# Information Retrieval

## Search Engine Bandits

#### Sara Asaad – 207674318

#### Ward Zidani – 318531944

#### Safwan Halabi – 213784127

[GitHub Link](https://github.com/Safwan-Halabi/Information-Retrieval.git): - <https://github.com/Safwan-Halabi/Information-Retrieval.git>

**Question 1:**

**Link Analysis Diagram:**

**Step 1: Defining Documents and Categories**

Assume we have five documents D1, D2, D3, D4, and D5 represented by one-dimensional values:

* D1=0
* D2=1
* D3=4
* D4=5
* D5=2

Assume there are two categories:

C1 includes D1 and D2.

C2 includes D3 and D4.

**Step 2: Determining the Centroids of the Categories**

The centroid of C1 is (0+1)/2 = 0.5.

The centroid of C2 is (4+5)/2 = 4.5.

**Step 3: New Learning Sample**

Our new learning sample is D5=2. We want to determine which category D5 belongs to.

**Step 4: Calculating Distances Using Rocchio**

Calculate the distance of D5 from the centroids:

Distance from C1: |0.5-2| = 1.5

Distance from C2: |4.5-2| = 2.5

According to the Rocchio algorithm, D5 is assigned to category C1 because the distance is smaller.

**Step 5: Analysis Using Link Analysis**

Assume the links between documents are built based on their proximity. We will construct a graph with the links as shown in the diagram:

D1 links to D2.

D2 links to D3.

D3 links to D4.

D5 links to D3.

**Step 6: PageRank Analysis**

**Perform PageRank analysis:**

Initial PageRank values of 1 for each node.

In each iteration, the PageRank values of each node are divided among the linked nodes.

After several iterations, we get the weighted PageRank values according to the links.

Now, calculate the PageRank values in a simple manner and see how it affects the belonging of D5:

Step 1: Defining Initial PageRank Values

Step 2: PageRank Iterations:

Assume a damping factor of 0.85 used to reduce the influence of certain links and prevent infinite loops. The values will be updated in each iteration according to the following formula:

A white background with text and numbers

Description automatically generated with medium confidence

**Definitions:**

d is the damping factor, let's assume d=0.85

N is the number of nodes in the graph, in our case N=5

I(p) is the set of nodes linked to node p

D(j) is the number of outbound links from node j

**Perform the calculation in steps:**

**Initial Calculation:**

Define initial PageRank values of 1 for each node:

* R(D1)=1
* R(D2)=1
* R(D3)=1
* R(D4)=1
* R(D5)=1

**First Iteration:**

**Second Iteration:**

**Third Iteration:**

**Fourth Iteration:**

Values are stable, we stop the iterations. In PageRank analysis, document D4 gets the highest PageRank value.

**Step 7: Analyzing the Result**

After several iterations, it is seen that R(D4) is the highest. Since D4 belongs to category C2, we can conclude that D5 should belong to category C2 due to its programmatic links with D3 and D4.

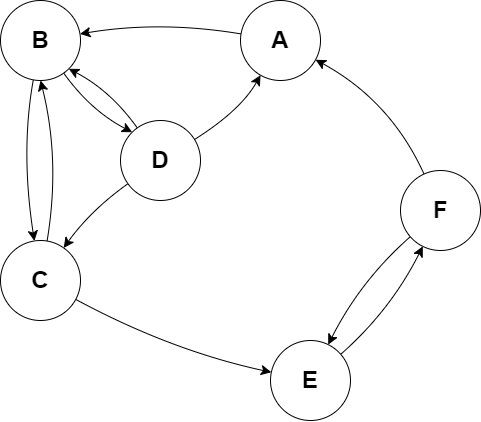
This one-dimensional case demonstrates how the Rocchio classifier can make mistakes when there are outliers. Combining Link Analysis provides additional information on the links between items, which can improve classification and correct such errors.

**Conclusion:**

This analysis shows that using the Rocchio algorithm alone, D5 would belong to category C1, but using Link Analysis, we can see that D5 should belong to category C2. This demonstrates a case where Rocchio assigns the wrong tag to a learning sample.

The resulting graph shows that D5 belongs to category C2 instead of C1, as incorrectly classified by the Rocchio algorithm.

