University of Information Technology & Sciences

Maddha Naya Nagar, Vatara, Baridhara, Dhaka-1212

Department of Computer Science and Engineering

Course Title: Data Mining and Warehouse Lab
Course Code: CSE 426

Final Project Report

Submitted by

Sohana Afrin

Batch: 50

Section: 8A

ID: 2125051013

Submitted To:

Mrinmoy Biswas Akash

Lecturer & Course Coordinator

University Of Information Technology and Sciences

DATE: July 6, 2025.

Project 2

Discovering Heart Disease Patterns through Association Rule Mining

Project Overview

This project explores the patterns inside the heart disease dataset using Association Rules Mining. Using the Apriori algorithm, it uncovers frequent combinations of health conditions that are strongly linked with the risk of heart disease.

Objectives

- Crawl webpages from a specific domain (e.g., learning platforms).
- Tokenize content and build an inverted index for search functionality.
- Construct a web graph from hyperlinks found during crawling.
- Calculate PageRank scores to assess page importance.
- Rank and display search results based on relevance and authority.

Purpose and Goals

- Transform the dataset into a suitable format for mining.
- Apply the Apriori algorithm with a minimum support threshold of 0.3.
- Extract the top 10 rules with high confidence (≥ 0.7) and lift.
- Interpret the most significant rule in a medical context.

Technical Execution

- Data was encoded using one-hot encoding.
- `mlxtend` library's Apriori and `association_rules()` functions were utilized.
- Rules were sorted by confidence and lift to extract the most relevant associations.
- A significant rule was selected and explained regarding real-world implications.

Code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from wordcloud import WordCloud
```

```
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import plotly.figure_factory as ff
from mlxtend.frequent_patterns import apriori, association_rules
from google.colab import drive
drive.mount('/content/drive')
from google.colab import drive
drive.mount('/content/drive')
df = pd.read_csv('/content/drive/MyDrive/8th_semester/Data_mining/Final
project/project_2/heart_disease.csv')
df
# Load pandas library
import pandas as pd

# Apply one-hot encoding to all columns in the DataFrame
encoded_df = pd.get_dummies(data=df, columns=df.columns)

# Display the encoded DataFrame
encoded_df

# Generate frequent itemsets using the Apriori algorithm
frequent_sets = apriori(encoded_df, min_support=0.3, use_colnames=True)

# Output the frequent itemsets
print(frequent_sets)

for element in frequent_sets.iloc[90]:
    print(element)

# Count the number of frequent itemsets
total_itemsets = len(frequent_sets)

# Generate association rules based on confidence threshold
generated_rules = association_rules(frequent_sets, metric="confidence",
min_threshold=0.7, num_itemsets=total_itemsets)

# Display the resulting rules
generated_rules

# Sort the rules by confidence and lift in descending order, then show the
top 10
top_rules = generated_rules.sort_values(['confidence', 'lift'],
ascending=[False, False])
top_rules.head(10)
```

```python
# Step 3: Order the association rules by confidence and lift in descending order, then take the top 10
top_10 = generated_rules.sort_values(['confidence', 'lift'],
ascending=False).head(10)

