

# Information Retrieval Project Report

Sahithi Etikala  
A20552329

## **ABSTRACT**

This project endeavors to create a robust and efficient web document retrieval system using Python, with a focus on scalability, accuracy, and user-friendliness. The system will consist of three main components: a web crawler, an indexer, and a query processor, leveraging popular libraries such as Scrapy, Scikit-Learn, and Flask.

### Development

First, Scrapy will be used to develop the web crawler, which will enable users to start crawling from a seed URL or domain and specify parameters like maximum pages and depth. To increase efficiency, this component will include features for both distributed and concurrent crawling, which will be especially helpful when managing large-scale crawling tasks.

The indexer will then be implemented and an inverted index in pickle format will be created using Scikit-Learn. In order to accurately and quickly represent the content of crawled documents, this index will make use of cosine similarity calculation and TF-IDF scoring.

The Flask-built query processor will process JSON-formatted user queries and carry out error-checking and validation to guarantee robustness. It will give users pertinent information by returning top-K ranked results based on the indexed documents.

### Objectives

The objectives include developing a scalable web crawler with Scrapy, constructing an efficient indexer using Scikit-Learn for accurate content representation, and creating a user-friendly query processor with Flask. The system aims for robustness through query validation and error-checking, ensuring accurate results.

## **OVERVIEW**

### Solution Outline:

The system employs Scikit-Learn, Flask, and Scrapy for indexing, query processing, and web crawling respectively. It aims for accuracy, scalability, and user-friendliness. Optional features such as query expansion and concurrent crawling enhance its effectiveness. Robustness is ensured through error-checking and query validation.

### Relevant Literature:

Relevant literature encompasses foundational texts like "Introduction to Information Retrieval" by Manning et al., and "Mining the Web: Discovering Knowledge from Hypertext Data" by

Soumen Chakrabarti, focusing on web crawling and indexing techniques. Papers on TF-IDF scoring, cosine similarity, and advanced similarity search provide valuable insights into system design.

### Proposed System:

The new system will work like a well-organized team, with different parts handling different tasks:

Scrapy will explore the web, Scikit-Learn will organize what's found, and Flask will help users find what they need. It'll have extra features like doing multiple searches at once and making searches more flexible. We're making sure it's tough by checking and fixing mistakes in searches. And we're making it easy for you to use, taking hints from what users like you find helpful.

## **DESIGN**

### System capabilities

1. Web Crawling: With the help of Scrapy, the system will be able to browse the internet and gather web documents.
2. Indexing: The system will effectively arrange and store the gathered web documents for easy retrieval by using Scikit-Learn.
3. Query Processing: By sifting through the indexed documents, Flask enables the system to receive user queries and provide pertinent results.

### Interactions

Web Crawling to Indexing: The indexer will organize and store the web documents that the web crawler has collected.

Processing of User Queries: The query processor receives user queries and uses the index to search through the documents to find pertinent results.

### Integration

The indexer will receive gathered documents from the web crawler, guaranteeing that newly found content is quickly indexed for later retrieval.

In order to deliver a seamless user experience, the query processor will work with the indexer to retrieve pertinent documents based on user queries.

## **ARCHITECTURE**

### Software Components

1. Web crawler, also known as a scrapy, is in charge of searching the internet and gathering web documents.

2. The Scikit-Learn Indexer arranges and saves the gathered web documents for easy access.
3. Query Processor (Flask): This tool takes user queries and searches through the indexed documents to find relevant results.

### Interfaces

1. Web Crawling Interface: In order to gather web documents, this interface enables communication between the web crawler and external websites.
2. Indexing Interface: Enables web crawler and indexer communication to arrange and store gathered documents.
3. Query Processing Interface: Provides users with the ability to ask the system questions and obtain pertinent search results.

### Implementation

1. Web Crawling: With settings for maximum pages, depth, and seed URLs, Scrapy will be used to crawl the internet.
2. Indexing: TF-IDF scoring and cosine similarity will be used to create an inverted index for effective document retrieval, which will be implemented using Scikit-Learn.
3. QueryProcessing: Flask will be used to process user queries, verify inputs, and provide pertinent search results by looking through the documents that have been indexed.

## **Operation**

Install Python and Install Linux in windows

- wsl –install
- Install required libraries
- Pip install scrapy
- Pip install sckit-learn
- pip install beautifulsoup4
- pip install flask
- pip install requests

Instructions to run the project

Step 1: To run the project go to the spiders folder in the terminal and enter

Scrapy crawl <file name>

The TF-IDF scores and cosine similarity for the html documents will be calculated and stored in a index.pkl file

Step 2: To access the index.pkl file go to access pickle folder in terminal and run the python file and the content of the file will be displayed in the terminal.