**Question 2.1:**



|  |  |  |
| --- | --- | --- |
|  | Score Authority (in) | Score Hub (out) |
| A | 2 | 1 |
| B | 3 | 2 |
| C | 2 | 2 |
| D | 1 | 3 |
| E | 2 | 1 |
| F | 1 | 2 |

**Question 2.2:**

A diagram of a diagram

Description automatically generated

|  |  |  |
| --- | --- | --- |
|  | Score Authority (in) | Score Hub (out) |
| A | 2 | 3 |
| B | 2 | 1 |
| C | 2 | 2 |

**Normalization:**

Authority sum:

Hub sum:

**Dividing by the sum to get the normalized Authority score:**

Node A:

Node B:

Node C:

**Dividing by the sum to get the normalized Hub score:**

Node A:

Node B:

Node C:

**Full table after Normalization:**

|  |  |  |
| --- | --- | --- |
|  | Normalized Score Authority | Normalized Score Hub |
| A | 0.33 | 0.5 |
| B | 0.33 | 0.167 |
| C | 0.33 | 0.33 |

**Question 3.1:**

Here are some interesting technologies used in the project:

1. **Requests library** – used to send HTTP requests in code.
2. **BeautifulSoup library** – used to process the HTML pages returned from the requests.
3. **Pandas library** – used to save results in a DataFrame to later be exported to a csv file.
4. **NLTK library** – used to process text, specifically to remove stop-words from the texts returned.
5. **RE library** – used to process text and look for specific patterns in the strings.

**Question 3.2:**

For the **first Query** the runtime is about **1.5 minutes.**

For the **second Query** the runtime for a single execution is about **1.5 ~ 2 minutes.**

The code is **heavily dependent** on the request **bypassing LinkedIn’s bot detection** and the time it took to send various requests in both queries.

The **processing time** is **negligible**, it only takes a **fraction of a second** to extract the text, process it, save it in a DataFrame and export it to a csv.

If provided with a fast internet connection and run closer to a LinkedIn’s server the time could **potentially go down to less than a minute.**

**Question 3.3:**

**Query 1**: The pages returned are mainly **48 Independent pages**, we simply send 48 requests that aren’t connected at all, therefore we don’t have any **hubs** or **authorities**.

**Query 2**: The pages returned are mainly **1 Hub page** which has 60 links to the first 60 job listings.

Each job listing is a stand-alone page and isn’t associated with the rest of the job listings – These are the **Authority pages** of which there are **60** in total.

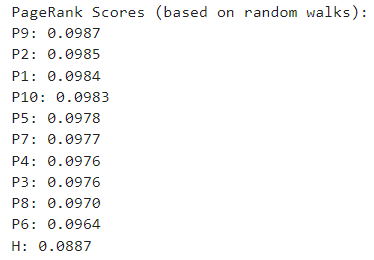
**Question 3.4:**

Given that our system has 1 **Hub** and several **Authorities** (which no connection between authorities) we’ll have to use a different variation of the PageRank formula – one that incorporates random walks:

Each Authority page has a 10% chance to go to a different page altogether.

Here are the results we got:

Run 1:



Run 2:

A screenshot of a computer code

Description automatically generated

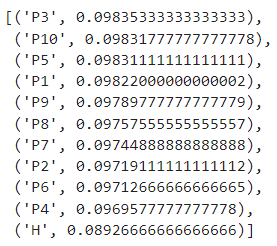
Run 3:

A screenshot of a computer

Description automatically generated

We see that a common trend we’re having is the **Hub** page having the lowest PageRank – which is logical given that **no pages have explicit links going into** the **Hub** page; moreover we see that the order of the Authority pages doesn’t matter and is constantly changing – which can be explained by the fact that each **Authority** page has **exactly 1 explicit link going into it** and they all **have a 10% chance to go out and branch into another random node** in the graph – meaning that the order of the nodes is reliant on chance alone.

