            for route in line_data['orderedLineRoutes']:
                stations_list = route['naptanIds']
                for i in range(len(stations_list) - 1):
                    start_station = stations_list[i]
                    end_station = stations_list[i + 1]
                    if (G.has_node(start_station)
                        and G.has_node(end_station)
                        and not G.has_edge(start_station, end_station)):

                        # Calculate distance between stations
                        start_pos = G.nodes[start_station]['pos']
                        end_pos = G.nodes[end_station]['pos']
                        distance = geodesic(
                            (start_pos[1], start_pos[0]),
                            (end_pos[1], end_pos[0])
                        ).kilometers

                        # Add edge to the graph
                        G.add_edge(
                            start_station,
                            end_station,
                            line=line_id,
                            color=line_colors.get(line_id.lower(), '#808080'),
                            distance=round(distance, 2)
                        )
    except requests.RequestException as e:
        logger.error(f"Error fetching line data for {line_id}: {e}")
```

[7]:

```
# Log analysis results
logger.info("Network Analysis:")
logger.info("Selected Lines:")
for line, stations in analysis['lines'].items():
    logger.info(f"- {line}: {len(stations)} stations")
logger.info(f"Total Stations: {analysis['stats']['num_stations']}")
```

```
Network Analysis:
Selected Lines:
- district: 60 stations
- piccadilly: 53 stations
- northern: 52 stations
- central: 49 stations
- circle: 36 stations
Total Stations: 212
```

[8]:

```
# Create an interactive map centered on London
m = folium.Map(
    location=[51.5074, -0.1278],
    zoom_start=12,
    tiles='cartodbpositron'
)
```

```python
# Add lines to the map
for (u, v, data) in G.edges(data=True):
    line_coords = [
        [G.nodes[u]['pos'][1], G.nodes[u]['pos'][0]],
        [G.nodes[v]['pos'][1], G.nodes[v]['pos'][0]]
    ]

    # Create popup content
    popup_html = f"""
        <div style="font-family: Arial, sans-serif;">
            <h4>{data['line']} Line</h4>
            <p>Distance: {data['distance']:.2f} km</p>
            <p>Stations: {G.nodes[u]['name']} → {G.nodes[v]['name']}</p>
        </div>
    """

    # Add the line to the map
    folium.PolyLine(
        line_coords,
        color=data['color'],
        weight=3,
        opacity=0.7,
        popup=folium.Popup(popup_html, max_width=300)
    ).add_to(m)

# Add stations to the map
for node, data in G.nodes(data=True):
    is_interchange = len(data['lines']) > 1

    # Create popup content
    popup_html = f"""
        <div style="font-family: Arial, sans-serif;">
            <h4>{data['name']}</h4>
            <p>Lines: {', '.join(data['lines'])}</p>
            <p>Type: {'Interchange' if is_interchange else 'Regular'} Station</
  ↪p>
        </div>
    """

    # Add the station to the map
    folium.CircleMarker(
        location=[data['pos'][1], data['pos'][0]],
        radius=8 if is_interchange else 5,
        color=station_colors['edge'],
        fill=True,
        fill_color=station_colors['interchange'] if is_interchange else␣
  ↪station_colors['regular'],
        weight=2 if is_interchange else 1,
        popup=folium.Popup(popup_html, max_width=300)
    ).add_to(m)

# Display the map
display(m)
```

```
<folium.folium.Map at 0x12039a920>
```

[9]:

```
# Optionally save the map to an HTML file
m.save('TfL-Map.html')
```

[10]:

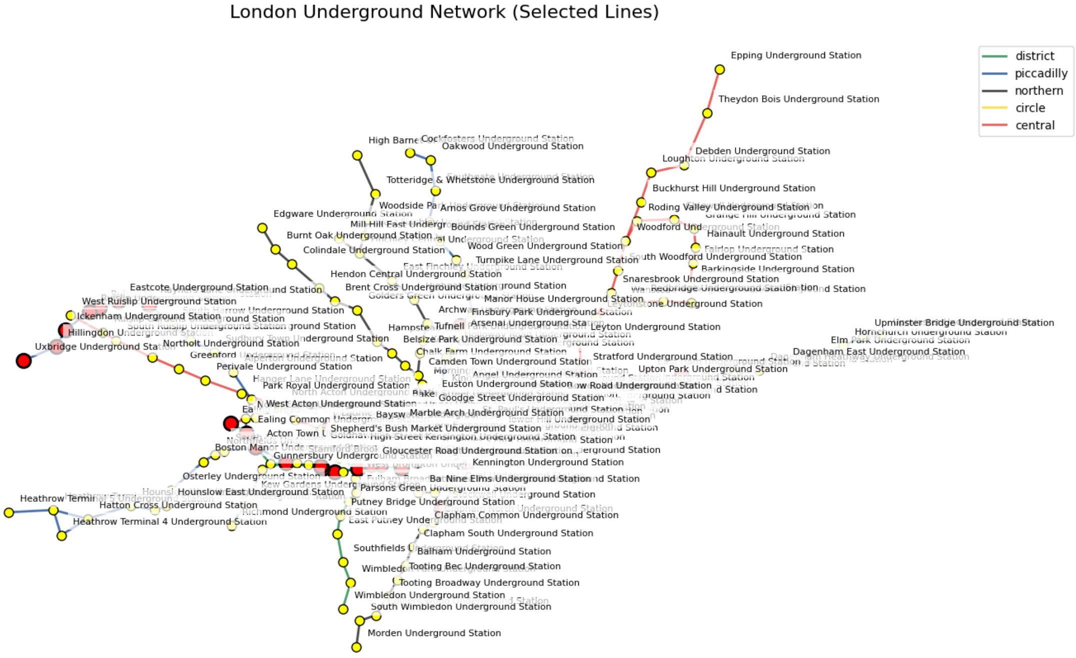
```
plt.figure(figsize=(15, 10))