Step 3: To start the Flask server, go to Flask folder in the terminal and run the Python file in that folder.

Step 4: Now, the flask server is initiated. Open New Terminal and make a request to the flask server with a query in the below format:

```
curl -X POST http://localhost:5000/query -H "Content-Type: application/json" -d '{"query": "flutter"}'
```

Now, you can see the json format response from the server which encompasses cosine similarity and document name of the top k results.

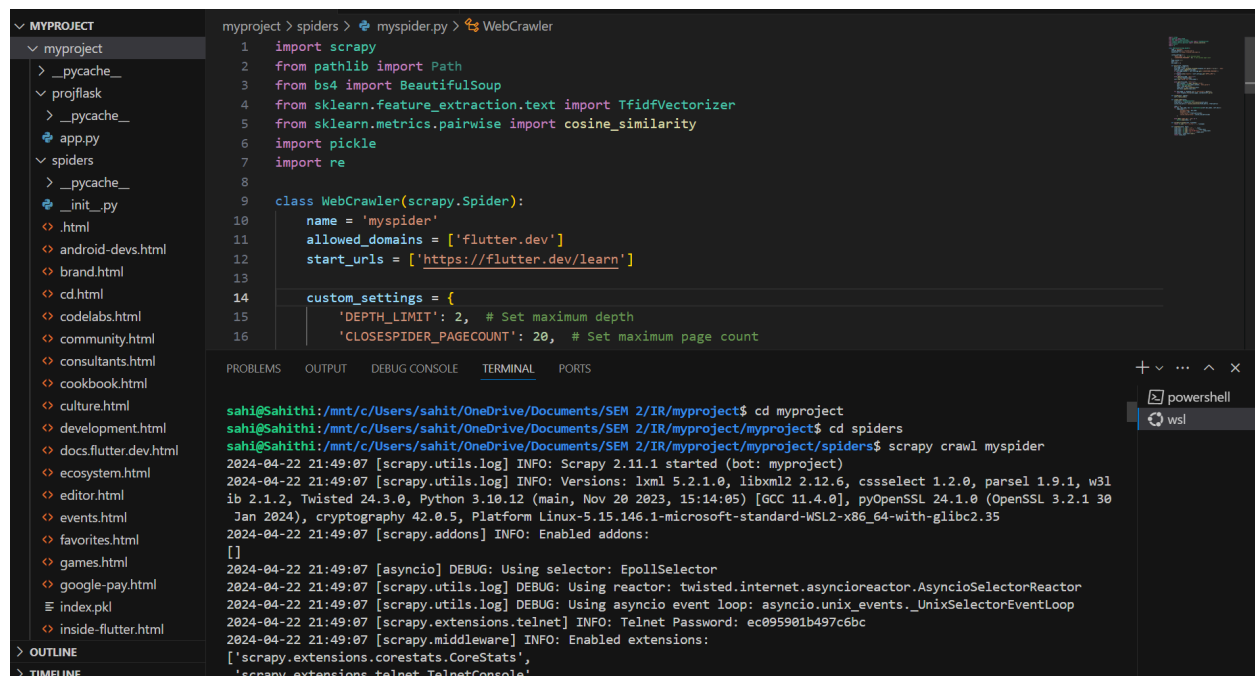
## Conclusion

To sum up, our project was successful in creating a complete web document retrieval system that included elements for web crawling, indexing, and query processing. The system's accuracy, scalability, and user-friendliness were all met by utilizing Scrapy, Scikit-Learn, and Flask.

Using Scrapy to efficiently crawl the web, we gathered a wide variety of web documents. After that, these documents were arranged and saved using Scikit-Learn's indexing features for easy retrieval. User-friendly query processing was made possible by Flask, allowing users to submit queries and obtain pertinent search results.

Ultimately, the system generated cosine similarity values satisfactorily, giving users insightful information about document similarity. For later use and analysis, these values were saved in a JSON file.