We’ll take the average of all the rankings of runs 1-3, we then get the following (final) ranking:



**Question 3.5:**

We asked three users to rate our PageRanks and provide us with some feedback in order to perform query optimization, here are the results we found:

**Unique keywords (for each job listing):**

|  |  |
| --- | --- |
| Job Listing | Unique Keywords |
| P1 | Deep learning |
| P2 | retention, predictive models, data visualization, Hadoop, Spark, SaaS, NLP, sci-kit learn |
| P3 | Forecasting, SQL, gradient boosting, MLOPs, reinforcement, Apache Airflow |
| P4 | Biomedical, deep learning |
| P5 | Research, publishing, paper, feature engineering, media |
| P6 | Formatting |
| P7 | Research, publishing, paper, feature engineering, media |
| P8 | ML, generative, AWS, GCP, RAG |
| P9 | pipelines, distributed programming, AWS |
| P10 | AWS, Big Data, CNN, RNN, LSTM, Transformer, sci-kit learn, Multimodal, transfer |

User 1’s Ranking:

|  |  |
| --- | --- |
| Rank | Page |
| 1 – Relevant | P3 |
| 2 – Relevant | P8 |
| 3 – Relevant | P9 |
| 4 – Relevant | P10 |
| 5 – Relevant | P2 |
| 6 – Non Relevant | P4 |
| 7 – Non Relevant | P6 |
| 8 – Non Relevant | P1 |
| 9 – Non Relevant | P7 |
| 10 – Non Relevant | P5 |

**Relevant keywords (keywords for relevant documents):**

Forecasting, SQL, gradient boosting, MLOPs, reinforcement, Apache Airflow, ML, generative, AWS, GCP, RAG, pipelines, distributed programming, Big Data, CNN, RNN, LSTM, Transformer, sci-kit learn, Multimodal, transfer, retention, predictive models, data visualization, Hadoop, Spark, SaaS, NLP.

**Non-Relevant keywords (keywords for non-relevant documents):**

Research, publishing, paper, feature engineering, media, formatting, biomedical.

User 2’s Ranking:

|  |  |
| --- | --- |
| Rank | Page |
| 1 – Relevant | P6 |
| 2 – Relevant | P7 |
| 3 – Relevant | P5 |
| 4 – Relevant | P1 |
| 5 – Relevant | P3 |
| 6 – Non Relevant | P10 |
| 7 – Non Relevant | P8 |
| 8 – Non Relevant | P9 |
| 9 – Non Relevant | P2 |
| 10 – Non Relevant | P4 |

**Relevant keywords (keywords for relevant documents):**

Formatting, research, publishing, paper, feature engineering, media, Deep learning, Forecasting, SQL, gradient boosting, MLOPs, reinforcement, Apache Airflow.

**Non-Relevant keywords (keywords for non-relevant documents):**

ML, generative, AWS, GCP, RAG, pipelines, distributed programming, AWS, Big Data, CNN, RNN, LSTM, Transformer, sci-kit learn, Multimodal, transfer, retention, predictive models, data visualization, Hadoop, Spark, SaaS, NLP, Biomedical, deep learning.

User 3’s Ranking:

|  |  |
| --- | --- |
| Rank | Page |
| 1 – Relevant | P1 |
| 2 – Relevant | P9 |
| 3 – Relevant | P4 |
| 4 – Relevant | P8 |
| 5 – Relevant | P5 |
| 6 – Non Relevant | P7 |
| 7 – Non Relevant | P3 |
| 8 – Non Relevant | P6 |
| 9 – Non Relevant | P10 |
| 10 – Non Relevant | P2 |

**Relevant keywords (keywords for relevant documents):**

Deep learning, pipelines, distributed programming, AWS, Biomedical, deep learning, ML, generative, GCP, RAG, Research, publishing, paper, feature engineering, media.

**Non-Relevant keywords (keywords for non-relevant documents):**

Research, publishing, paper, feature engineering, media, Forecasting, SQL, gradient boosting, MLOPs, reinforcement, Apache Airflow, Formatting, AWS, Big Data, CNN, RNN, LSTM, Transformer, sci-kit learn, Multimodal, transfer, retention, predictive models, data visualization, Hadoop, Spark, SaaS, NLP.

--------------------------------------------------------------------------------------------------------------

\*\* **The keywords highlighted in gray are unique keywords for relevant and non-relevant documents – therefore their use in query optimization is not advised**. \*\*

We can see that query optimization is a recommended option for users 1 and 2, user 3 however is a special case where a large percentage of the relevant documents he/she submitted share unique keywords with the non-relevant documents he/she submitted.

**Query optimization:**

**User 1:** MLOPs, generative, Big Data, data visualization, SaaS, NLP.

**User 2:** Research, feature engineering, SQL, Media, Forecasting.

**User 3:** Biomedical, deep learning, pipelines.