Massive Data Analysis

Instructor: Dr.Gholampour

Fall 2024

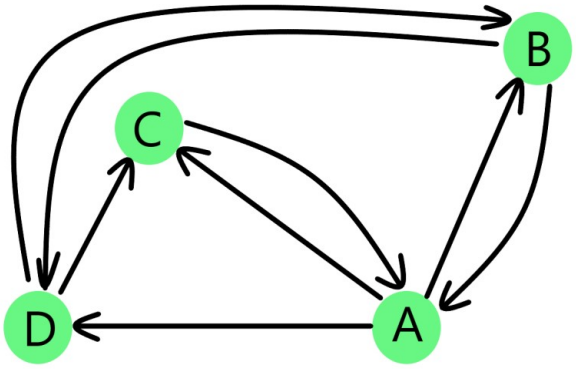
HW 4

Arman Yazdani - 400102255

Contents

1 PageRank I

1.1 Adjacency & Probablity Matrix



The matrices are given by:

$$A_{adj} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix} \quad A_{pr} = \begin{bmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix}$$

1.2 TrustRank & Spam mass

Since only page B is a trusted page, the initial trust vector T_0 is:

$$T_0 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Using the TrustRank formula:

$$T = \alpha P^T T + (1 - \alpha) T_0$$

let $\alpha = 0.85$:

$$P^T = \begin{bmatrix} 0 & \frac{1}{2} & 1 & 0 \\ \frac{1}{3} & 0 & 0 & \frac{1}{2} \\ \frac{1}{3} & 0 & 0 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{2} & 0 & 0 \end{bmatrix}$$

Multiplying:

$$P^T T_0 = \begin{bmatrix} \frac{1}{2} \cdot 1 + 0 \cdot 0 + 1 \cdot 0 + 0 \cdot 0 \\ 0 \cdot 1 + 0 \cdot 0 + 0 \cdot 0 + \frac{1}{2} \cdot 0 \\ 0 \cdot 1 + 0 \cdot 0 + 0 \cdot 0 + \frac{1}{2} \cdot 0 \\ 0 \cdot 1 + \frac{1}{2} \cdot 0 + 0 \cdot 0 + 0 \cdot 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Applying the damping factor:

$$T_1 = 0.85 \times \begin{bmatrix} \frac{1}{2} \\ 0 \\ 0 \\ 0 \end{bmatrix} + 0.15 \times \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$T_1 = \begin{bmatrix} 0.85 \times \frac{1}{2} + 0.15 \times 0 \\ 0.85 \times 0 + 0.15 \times 1 \\ 0.85 \times 0 + 0.15 \times 0 \\ 0.85 \times 0 + 0.15 \times 0 \end{bmatrix} = \begin{bmatrix} 0.425 \\ 0.15 \\ 0 \\ 0 \end{bmatrix}$$

Spam Mass of a page i is defined as:

$$\text{SpamMass}(i) = 1 - \frac{\text{TrustRank}(i)}{\text{PageRank}(i)}$$

Converged values for TrustRank, PageRank, and Spam Mass for each page are:

Page	TrustRank	PageRank	Spam Mass
A	0.2849	0.3246	0.1222
B	0.3235	0.2251	-0.4366
C	0.1735	0.2251	0.2296
D	0.2182	0.2251	0.0309

1.3 Hubs&Authorities

After applying the *HITS (Hyperlink-Induced Topic Search) algorithm*, the computed scores are:

$$S_{authorities} = \begin{cases} A = 0.1754 \\ B = 0.6032 \\ C = 0.6032 \\ D = 0.4915 \end{cases}$$

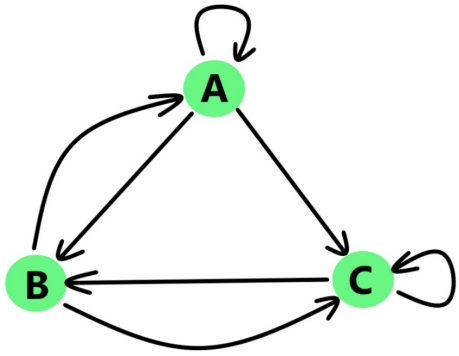
Hub Scores:

$$S_{hubs} = \begin{cases} A = 0.7739 \\ B = 0.3039 \\ C = 0.0799 \\ D = 0.5499 \end{cases}$$

This shows that:

- Nodes B and C are the best **authorities** (most referenced nodes).
- Node A is the strongest **hub** (links to many good authorities).

