Contribution of economic sectors to GVA values for all countries over time

Introduction

My project analyzes economic data about GVA from the UN database website to compute the influence of various economic sectors over time using a PageRank algorithm. The dataset contains Gross Value Added (GVA) information across sectors such as Agriculture, hunting, forestry and fishing, Industry, and Services, which is processed to determine how these sectors interact and influence one another. GVA, gross value added, is used to indicate the economic performance. It is an economic productivity metric, similar to GDP, but GVA focuses on individual industry outputs. This data set looks at the GVA value added by different primary/ main economic activities of countries over time.

The project models economic sectors as nodes in a graph, where edges represent transitional probabilities based on average GVA values. Random walks simulate sector interactions, and PageRank scores quantify sector influence. The main goal of this project is to look at how each sector contributes to GVA overtime for all countries and see which economic sector is the most prominent in the world.

Code Explanation

My project is divided into 4 modules:

utils.rs

The utils.rs module provides a utility function, parse_csv, designed to read and process economic GVA data from a CSV file. It uses the csv crate's ReaderBuilder to handle flexible parsing of the file specified by the file_path parameter. The function initializes a CSV reader, configured to accommodate non-standard CSV formats, and ensures that the file opens successfully with error handling using expect. Once the file is opened, the function iterates over each record in the CSV.

For each record, the function extracts the value from the Series column (the fourth column, indexed as 3) and the numeric Value column (the fifth column, indexed as 4). It converts the Value column into a f64 type, defaulting to 0.0 if parsing fails. The function uses f64 for numeric values to handle decimal precision in the GVA data and defaults invalid or missing entries to 0.0 for robust processing. If the Series field is not empty, the function stores the series name (e.g., Industry) and its corresponding value as a tuple in a vector.

The resulting vector, data, contains pairs of sector names and their associated values, effectively transforming raw CSV data into a structured format suitable for further analysis. This function plays a crucial role in preparing the dataset for graphs and PageRank in other parts of the project.

graph.rs

The graph.rs module handles the transformation of GVA (Gross Value Added) data into a weighted directed graph that can later be analyzed using PageRank. The module defines a type alias Graph as a BTreeMap<String, BTreeMap<String, f64>>. This represents a directed graph where each node (a sector name like "Services") is a key, and its values are mapped to other sectors with their corresponding transition probabilities (weights) as f64 values. This graph represents relationships between sectors, with edge weights denoting the likelihood of economic influence or transitions between sectors.

The compute_averages function computes the average contribution of each economic sector to the GVA data. It first initializes a BTreeMap<String, (f64, i32)> to store the sum of values and counts for each sector. The loop iterates through the input data, updating these sums and counts. Then, the function calculates the average contribution for each sector by dividing the total sum by the count (converted to f64) to ensure precision, storing these averages in another BTreeMap<String, f64>.

The build_graph function builds the weighted directed graph using these computed averages. First, it calculates the total GVA across all sectors. For each sector's average, it computes a transition probability by dividing that average by the total GVA. This models how influential a sector's contribution is relative to the entire economy. Then, for each pair of sectors, it assigns a transition probability as the edge weight in the graph. This weight is uniformly distributed across other sectors by dividing the computed transition probability by (number of sectors - 1). The resulting graph captures how sectors influence each other economically, serving as input to the PageRank algorithm for further analysis.

pagerank.rs

The pagerank.rs module implements the PageRank algorithm to compute the influence scores of nodes (economic sectors) in the directed graph built in the previous modules. It uses random walks to approximate these scores by simulating movement through the graph. The function compute_pagerank initializes a random number generator (rng) with a fixed seed (42) to ensure reproducibility and consistency whenever the code is run. The number of random walks is defined by num_walks = 80, and each walk consists of walk_length = 80 steps. A damping factor of 0.1 is introduced to simulate the probability of teleportation—a random jump to any node—mimicking the behavior in traditional PageRank algorithms.