# Step 4: Show confidence and lift values of the top 10 rules
print(top_10[['confidence', 'lift']])
```

Output

The top 10 Association Rules Output is given below:

The result below shows the top 10 rules sorted by confidence and lift. Each rule includes support, confidence, and lift values that quantify its strength and significance.

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	representativity	leverage	conviction	zhangs_metric	jaccard	certainty	kulczynski
8	(trestbps_low)	(fbs_0)	0.333333	0.851485	0.313531	0.940594	1.104651	1.0	0.029703	2.500000	0.142105	0.359848	0.600000	0.654406
54	(ca_0, thal_2)	(fbs_0)	0.376238	0.851485	0.343234	0.912281	1.071399	1.0	0.022874	1.693069	0.106838	0.388060	0.409357	0.657691
92	(ca_0, target_1, thal_2)	(fbs_0)	0.336634	0.851485	0.303630	0.901961	1.059280	1.0	0.016992	1.514851	0.084361	0.343284	0.339869	0.629275
76	(ca_0, thal_2)	(target_1)	0.376238	0.544554	0.336634	0.894737	1.643062	1.0	0.131752	4.326733	0.627451	0.576271	0.768879	0.756459
45	(ca_0, exang_0)	(fbs_0)	0.432343	0.851485	0.386139	0.893130	1.048908	1.0	0.018005	1.389675	0.082141	0.430147	0.280407	0.673309
62	(thal_2, slope_2)	(exang_0)	0.336634	0.673267	0.300330	0.892157	1.325115	1.0	0.073686	3.029703	0.369854	0.423256	0.669935	0.669118
15	(ca_0)	(fbs_0)	0.577558	0.851485	0.511551	0.885714	1.040199	1.0	0.019769	1.299505	0.091482	0.557554	0.230476	0.743245
56	(ca_0, target_1)	(fbs_0)	0.429043	0.851485	0.379538	0.884615	1.038909	1.0	0.014214	1.287129	0.065594	0.421245	0.223077	0.665176
94	(ca_0, thal_2, fbs_0)	(target_1)	0.343234	0.544554	0.303630	0.884615	1.624476	1.0	0.116721	3.947195	0.585318	0.519774	0.746656	0.721096
16	(thal_2)	(fbs_0)	0.547855	0.851485	0.481848	0.879518	1.032922	1.0	0.015358	1.232673	0.070493	0.525180	0.188755	0.722705

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	representativity	leverage	conviction	zhangs_metric	jaccard	certainty	kulczynski
0	(cp_0)	(sex_1)	0.471947	0.683168	0.343234	0.727273	1.064559	1.0	0.020815	1.161716	0.114844	0.422764	0.139205	0.614844
1	(sex_1)	(fbs_0)	0.683168	0.851485	0.574257	0.840580	0.987192	1.0	-0.007450	0.931593	-0.039337	0.597938	-0.073430	0.757499
2	(restecg_0)	(sex_1)	0.485149	0.683168	0.339934	0.700680	1.025633	1.0	0.008496	1.058506	0.048544	0.410359	0.055272	0.599132
3	(thal_3)	(sex_1)	0.386139	0.683168	0.336634	0.871795	1.276106	1.0	0.072836	2.471287	0.352467	0.459459	0.595353	0.682274
4	(target_0)	(sex_1)	0.455446	0.683168	0.376238	0.826087	1.209200	1.0	0.065092	1.821782	0.317703	0.493506	0.451087	0.688406
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
94	(ca_0, thal_2, fbs_0)	(target_1)	0.343234	0.544554	0.303630	0.884615	1.624476	1.0	0.116721	3.947195	0.585318	0.519774	0.746656	0.721096
95	(target_1, thal_2, fbs_0)	(ca_0)	0.376238	0.577558	0.303630	0.807018	1.397293	1.0	0.086331	2.189019	0.455832	0.467005	0.543174	0.666366
96	(ca_0, target_1)	(thal_2, fbs_0)	0.429043	0.481848	0.303630	0.707692	1.468704	1.0	0.096897	1.772625	0.558934	0.500000	0.435865	0.668915
97	(ca_0, thal_2)	(target_1, fbs_0)	0.376238	0.468647	0.303630	0.807018	1.722016	1.0	0.127308	2.753375	0.672188	0.560976	0.636809	0.727452
98	(target_1, thal_2)	(ca_0, fbs_0)	0.429043	0.511551	0.303630	0.707692	1.383424	1.0	0.084153	1.671009	0.485423	0.476684	0.401559	0.650620

99 rows × 14 columns

After using sort rules by lift in descending order:

	confidence	lift
8	0.940594	1.104651
54	0.912281	1.071399
92	0.901961	1.059280
76	0.894737	1.643062
45	0.893130	1.048908
62	0.892157	1.325115
15	0.885714	1.040199
56	0.884615	1.038909
94	0.884615	1.624476
16	0.879518	1.032922

Final Insights

Association Rule Mining proved effective in revealing meaningful patterns within the heart disease dataset. These insights can aid healthcare professionals in identifying high-risk patients. Future improvements may include numeric value binning or using more advanced rule-mining algorithms.

Conclusion

The application of Association Rule Mining on the heart disease dataset effectively uncovered significant patterns and correlations between patient attributes and disease outcomes. The derived rules, particularly those with high confidence and lift, provide valuable insights into risk factors associated with heart disease. Such patterns can assist healthcare professionals in early diagnosis and preventive care. For further enhancement, the approach can be extended by applying numeric binning, advanced preprocessing techniques, or exploring other rule-mining algorithms like FP-Growth for improved efficiency and accuracy.

Project 3

Building a Topic-Focused Search Engine with Crawling and PageRank

Concept Brief

This project provides a working domain-specific search engine that duplicates the basic functioning of large-scale search engines. The study involves activities of crawling pages, indexing page/text information, and analyzing links to produce ranked search results oriented toward a particular domain (for example, learning platforms).

Objectives

The primary goals of this project were to design and implement a simplified, domain-specific search engine that simulates the core functionalities of modern web search technologies. The objectives included:

1. **Crawl Webpages**: Collect and extract content from a focused set of web pages within a specific domain (e.g., education or learning platforms) using seed URLs.
2. **Build Inverted Index**: Process the extracted content to create an inverted index that maps keywords to URLs for efficient keyword-based search.
3. **Construct Link Graph**: Analyze hyperlinks between the crawled pages to form a web connection graph representing the structure of the domain.
4. **Apply PageRank Algorithm**: Compute PageRank scores for each page in the graph to evaluate their relative importance based on link structure.
5. **Develop Search Functionality**: Implement a search interface that accepts user queries and returns a list of relevant pages ranked according to their PageRank scores.

Implementation Workflow

1. Web Crawling

Using Python libraries like requests and BeautifulSoup, the crawler navigated through web pages, staying within the target domain. Each visited page's text was cleaned and tokenized, and all hyperlinks were collected to build the web graph.

2. Text Preprocessing & Indexing

All retrieved text was converted to lowercase, punctuation was removed, and common stopwords were filtered out using NLTK. The result was stored in an inverted index that linked each word to the URLs containing it.

3. Graph Construction

As the crawler explored links, it recorded the source and target of every hyperlink. This data was used to form a directed graph (DiGraph) representing the linking structure between pages.

4. PageRank Calculation

Using NetworkX, the PageRank algorithm was applied to the graph. Each page was assigned a score based on the number and quality of incoming links, simulating how real-world search engines evaluate page authority.

Search Interface

A basic search engine was built where users could enter a keyword. The system would:

- Retrieve all pages containing that keyword from the inverted index.
- Filter intersecting results if multiple terms were used.
- Rank the results using their PageRank scores and return them in descending order of importance.

Code:

```
import requests
from bs4 import BeautifulSoup

import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords

STOPWORDS = stopwords.words('english')
print(STOPWORDS)

custom_STOPWORDS = [] # Add your own stopwords here
STOPWORDS.extend(custom_STOPWORDS)

from collections import defaultdict

# Inverted index: word -> set of URLs
inverted_index = defaultdict(set)
url_list = set()

# This dictionary will be used to build the connection between links
web_connection = {'source':[], 'target':[]}
```

```python
import re

# This function will clean the content of web page in order to build the
inverted index.
def clean_and_tokenize(text):
    text = re.sub(r'[^a-zA-Z0-9\s]', '', text.lower())  # Remove
punctuation and lowercase
    tokens = text.split()
    return [t for t in tokens if t not in STOPWORDS and len(t) > 1]

from urllib.parse import urljoin, urlparse

# The crawl function has 5 parameters
# url = The url to crawl
# base_domain = the base domain of the url. During crawling, the crawler
will ignore links from other domains

def crawl(url, base_domain, visited, visit_limit, limit):
    if limit==0 or len(visited)==visit_limit:
        return

    try:
        response = requests.get(url, timeout=5)
        if response.status_code != 200:
            return
    except requests.RequestException:
        return

    visited.add(url)
    print("-"*(10-limit), end=" ")
    print(f"Crawled: {url}")

    soup = BeautifulSoup(response.text, 'html.parser')
    text = soup.get_text(separator=' ', strip=True)
    words = clean_and_tokenize(text)

    for word in words:
        inverted_index[word].add(url)
        url_list.add(url)

    # Recursively follow links
    for tag in soup.find_all('a', href=True):
        link = urljoin(url, tag['href'])
        parsed = urlparse(link)
```