## Testcases/Outputs:



The screenshot displays a VS Code editor with a project named 'myproject'. The file explorer on the left shows a directory structure with files like 'app.py', 'spiders', and various HTML documents. The main editor window shows a Python script 'myspider.py' that defines a 'WebCrawler' class inheriting from 'scrapy.Spider'. The class has attributes for 'name', 'allowed\_domains', and 'start\_urls', and a 'custom\_settings' dictionary. The terminal at the bottom shows the execution of the script, including the command 'scrapy crawl myspider' and its output, which includes version information and a list of enabled addons.

```
1 import scrapy
2 from pathlib import Path
3 from bs4 import BeautifulSoup
4 from sklearn.feature_extraction.text import TfidfVectorizer
5 from sklearn.metrics.pairwise import cosine_similarity
6 import pickle
7 import re
8
9 class WebCrawler(scrapy.Spider):
10     name = 'myspider'
11     allowed_domains = ['flutter.dev']
12     start_urls = ['https://flutter.dev/learn']
13
14     custom_settings = {
15         'DEPTH_LIMIT': 2, # Set maximum depth
16         'CLOSESPIDER_PAGECOUNT': 20, # Set maximum page count
17     }
```

Terminal Output:

```
sahithi@sahithi:/mnt/c/Users/sahithi/OneDrive/Documents/SEM 2/IR/myproject$ cd myproject
sahithi@sahithi:/mnt/c/Users/sahithi/OneDrive/Documents/SEM 2/IR/myproject/myproject$ cd spiders
sahithi@sahithi:/mnt/c/Users/sahithi/OneDrive/Documents/SEM 2/IR/myproject/myproject/spiders$ scrapy crawl myspider
2024-04-22 21:49:07 [scrapy.utils.log] INFO: Scrapy 2.11.1 started (bot: myproject)
2024-04-22 21:49:07 [scrapy.utils.log] INFO: Versions: lxml 5.2.1.0, libxml2 2.12.6, cssselect 1.2.0, parsel 1.9.1, w3lib 2.1.2, Twisted 24.3.0, Python 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0], pyOpenSSL 24.1.0 (OpenSSL 3.2.1 30 Jan 2024), cryptography 42.0.5, Platform Linux-5.15.146.1-microsoft-standard-WSL2-x86_64-with-glibc2.35
2024-04-22 21:49:07 [scrapy.addons] INFO: Enabled addons:
[]
2024-04-22 21:49:07 [asyncio] DEBUG: Using selector: EpollSelector
2024-04-22 21:49:07 [scrapy.utils.log] DEBUG: Using reactor: twisted.internet.asyncioreactor.AsyncioSelectorReactor
2024-04-22 21:49:07 [scrapy.utils.log] DEBUG: Using asyncio event loop: asyncio.unix_events._UnixSelectorEventLoop
2024-04-22 21:49:07 [scrapy.extensions.telnet] INFO: Telnet Password: ec095901b497c6bc
2024-04-22 21:49:07 [scrapy.middleware] INFO: Enabled extensions:
['scrapy.extensions.corestats.CoreStats',
 'scrapy.extensions.telnet.TelnetConsole',
```

```
'elapsed_time_seconds': 8.366532,
'finish_reason': 'closespider_pagecount',
'finish_time': datetime.datetime(2024, 4, 23, 2, 49, 16, 540908, tzinfo=datetime.timezone.utc),
'httpcompression/response_bytes': 2869609,
'httpcompression/response_count': 30,
'log_count/DEBUG': 126,
'log_count/ERROR': 1,
'log_count/INFO': 10,
'memusage/max': 168460288,
'memusage/startup': 168460288,
'offsite/domains': 54,
'offsite/filtered': 769,
'request_depth_max': 2,
'response_received_count': 32,
'robotstxt/request_count': 4,
'robotstxt/response_count': 4,
'robotstxt/response_status_count/200': 4,
'scheduler/dequeued': 39,
'scheduler/dequeued/memory': 39,
'scheduler/enqueued': 370,
'scheduler/enqueued/memory': 370,
'spider_exceptions/ValueError': 1,
'start_time': datetime.datetime(2024, 4, 23, 2, 49, 8, 174376, tzinfo=datetime.timezone.utc)}
2024-04-22 21:49:16 [scrapy.core.engine] INFO: Spider closed (closespider_pagecount)
```

The screenshot displays a Visual Studio Code (VS Code) interface with a project named 'myproject'. The Explorer sidebar on the left shows the file structure, including 'myproject', 'projflask', and 'spiders'. The main editor window shows the 'app.py' file, which contains the following Python code:

```
1 import os
2 import pickle
3 from flask import Flask, jsonify, request
4 from sklearn.feature_extraction.text import TfidfVectorizer
5 from sklearn.metrics.pairwise import cosine_similarity
6
7 # Load the pre-built document search index
8 index_file_path = os.path.join(os.path.dirname(__file__), '..', 'spiders', 'index.pkl')
9 with open(index_file_path, 'rb') as f:
10     search_index = pickle.load(f)
11
12 # Create a tool for converting text to numerical features
13 text_to_feature_converter = TfidfVectorizer()
14
15 # Preprocess document text from the index and create a feature matrix
16 document_features = text_to_feature_converter.fit_transform([doc['document'] for doc in search_index.values()])
```