# Use geographic coordinates for plotting
pos = {node: data['pos'] for node, data in G.nodes(data=True)}

# Draw edges (lines)
drawn_lines = set()
for (u, v, data) in G.edges(data=True):
    plt.plot(
        [pos[u][0], pos[v][0]],
        [pos[u][1], pos[v][1]],
        color=data['color'],
        linewidth=2,
        alpha=0.7,
        label=data['line'] if data['line'] not in drawn_lines else "",
        zorder=1
    )
    drawn_lines.add(data['line'])

# Draw stations
for node, data in G.nodes(data=True):
    is_interchange = len(data['lines']) > 1
    plt.plot(
        data['pos'][0],
        data['pos'][1],
        'o',
        color=station_colors['interchange'] if is_interchange else␣
  ↪station_colors['regular'],
        markeredgecolor=station_colors['edge'],
        markersize=12 if is_interchange else 8,
        markeredgewidth=2 if is_interchange else 1,
        zorder=2
    )
    # Add station labels
    plt.annotate(
        data['name'],
        (data['pos'][0], data['pos'][1]),
        xytext=(10, 10),
        textcoords='offset points',
        fontsize=8,
        bbox=dict(facecolor='white', edgecolor='none', alpha=0.7),
        zorder=3
    )

plt.title("London Underground Network (Selected Lines)", pad=20, fontsize=16)
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.axis('off')
plt.show()
```


London Underground Network (Selected Lines)
district
piccadilly
northern
circle
central
Epping Underground Station
Theydon Bois Underground Station
Debden Underground Station
Loughton Underground Station
Buckhurst Hill Underground Station
Roding Valley Underground Station
Grange Hill Underground Station
Woodford Underground Station
Hainault Underground Station
Fairlop Underground Station
South Woodford Underground Station
Barkingside Underground Station
Snaresbrook Underground Station
Redbridge Underground Station
Leyton Underground Station
Stratford Underground Station
Upton Park Underground Station
Upminster Bridge Underground Station
Hornchurch Underground Station
Elm Park Underground Station
Dagenham East Underground Station
High Barnet
Cockfosters Underground Station
Oakwood Underground Station
Totteridge & Whetstone Underground Station
Southgate Underground Station
Woodside Park
Arnos Grove Underground Station
Edgware Underground Station
Mill Hill East Underground Station
Bounds Green Underground Station
Burnt Oak Underground Station
Colindale Underground Station
Wood Green Underground Station
Turnpike Lane Underground Station
East Finchley Underground Station
Hendon Central Underground Station
Brent Cross Underground Station
Golders Green Underground Station
Manor House Underground Station
Archway
Finsbury Park Underground Station
Arsenal Underground Station
Hampstead
Tufnell Park
Belsize Park Underground Station
Chalk Farm Underground Station
Camden Town Underground Station
Mornington Crescent
Angel Underground Station
Euston Underground Station
Goodge Street Underground Station
Eastcote Underground Station
West Ruislip Underground Station
Ickenham Underground Station
Hillingdon Underground Station
Uxbridge Underground Station
Northolt Underground Station
Greenford Underground Station
Perivale Underground Station
Hanger Lane Underground Station
Park Royal Underground Station
North Acton Underground Station
West Acton Underground Station
Ealing Common Underground Station
Acton Town Underground Station
Baker Street
Bayswater
Marble Arch Underground Station
Holland Park Underground Station
Shepherd's Bush Market Underground Station
High Street Kensington Underground Station
Boston Manor Underground Station
Gunnersbury Underground Station
Gloucester Road Underground Station
Kennington Underground Station
Osterley Underground Station
Kew Gardens Underground Station
Nine Elms Underground Station
Heathrow Terminal 5 Underground Station
Hounslow East Underground Station
Parsons Green Underground Station
Hatton Cross Underground Station
Putney Bridge Underground Station
Richmond Underground Station
East Putney Underground Station
Clapham Common Underground Station
Heathrow Terminal 4 Underground Station
Clapham South Underground Station
Southfields Underground Station
Balham Underground Station
Wimbledon Park
Tooting Bec Underground Station
Tooting Broadway Underground Station
Wimbledon Underground Station
South Wimbledon Underground Station
Morden Underground Station

[11]:

```
# Calculate network statistics
stats = {
    'num_stations': G.number_of_nodes(),
    'num_connections': G.number_of_edges(),
    'num_interchanges': sum(
        1 for _, data in G.nodes(data=True)
        if len(data['lines']) > 1
    ),
    'total_distance': sum(
        data['distance']
        for _, _, data in G.edges(data=True)
    ),
    'is_connected': nx.is_connected(G)
}

# Display network statistics
logger.info("Network Statistics:")
for key, value in stats.items():
    logger.info(f"{key}: {value}")
```

```
Network Statistics:
num_stations: 212
num_connections: 230
num_interchanges: 78
total_distance: 272.32000000000005
is_connected: True
num_stations: 212
num_connections: 230
num_interchanges: 78
total_distance: 272.32000000000005
is_connected: True
```