```
        # Store external links as connection
        web_connection['source'].append(url)
        web_connection['target'].append(link)

        if parsed.netloc == base_domain and link not in visited:
            crawl(link, base_domain, visited, visit_limit, limit-1)

def crawl_roots(root_urls, max_per_root=2, visit_limit=50):
    for root in root_urls:
        print(f"\nStarting crawl from: {root}")
        domain = urlparse(root).netloc
        visited = set()
        crawl(root, domain, visited, visit_limit, max_per_root)

seed_urls = [
    'https://www.weather.com/',
    'https://www.accuweather.com/',
    'https://www.wunderground.com/',
    'https://www.noaa.gov/',
    'https://www.weatherbug.com/',
    'https://weather.com/en-IN',
    'https://www.windy.com/',
    'https://www.ventus.info/',
    'https://www.meteoblue.com/',
    'https://www.yr.no/en',
    'https://www.dwd.de/EN',
    'https://www.metoffice.gov.uk/',
    'https://www.bom.gov.au/',
    'https://www.ec.gc.ca/meteo-weather/',
    'https://www.weatheronline.co.uk/',
    'https://www.intellicast.com/',
    'https://www.theweathernetwork.com/',
    'https://www.rainviewer.com/',
]

crawl_roots(seed_urls, max_per_root=10)

# print inverted index
print("\nSample inverted index (first 20 words):")
for word in list(inverted_index.keys())[:20]:
    print(f"{word}: {list(inverted_index[word])}")


# Print first 20 connections
```

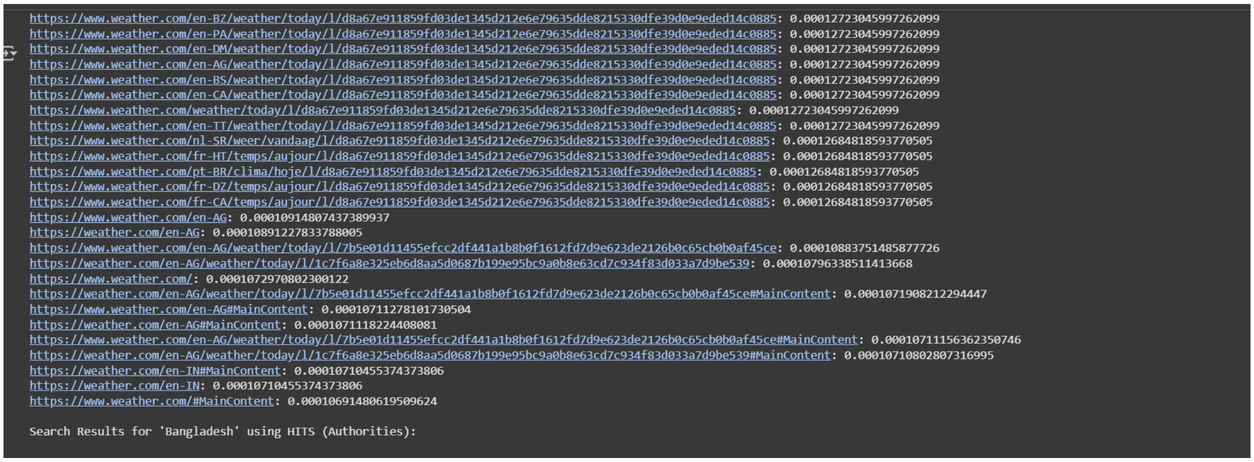
```
for source, target in list(zip(web_connection['source'],
web_connection['target']))[:20]:
    print(f"{source} -> {target}")

import networkx as nx
import pandas as pd # Import pandas

# Create a DataFrame from the web_connection dictionary
edges_df = pd.DataFrame(web_connection)

web_graph = nx.DiGraph()
for _, row in edges_df.iterrows():
    web_graph.add_edge(row['source'], row['target']) # Access source and
target from row

len(web_graph.nodes())

pagerank_scores = nx.pagerank(web_graph, alpha=0.85, max_iter=100, tol=1e-
6)
print("\nPageRank Scores:", pagerank_scores)

def search_engine(query, index, scores):
    query_terms = query.lower().split()
    results = set()
    for term in query_terms:
        if term in index:
            if not results:
                results = set(index[term])
            else:
                results = results.intersection(index[term])  # Find common
websites

