Graph Analysis to Determine Relationships between Employee Attributes and Attrition

Selina Manua

Businesses rely on their employees, and one of the main things companies seek to improve is employee retention. This is because the costs of employee turnover are significant, as retraining and team reorganizations need to be done, which decreases overall productivity. There are many factors that may affect employee attrition, such as age, distance from home, years worked in the company, and many more.

This project utilizes a dataset from Kaggle called IBM HR Analytics Employee Attrition & Performance to build an undirected graph data structure in Rust that describes the relationships between employees based on attributes, such as age, department, daily rate, education, etc. The project aims to find the shortest path between two employees who choose to leave the company. This can help companies understand potential attrition patterns and identify employees who are at risk of attrition. The three main steps in this project are data processing, graph building, and graph analysis.

In this project, the dataset is first read and deserialized into a vector of "Record" structs. Then, the GraphMatrix is created based on the similarity between all pairs of employees. If the similarity score is greater than 40, the matrix entry is set to 0, indicating that the employees are not connected in the graph. The similarity is determined by a similarity score equation that was created, which sums up the difference between several attributes in the employee records. The function computes absolute differences for numerical attributes and assigns points for categorical attributes. The sum of these values then determines the score, which if higher, means that there is lower similarity between the two employees. Then, a graph analysis is performed, where a VecDeque is used to implement the Breadth First Search (BFS) algorithm to navigate through the graph. BFS is used to determine the shortest distance between a given node (an employee) to another node with attrition that is equal to “Yes”.

To run the code, cargo run can be written on the terminal. The program will then request for a node number to search. Then, after the node number has been inputted, the distance between the requested node and the nearest node with attrition “Yes” will be shown. If no node is found, the program will not output anything.

Sample of output when we type “cargo run” on Terminal.

```
Output: Node to search:
Input: 3
Output: Distance between nodes: 1
```