2 PageRank II



The given adjacency matrix A_{adj} and transition probability matrix P are:

$$A_{\text{adj}} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}, P = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

The PageRank update rule is:

$$r^{(k+1)} = \beta P r^{(k)} + (1-\beta)\frac{1}{N}\mathbf{1}$$

where:

- $\beta = 0.8$
- $N = 3$ (number of nodes)
- $r^{(0)} = \frac{1}{3}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ (initial uniform PageRank vector)

Now let's compute the PageRank vector after one iteration.

Initial values are:

$$r^{(0)} = \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{bmatrix}$$

Computing:

$$r^{(1)} = 0.8Pr^{(0)} + (1 - 0.8)\frac{1}{3}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

First, let's compute $Pr^{(0)}$:

$$Pr^{(0)} = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{bmatrix} = \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{bmatrix}$$

Now multiply by 0.8:

$$0.8Pr^{(0)} = \begin{bmatrix} 0.8 \times \frac{1}{3} \\ 0.8 \times \frac{1}{3} \\ 0.8 \times \frac{1}{3} \end{bmatrix} = \begin{bmatrix} \frac{4}{15} \\ \frac{4}{15} \\ \frac{4}{15} \end{bmatrix}$$

Adding the teleportation factor $(1 - 0.8)\frac{1}{3} = 0.2 \times \frac{1}{3} = \frac{1}{15}$ to each element:

$$r^{(1)} = \begin{bmatrix} \frac{4}{15} + \frac{1}{15} \\ \frac{4}{15} + \frac{1}{15} \\ \frac{4}{15} + \frac{1}{15} \end{bmatrix} = \begin{bmatrix} \frac{5}{15} \\ \frac{5}{15} \\ \frac{5}{15} \end{bmatrix} = \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{bmatrix}$$

Since $r^{(1)} = r^{(0)}$, the PageRank vector has already converged after one iteration.

Thus, the final *converged PageRank vector* is:

$$r = \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{bmatrix}$$

Each node has an equal PageRank score of $\frac{1}{3}$, which makes sense given the almost symmetric link structure.

3 Stream data

3.1 Counting distinct items

Here are steps to Implement Flajolet-Martin Algorithm for counting unique students for each university:

1. Sampling
   - Randomly sample 5% of the tuples from the stream. Ensure that the sampling is uniform and unbiased.
   - For each sampled tuple $(university, courseId, studentId, grade)$, extract the $university$ and $studentId$.
2. Apply Flajolet-Martin Algorithm per University
   - For each university in the sampled data, maintain a separate Flajolet-Martin bitmap (or a set of bitmaps if using multiple hash functions for better accuracy).
   - For each $studentId$ in the sampled data, hash it using a hash function and update the corresponding university's bitmap(s) based on the position of the least significant 1-bit.
3. Estimate Distinct Students per University
   - For each university, use the Flajolet-Martin algorithm to estimate the number of distinct $studentId$s in the sampled data.
   - Let R be the position of the rightmost 1-bit in the bitmap. The estimated number of distinct students in the sampled data is approximately $\frac{2^R}{\phi}$, where $\phi \approx 0.77351$ is a correction factor.
4. Scale the Estimate
   - Since the sampled data represents 5% of the total data, scale the estimate by a factor of 20 (i.e., $\frac{1}{0.05}$) to estimate the total number of distinct students per university.
5. Example: Suppose:
   - For a university U, the Flajolet-Martin algorithm estimates 100 distinct students in the sampled data.
   - Since the sample is 5% of the total data, the estimated total number of distinct students in university U is:

$$\text{Estimated total students} = 100 \times \left(\frac{1}{0.05}\right) = 2000$$

6. Pseudocode

Algorithm 1 Flajolet-Martin Algorithm for Estimating Students per University

Initialize a dictionary to store bitmaps for each university: $university\_bitmaps = \{\}$
for each $(university, \_, studentId, \_)$ in sampled data **do**
    if $university \notin university\_bitmaps$ **then**
        $university\_bitmaps[university] = \{\}$
    end if
    $hash\_value = \text{hash\_function}(studentId)$
    $lsb = \text{least\_significant\_bit}(hash\_value)$
    $university\_bitmaps[university].add(lsb)$
end for
for each $university, bitmap$ in $university\_bitmaps$ **do**
    $sampled\_estimate = \frac{2^{\max(bitmap)}}{0.77351}$
    $total\_estimate = sampled\_estimate \times 20$
    Print "University $university$: Estimated $total\_estimate$ students"
end for

3.2 Filtering Data streams I

To estimate the percentage of students with GPA ≥ 3.5, we use a sampling-based approach combined with Bloom filtering. The key steps are:

1. Randomly sample 5% of the tuples from the stream.
2. Use a Bloom filter to track unique students across universities.
3. Compute GPAs for the sampled students.
4. Estimate the percentage of students with GPA ≥ 3.5 based on the sample.