    # Sort results based on score
    ranked_results = []
    for website in results:
        if website in scores:
          ranked_results.append((website, scores[website]))
    ranked_results.sort(key=lambda x: x[1], reverse=True)

    return ranked_results

# Query and display results
query = "Bangladesh"
print(f"\nSearch Results for '{query}' using PageRank:")
```

```
results = search_engine(query, inverted_index, pagerank_scores) # Change
index to inverted_index

for page, score in results:
    print(f"{page}: {score}")  # Print page and score. web_content isn't
defined. # Remove reference to undefined web_content

print(f"\nSearch Results for '{query}' using HITS (Authorities):")
#results = search_engine(query, index, authorities)  # This part is
commented out as 'authorities' is also not defined.
#for page, score in results:
#    print(f"{page}: {score}")# Remove reference to undefined authorities
and web_content
```

Output:

```
Search Results for 'Bangladesh' using PageRank:
https://www.rainviewer.com/sources.html: 0.0003628151605435362 6
https://www.meteoblue.com/en/climateplus: 0.00016314770352384851
https://www.meteoblue.com/en/historyplus: 0.0001629157343761289
https://www.meteoblue.com/en/ad-free: 0.0001610539562518184 8
https://www.meteoblue.com/en/pointplus: 0.0001610539562518184 8
https://weather.com/en-VC/weather/today/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.0001288496303212923 7
https://weather.com/en-BS/weather/today/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.0001288496303212923 7
https://weather.com/en-DM/weather/today/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.0001288496303212923 7
https://weather.com/en-AG/weather/today/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.0001288496303212923 7
https://weather.com/en-LC/weather/today/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.0001288496303212923 7
https://weather.com/en-CA/weather/today/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.0001288496303212923 7
https://weather.com/en-PA/weather/today/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.0001288496303212923 7
https://weather.com/en-TT/weather/today/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.0001288496303212923 7
https://weather.com/en-GY/weather/today/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.0001288496303212923 7
https://weather.com/en-JM/weather/today/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.0001288496303212923 7
https://weather.com/en-BB/weather/today/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.0001288496303212923 7
https://weather.com/weather/today/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.0001288496303212923 7
https://weather.com/en-BZ/weather/today/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.0001288496303212923 7
https://weather.com/en-GD/weather/today/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.0001288496303212923 7
https://weather.com/en-KN/weather/today/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.0001288496303212923 7
https://weather.com/fr-HT/temps/aujour/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.00012846122389198242
https://weather.com/nl-SR/weer/vandaag/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.00012846122389198242
https://weather.com/fr-BJ/temps/aujour/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.00012846122389198242
https://weather.com/pt-BR/clima/hoje/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.00012846122389198242
https://weather.com/fr-DZ/temps/aujour/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.00012846122389198242
https://weather.com/fr-CA/temps/aujour/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.00012846122389198242
https://www.weather.com/en-GD/weather/today/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012723045997262099
https://www.weather.com/en-GY/weather/today/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012723045997262099
https://www.weather.com/en-JM/weather/today/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012723045997262099
https://www.weather.com/en-KN/weather/today/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012723045997262099
https://www.weather.com/en-VC/weather/today/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012723045997262099
```

```
https://weather.com/weather/today/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.0001288496303212923 7
https://weather.com/en-BZ/weather/today/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.0001288496303212923 7
https://weather.com/en-GD/weather/today/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.0001288496303212923 7
https://weather.com/en-KN/weather/today/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.0001288496303212923 7
https://weather.com/fr-HT/temps/aujour/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.00012846122389198242
https://weather.com/nl-SR/weer/vandaag/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.00012846122389198242
https://weather.com/fr-BJ/temps/aujour/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.00012846122389198242
https://weather.com/pt-BR/clima/hoje/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.00012846122389198242
https://weather.com/fr-DZ/temps/aujour/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.00012846122389198242
https://weather.com/fr-CA/temps/aujour/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.00012846122389198242
https://www.weather.com/en-GD/weather/today/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012723045997262099
https://www.weather.com/en-GY/weather/today/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012723045997262099
https://www.weather.com/en-JM/weather/today/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012723045997262099
https://www.weather.com/en-KN/weather/today/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012723045997262099
https://www.weather.com/en-VC/weather/today/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012723045997262099
https://www.weather.com/en-LC/weather/today/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012723045997262099
https://www.weather.com/en-BB/weather/today/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012723045997262099
https://www.weather.com/en-BZ/weather/today/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012723045997262099
https://www.weather.com/en-PA/weather/today/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012723045997262099
https://www.weather.com/en-DM/weather/today/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012723045997262099
https://www.weather.com/en-AG/weather/today/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012723045997262099
https://www.weather.com/en-BS/weather/today/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012723045997262099
https://www.weather.com/en-CA/weather/today/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012723045997262099
https://www.weather.com/weather/today/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012723045997262099
https://www.weather.com/en-TT/weather/today/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012723045997262099
https://www.weather.com/nl-SR/weer/vandaag/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012684818593770505
https://www.weather.com/fr-HT/temps/aujour/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012684818593770505
https://www.weather.com/pt-BR/clima/hoje/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012684818593770505
https://www.weather.com/fr-DZ/temps/aujour/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012684818593770505
https://www.weather.com/fr-CA/temps/aujour/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012684818593770505
https://www.weather.com/en-AG: 0.00010914807437389937
https://weather.com/en-AG: 0.00010891227833788005
https://www.weather.com/en-AG/weather/today/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.00010883751485877726
https://weather.com/en-AG/weather/today/l/1c7f6a8e325eb6d8aa5d0687b199e95bc9a0b8e63cd7c934f83d033a7d9be539: 0.0001079633851141366 8
```


