CS101 Project 2

Lakshay Kumar (Entry number: 2023CSB1132)

April 25, 2024

Part 1 Pagerank using a random walk

1 Introduction

In the world of computer science and web search algorithms, pagerank plays a crucial role in determining the importance of web nodes. Pagerank is essentially a measure of the importance of a web node based on the number and quality of links pointing to it. One way to compute pagerank is by using a random walk approach, where we simulate a random surfer moving through a network of web nodes.

2 Pagerank using a random walk with gold coin method

Imagine a group of people, each representing nodes in our network. To find pagerank using a random walk, we'll employ a gold coin method. Here's how it works:

- **Gold Coin Method**:
  - We start with a bag of 10,000,000 gold coins.
  - Each time we meet a person (representing a web node), we give them a gold coin.
  - After distributing all the coins, we calculate pagerank by dividing everyone's coins by 10,000,000 (1 crore).

3 Using Eigenvalue method and matrices

Now, let's delve into the mathematical approach, particularly using matrices and the eigenvalue method to calculate pagerank.

3.1 Linear Algebra Perspective

Consider a network of nodes represented by an $n \times n$ matrix M, where n is the number of web nodes. Each element M_{ij} represents the probability of going from node i to node j.

- We know that matrices can be represented as linear transformations.
- Let there be n eigenvectors for the $n \times n$ matrix. We express each vector as a linear combination of these eigenvectors.
- When we repeatedly apply the matrix M to an initial probability distribution vector, as the power of n becomes very large, the right-hand side converges to the power of only one eigenvalue.
- This convergence leads to the convergence of pagerank.

4 Pagerank Calculation Using Eigenvalue Method

Consider a network of n web nodes represented by an $n \times n$ matrix M, where M_{ij} represents the probability of transitioning from node i to node j. Let's denote the Pagerank vector as PR, where each element PR_i represents the importance of node i.

4.1 Eigenvalue Decomposition

First, we perform an eigenvalue decomposition of M:

$$Mv = \lambda v$$

Where:

- v is an eigenvector.
- λ is the corresponding eigenvalue.

4.2 Linear Combination of Eigenvectors

Let $v_1, v_2, ..., v_n$ be the eigenvectors of M. We express any vector x as a linear combination of these eigenvectors:

$$x = c_1v_1 + c_2v_2 + ... + c_nv_n$$

Where $c_1, c_2, ..., c_n$ are constants.

4.3 Pagerank Convergence

As we repeatedly multiply M to an initial probability distribution vector PR_0 and raise M to a large power, the right-hand side of the equation converges to the power of only one eigenvalue, λ_{max} (the eigenvalue with the largest magnitude). This convergence leads to the convergence of pagerank.

5 Introduction

In our experiment among around 140 people, each person asked some other person a question, resulting in a directed graph. We utilized pagerank, a web search algorithm, to determine the leader of the class based on the interactions among the individuals.

6 Directed Graph Representation

Consider a directed graph $G = (V, E)$, where:

- V represents the set of individuals (vertices).
- E represents the set of interactions between individuals (which exists between A and B if A liked B's answer (edges).

7 Dangling Nodes and teleportation

Dangling nodes are nodes with no outgoing links. When a random surfer arrives at a dangling node, they get stuck. Without a damping factor, this would cause the PageRank algorithm to get stuck as well, as the random surfer would never move to another node.

8 Damping Factor

The damping factor, often denoted as d, is introduced to address the issue of dangling nodes and to ensure that the surfer can eventually leave any node. Typically, it's set to 0.85 as suggested by Sergey Brin and Larry Page in their original paper. The choice of 0.85 is somewhat arbitrary but works well in practice.

- The damping factor d represents the probability that the random surfer will continue clicking on links rather than jumping to a random node.
- The complement $(1 - d)$ represents the probability that the surfer will jump to a random node.

9 Teleportation

With the damping factor, the random surfer either follows links with probability d, or jumps to a random node with probability $(1 - d)$. This is called "teleportation". It prevents the surfer from getting stuck at dangling nodes.

- When the surfer follows a link, they follow it with probability d. This helps to distribute PageRank scores among nodes.
- When the surfer teleports, they land on a random node. This ensures that even if the surfer encounters a dangling node, they can still continue to explore other nodes.

Using the above stated algorithm, we found out the leader using teleportation method using a damping factor of 0.85 and hence, Leader was 2023CSB1091

Part 2 Finding the Missing Links in the Graph

10 Introduction

To find missing links in a graph, we utilized the method of least squares leveraging our knowledge of linear algebra. By constructing an adjacency matrix and performing calculations, we were able to infer missing connections in the graph.

11 Finding Links using Matrix Method

11.1 Adjacency Matrix Construction

We constructed an adjacency matrix A representing the directed graph, where $A_{ij} = 1$ if node i liked node j, and $A_{ij} = 0$ otherwise.

11.2 Least Squares Method

If a row in the adjacency matrix contains a zero element, it indicates a missing link. We removed the corresponding row and column and solved the resulting system of linear equations using the least squares method.

11.3 Solving Linear Equations

We solved the equation $X \cdot \mathbf{x} = Y$ for the coefficients $\mathbf{x}$, where X is the coefficient matrix and Y is the vector representing the missing link.