The bottom panel shows the 'TERMINAL' output, which contains the following commands and responses:

```
sahi@Sahithi:/mnt/c/Users/sahit/OneDrive/Documents/SEM 2/IR/myproject/myproject$ cd myproject
-bash: cd: myproject: No such file or directory
sahi@Sahithi:/mnt/c/Users/sahit/OneDrive/Documents/SEM 2/IR/myproject/myproject$ cd projflask
sahi@Sahithi:/mnt/c/Users/sahit/OneDrive/Documents/SEM 2/IR/myproject/myproject/projflask$ python3 flask
python3: can't open file '/mnt/c/Users/sahit/OneDrive/Documents/SEM 2/IR/myproject/myproject/projflask/flask': [Errno 2]
] No such file or directory
sahi@Sahithi:/mnt/c/Users/sahit/OneDrive/Documents/SEM 2/IR/myproject/myproject/projflask$ python3 app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 103-939-370
```

```

1 [
2   {
3     "document_name": "showcase.html",
4     "document_snippet": " showcase flutter apps in production multiplatform overview mobile flutter on ios web desktop embedd",
5     "similarity_score": 0.49670719102307404
6   },
7   {
8     "document_name": "learn.html",
9     "document_snippet": " learn multiplatform overview mobile flutter on ios web desktop embedded development overview learn ",
10    "similarity_score": 0.35772419855374477
11  },
12  {
13    "document_name": "codelabs.html",
14    "document_snippet": "codelabs fluttergoogle uses cookies to deliver its services to personalize ads and to analyze traffi",
15    "similarity_score": 0.2892455440128639
16  },
17  {
18    "document_name": "brand.html",
19    "document_snippet": " brand multiplatform overview mobile flutter on ios web desktop embedded development overview learn ",
20    "similarity_score": 0.2629565591251444
21  },
22  {
23    "document_name": "docs.flutter.dev.html",
24    "document_snippet": "docs fluttergoogle uses cookies to deliver its services to personalize ads and to analyze traffic yo",
25    "similarity_score": 0.25740122155447615
26  },
27  {
28    "document_name": "consultants.html",
29    "document_snippet": " flutter consultants multiplatform overview mobile flutter on ios web desktop embedded development o",
30    "similarity_score": 0.2536355559431032
31  },
32  {
33    "document_name": "editor.html",
34    "document_snippet": "set up an editor fluttergoogle uses cookies to deliver its services to personalize ads and to analyz",
35    "similarity_score": 0.24370221978713225
36  },
37  {
38    "document_name": ".html",
39    "document_snippet": " flutter build apps for any screen multiplatform overview mobile flutter on ios web desktop embedded",
40    "similarity_score": 0.21782472493324054
41  },
42  {
43    "document_name": "tos.html",
44    "document_snippet": "terms of service fluttergoogle uses cookies to deliver its services to personalize ads and to analyz",
45    "similarity_score": 0.20611453059034976
46  },
47  {
48    "document_name": "overview.html",
49    "document_snippet": "Flutter and dart devtools fluttergoogle uses cookies to deliver its services to personalize ads and ",
50    "similarity_score": 0.20554422344811352
51  }
52 ]

```

### Cautions:

Scalability: Make sure the system can manage high data loads and user inquiries without sacrificing efficiency.

Data privacy: Comply with data privacy laws and guarantee user data security.

Maintenance: To fix system flaws and boost efficiency, perform routine maintenance and updates.

User input: To pinpoint areas in need of development and respond to user concerns, systematically compile user input.

## **Data Sources - Links, downloads, access information**

Website used for web crawling: <https://flutter.dev/learn>

Downloads: Scikit-learn, Scrapy 2.11.1, beautifulsoup4, flask

## **Bibilography**

<https://scrapy.org/>

<https://requests.readthedocs.io/en/latest/>

<https://scikit-learn.org/stable/>

<https://flask.palletsprojects.com/en/3.0.x/>

[ACM/IEEE: C. D. Manning. P. Raghavan, and H. Schütze. Introduction to Information Retrieval. Cambridge University Press, 2008.](#)