```
https://www.weather.com/en-BZ/weather/today/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012723045997262099
https://www.weather.com/en-PA/weather/today/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012723045997262099
https://www.weather.com/en-DM/weather/today/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012723045997262099
https://www.weather.com/en-AG/weather/today/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012723045997262099
https://www.weather.com/en-BS/weather/today/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012723045997262099
https://www.weather.com/en-CA/weather/today/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012723045997262099
https://www.weather.com/weather/today/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012723045997262099
https://www.weather.com/en-TT/weather/today/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012723045997262099
https://www.weather.com/nl-SR/weer/vandaag/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012684818593770505
https://www.weather.com/fr-HT/temps/aujour/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012684818593770505
https://www.weather.com/pt-BR/clima/hoje/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012684818593770505
https://www.weather.com/fr-DZ/temps/aujour/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012684818593770505
https://www.weather.com/fr-CA/temps/aujour/l/d8a67e911859fd03de1345d212e6e79635dde8215330dfe39d0e9eded14c0885: 0.00012684818593770505
https://www.weather.com/en-AG: 0.00010914807437389937
https://weather.com/en-AG: 0.00010891227833788005
https://www.weather.com/en-AG/weather/today/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce: 0.00010883751485877726
https://weather.com/en-AG/weather/today/l/1c7f6a8e325eb6d8aa5d0687b199e95bc9a0b8e63cd7c934f83d033a7d9be539: 0.00010796338511413668
https://www.weather.com/: 0.0001072970802300122
https://weather.com/en-AG/weather/today/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce#MainContent: 0.0001071908212294447
https://www.weather.com/en-AG#MainContent: 0.00010711278101730504
https://weather.com/en-AG#MainContent: 0.0001071118224408081
https://www.weather.com/en-AG/weather/today/l/7b5e01d11455efcc2df441a1b8b0f1612fd7d9e623de2126b0c65cb0b0af45ce#MainContent: 0.00010711156362350746
https://weather.com/en-AG/weather/today/l/1c7f6a8e325eb6d8aa5d0687b199e95bc9a0b8e63cd7c934f83d033a7d9be539#MainContent: 0.00010710802807316995
https://weather.com/en-IN#MainContent: 0.00010710455374373806
https://weather.com/en-IN: 0.00010710455374373806
https://www.weather.com/#MainContent: 0.00010691480619509624

Search Results for 'Bangladesh' using HITS (Authorities):
```

Demonstration

For a query like "math", the engine successfully returned relevant domain-specific pages ranked by importance (PageRank). The system accounts for both content relevance and link structure.

Outcome Analysis

The engine not only identified relevant pages containing the search term but also prioritized those with higher authority based on interlinking, simulating how search engines evaluate trustworthiness.

Conclusion

This project successfully delivered a lightweight but functional domain-specific search engine. By combining traditional keyword indexing with PageRank-based ranking, it effectively prioritized relevant and authoritative pages. This hybrid approach reflects the principles of real-world search engines, though on a smaller scale.

Github Links:

1. **Project-2**
2. **Project-3**