11.4 Filling Missing Links

The coefficients $\mathbf{x}$ represent the values to be multiplied with each column to fill the missing link. We replaced the missing link with 1 if the value is greater than zero, and 0 otherwise.

11.5 Bidirectional Links

We determined if the link was bidirectional, unidirectional pointing to whom, or no link based on the computed coefficients.

To find the missing links in the graph, we employed the method of least squares using our knowledge of linear algebra. We first constructed an adjacency matrix where we added 1 if node A liked node B, as this was a directed graph.

Adjacency Matrix

Let's denote the adjacency matrix as A, where $A_{ij} = 1$ if node i liked node j, and $A_{ij} = 0$ otherwise.

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Suppose we have a missing link from node 3 to node 4, represented by $A_{34} = 0$.

Least Squares Method

If we encounter a 0 in any row, we remove that row and column. With the remaining matrix, we solve the equations using the least squares method to find the coefficients x_1 to x_n to be multiplied with that column to fill that cell.

$$Ax = b$$

Where x is the coefficient vector and b is the target vector. Solving this equation using least squares, we get the coefficients.

Example: Filling Missing Link A_{34}

Suppose we remove row 3 and column 4 from matrix A:

$$A' = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

We want to find the coefficients x for the equation $A'x = b$ where $b = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$.

$$A'x = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Solving this system of equations, we get $x \approx \begin{bmatrix} -0.5 \\ 0.5 \\ 0.5 \end{bmatrix}$.

$$S = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

Now, $x^T S$ is 0.5

Since the value range is from -ve to positive, we replace the cell with 1 if the value is greater than zero and zero otherwise.

$$A_{34} = \begin{cases} 1 & \text{if } x_3 >= 0 \\ 0 & \text{otherwise} \end{cases}$$

In this case, $x_3 = 0.5 > 0$, so $A_{34} = 1$.

Resultant Adjacency Matrix

After filling the missing link A_{34}, the updated adjacency matrix becomes:

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Analysis of Links

Now, we check if the link is bidirectional, unidirectional pointing to whom, or no link.

- Bidirectional link: Both A_{ij} and A_{ji} are 1.
- Unidirectional link pointing to node j: $A_{ij} = 1$ and $A_{ji} = 0$.
- No link: Both A_{ij} and A_{ji} are 0.

In this example, we can see that $A_{34} = 1$ and $A_{43} = 0$, indicating atleast a unidirectional link from 3 to 4

In a similar way, we find the missing zeroes on the adjacency matrix of the impression network. And then, we will decide that if the persons talked to each other, would they have liked each other or not or 1 would have liked the other.

12 Conclusion

By employing the method of least squares and linear algebra, we were able to find missing links in a graph and infer bidirectional or unidirectional connections. This approach provides a systematic way to analyze and fill missing links in a graph. In a similar way as demonstrated above, we find the missing zeroes on the adjacency matrix of the impression network. And then, we will decide that if the persons talked to each other, would they have liked each other or not or 1 would have liked the other.

Part 3 Finding a new question from the impression Network

13 Introduction

Consider a directed graph representing connections between students from the Computer Science (CS) and Mathematics and Computing (MnC) departments. We aim to analyze the variance and stability of this graph after removing students who have more friends in the other department than in their own.

14 Variance Calculation

To calculate the variance of the nodeRank scores in the original graph, we use the following formula:

$$\text{Variance} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}$$

where x_i are the nodeRank scores of individual students, $\bar{x}$ is the mean nodeRank score, and n is the total number of students.

15 Stability Calculation

Stability measures the change in variance before and after the removal of nodes. We use the following formula to calculate stability:

$$\text{Stability (\% change)} = \left(\frac{\text{New Variance} - \text{Old Variance}}{\text{Old Variance}}\right) \times 100$$

16 Analysis

After removing students with more friends in the other department, we recalculate the variance and stability of the graph.

16.1 Variance Analysis

We find the variance of the nodeRank scores in the original graph to be $V_{\text{old}} = X$. After removing certain nodes, the new variance is $V_{\text{new}} = Y$.

16.2 Stability Analysis

The stability of the graph is calculated by comparing the old and new variances:

$$\text{Stability (\% change)} = \left(\frac{Y - X}{X}\right) \times 100$$

17 Results

17.1 Variance

The original variance $V_{\text{old}} = X$, and the new variance after node removal is $V_{\text{new}} = Y$.

17.2 Stability

The percentage change in stability is calculated as:

$$\text{Stability (\% change)} = \left(\frac{Y - X}{X}\right) \times 100$$

18 Conclusion

The analysis reveals interesting observations:

- The stability of the CS subgraph increased drastically by 76.94%, while that of the MnC subgraph increased by 0.5%. This indicates that the structural changes induced by removing nodes had a significant impact on the CS subgraph's stability.
- The CS subgraph initially had almost 100 students, while the MnC subgraph had around 40 students. This difference in the number of students likely influenced the variance and stability results.
- Among the students in the CS subgraph, only 15 out of nearly 100 had more friends in the MnC department than in their own. Similarly, only 3 out of around 40 students in the MnC subgraph had more friends in the CS department. The relatively small number of such students may have contributed to the modest increase in stability observed in the MnC subgraph.

These observations suggest that the distribution of connections within the graph, along with the size of the subgraphs, plays a crucial role in determining the impact of node removal on variance and stability.