Let's break down the process in more detail:

1. Starting with the initial setup:
    - Create a Bloom filter with an appropriate size and number of hash functions.
    - Create a hash table or dictionary to store `(university, studentId)` as keys and their corresponding grades as values.
2. Continiueing with processing the Stream:
   For each tuple `(university, courseId, studentId, grade)` in the stream:
    (a) Randomly sample the tuple with a probability of 0.05.

(b) If sampled, hash `(university, studentId)` and check if it exists in the Bloom filter:

- If it doesn't exist, add it to the Bloom filter and initialize a list of grades for this student.
- If it exists, append the grade to the student's list of grades.

3. For Computing GPAs we have:
   For each unique student in the Bloom filter:
   (a) Compute their GPA by averaging their grades.
   (b) Check if the GPA ≥ 3.5.
4. We'll estimate the Percentages
   - Let N be the total number of unique students in the sample.
   - Let M be the number of unique students in the sample with GPA ≥ 3.5.
   - The estimated percentage of students with GPA ≥ 3.5 is:
   $$\text{Percentage} = \left(\frac{M}{N}\right) \times 100$$
5. But how to handle False Positives?
   Bloom filters can produce false positives (i.e., incorrectly claiming a student is already in the set). To mitigate this:
   - Use a large enough Bloom filter to reduce the false positive rate.
   - If a false positive occurs, it may lead to slight inaccuracies in the GPA computation, but the overall estimate should still be reasonable.

3.3 Filtering Data streams II

The goal is to estimate the percentage of courses where at least half of the students received an 'A' grade.
Our approach involves using two Bloom filters: one to track courses with 'A' grades and another to sample the data.

Let's break down the process in more detail:

1. **Track Courses with 'A' Grades:**
   - For each tuple (university, courseId, studentId, grade):
   - If grade = A, add (university, courseId) to a Bloom filter.

2. **Sample the Data:**
   - Randomly sample 5% of the tuples.
   - For each sampled tuple, if grade = A, add (university, courseId) to a second Bloom filter (sampled Bloom filter).
3. **Estimate the Number of Courses with at Least 50% 'A' Grades:**
   - For each course (university, courseId) in the sampled Bloom filter:
   - Use the full Bloom filter to estimate the total number of students who received an 'A' grade in that course.
   - If the estimated number of 'A' grades is at least 50% of the total students in the course, count it as a course meeting the condition.
4. **Calculate the Percentage:**
   - Divide the number of courses meeting the condition by the total number of unique courses in the sampled Bloom filter.
   - Scale this percentage to the entire dataset.

3.3.1 Example

Consider the following stream of tuples:

$$[(\text{U1}, \text{C1}, \text{S1}, \text{A}), (\text{U1}, \text{C1}, \text{S2}, \text{B}), (\text{U1}, \text{C1}, \text{S3}, \text{A}),$$
$$(\text{U1}, \text{C2}, \text{S1}, \text{A}), (\text{U1}, \text{C2}, \text{S2}, \text{A}), (\text{U2}, \text{C1}, \text{S1}, \text{B}),$$
$$(\text{U2}, \text{C1}, \text{S2}, \text{A}), (\text{U2}, \text{C2}, \text{S1}, \text{A}), (\text{U2}, \text{C2}, \text{S2}, \text{B}), \dots]$$

Let's walk through the steps with this example:

1. Track Courses with 'A' Grades
   Add (university, courseId) to the Bloom filter for tuples with grade = A:
   $$\text{Bloom Filter: } \{(\text{U1}, \text{C1}), (\text{U1}, \text{C2}), (\text{U2}, \text{C1}), (\text{U2}, \text{C2})\}.$$
2. Sample the Data
   Assume we sample 5% of the tuples (e.g., 1 tuple). Suppose the sampled tuple is:
   $$(\text{U1}, \text{C1}, \text{S1}, \text{A}).$$
   Add (U1, C1) to the sampled Bloom filter.
3. Estimate the Number of Courses with at Least 50% 'A' Grades
   For the sampled course (U1, C1):

- Total students in U1, C1: 3.
- Estimated 'A' grades (from the full Bloom filter): 2.
- Since $2/3 \geq 0.5$, this course meets the condition.

4. Calculate the Percentage
   - Number of courses meeting the condition: 1.
   - Total unique courses in the sampled Bloom filter: 1.
   - Estimated percentage: 100%.