The function initializes a counter, visit_counts, to track how often each node is visited during these random walks. For each starting node (economic sector) in the graph, it simulates num_walks. At each step, the random walker either follows an outgoing edge with a probability

of (1 - damping_factor) or jumps to a random node with a probability of damping_factor. If a node has no outgoing edges, the walker will always jump to another random node. The walker chooses its next move by sampling edges based on their weights, ensuring weighted transitions are respected. After all walks are completed, visit counts are normalized by dividing by the total number of steps taken to compute the final PageRank scores, representing each sector's relative influence in the economic graph. This is done because normalization converts raw visit counts into probabilities, ensuring scores are comparable across nodes and represent the relative influence of each node in the graph. This makes the PageRank scores meaningful and interpretable as a measure of importance.

My logic for this module is based on the description for PageRank given in HW7.
My code closely aligns with the described PageRank process in HW7 in several ways. It simulates 80 random walks of 80 steps each for every vertex in the graph, with the behavior of each step determined by the vertex's outgoing edges. If the current vertex has no outgoing edges, my code correctly jumps to a uniformly random vertex. If outgoing edges exist, it adheres to the probability split: with 90% probability, the next step follows one of the outgoing edges (selected based on weighted probabilities), and with 10%, it jumps to a random vertex. Finally, my code approximates PageRank by counting visits to each vertex and normalizing these counts into probabilities, ensuring the total PageRank sums to 1, as described in the text.

There are 2 tests in this module. The testing module ensures that the computed PageRank scores are non-negative and that their total sums approximately to 1. The tests simulate a small graph with the series/economic sectors and verify these properties.

main.rs

The main.rs module serves as the entry point for the program and orchestrates the sequence of operations required to compute and display the PageRank scores of economic sectors based on GVA (Gross Value Added) data. It begins by loading data from a CSV file using the parse_csv function from the utils module, extracting sectoral GVA information for further analysis. Next, it uses the build_graph function from the graph module to construct a directed graph where sectors are nodes, and edges represent transition probabilities based on the relative GVA contributions.

Once the graph is built, the program invokes the compute_pagerank function from the pagerank.rs module to simulate random walks over the graph and calculate PageRank scores, which measure each sector's influence within the economic graph. The results are sorted in descending order of influence, and the top sectors with the highest scores are displayed in the console. This pipeline integrates data parsing, graph modeling, and PageRank computation to provide insights into sectoral influence over time for countries.

Running the code

Please update the file path on your computer environment in main.rs for data.csv

To run the code, implement the following steps in the terminal:

1. cd project_test
2. cargo run

Output

This is what the output looks like after running the code with cargo run:

```
$ cargo run
   Compiling project v0.1.0 (/opt/app-root/src/project_test)
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 1.44s
     Running `target/debug/project`
Loading CSV data...
Creating Graph...
Running PageRank computations...
Graph vertices: ["Agriculture, hunting, forestry and fishing (% of gross value added)", "Industry (% of gross value added)", "Services (% of gross value added)"]
Results (influence scores of economic sectors over time):
Sector: Agriculture, hunting, forestry and fishing (% of gross value added), Score: 0.3359
Sector: Services (% of gross value added), Score: 0.3323
Sector: Industry (% of gross value added), Score: 0.3318
```

Conclusion

The program's output represents the PageRank scores for different economic sectors, highlighting their relative influence within the constructed economic graph. The graph vertices correspond to sectors identified from the dataset: Agriculture, hunting, forestry and fishing, Industry, and Services, which represent their shares in gross value added (GVA) overall for all countries over time. The computed scores—approximately 0.3359, 0.3323, and 0.3318—indicate the fraction of random walks that terminated at each sector during the simulation. These values, which sum to 1.0, confirm proper normalization. The results suggest that all sectors contribute almost equally to the graph's overall influence, with Agriculture, hunting, forestry and fishing showing a slight edge. This analysis provides insight into the interconnectedness and impact of these sectors over time.