[CS-429]

Information Retrieval Project Report

Sahithi Balerao

A20552382

Abstract

This project demonstrates a mechanism for retrieving web documents. Web crawling, document indexing, and query processing are all integrated using Python and specialized libraries. The objective is providing accurate and pertinent search results in a user-friendly manner.

Overview

In this project, we are developing a search engine that utilizes TF-IDF (Term Frequency-Inverse Document Frequency) scoring to retrieve and rank web content. The solution outline and proposed system consists of 3 main components a web crawler, indexing engine, query processor. The development process encompasses several pivotal stages to ensure robust and precise search capabilities. Two separate APIs are developed to output top search results based on both standard and advanced indexing methods. This project is informed by Scikit-Learn documentation, recent research on semantic search and KNN methods.

Design

System Capabilities

The system has the following functionalities

Web crawling: The system uses a web crawler built on Scrapy to retrieve web documents from specified URLs while adhering to constraints like traversal depth and maximum page count.

Indexing:
The system uses a Scikit-Learn-based indexing approach to create an inverted index using TF-IDF scores. It also has advanced features like incorporating word embeddings and utilizing FAISS for improved similarity search.

Query Processing: Using cosine similarity and TF-IDF scores, the Flask processor processes user queries, verifies them, and returns the best results. Spell check and query expansion options are also included.

Interactions

Web Crawling to Indexing: The indexing engine uses crawled web content to produce an inverted index with TF-IDF scores.

Indexing to Query Processing: For the purpose of retrieving pertinent documents in response to user requests, the indexing engine generates an inverted index.

User Interaction: By entering queries and obtaining search results, users communicate with the system via the Flask-based query processor.

Integration

Two APIs handle search using standard and advanced indexing, connecting with the indexing engine and query processor. Web data goes to the indexing engine, then to the query processor, which interacts with both APIs. The modular design allows for future feature additions, ensuring an efficient system for accurate search results.

Architecture

Software components

1. **Web Crawler:** A Scrapy-based crawler for fetching web documents.
2. **Indexing Engine:** Scikit-Learn-based engine for TF-IDF indexing and optional advanced methods like word embeddings and FAISS.
3. **Query Processor:** Flask-based module for handling user queries, validation, and result retrieval.
4. **APIs:** Two separate APIs to serve search results based on standard and advanced indexing methods.

Interfaces

API Interfaces: RESTful APIs for search functionality.

Data Interface: Interaction between the crawler, indexing engine, and query processor through data pipelines.

The architecture is designed to ensure seamless interaction between components, with clear interfaces for data flow and modular implementation for scalability and extensibility.

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": "DSA"}'
```

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

Conclusion

The project demonstrates that both APIs return relevant documents as output. However, the ranking of the documents requires work. The Scikit-learn API also returns relevant documents for all queries. However, the precision is good enough. For certain additional inquiries, the results are consistent across all three circumstances.

Test Cases - Framework, harness, coverage

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

```
 'dupefilter/filtered': 10,
 'elapsed_time_seconds': 24.307951,
 'finish_reason': 'finished',
 'finish_time': datetime.datetime(2024, 4, 23, 2, 9, 9, 562368),
 'httpcompression/response_bytes': 145170281,
 'httpcompression/response_count': 452,
 'log_count/DEBUG': 2065,
 'log_count/ERROR': 1,
 'log_count/INFO': 10,
 'memusage/max': 134053888,
 'memusage/startup': 134053888,
 'offsite/domains': 7,
 'offsite/filtered': 8,
 'request_depth_max': 1,
 'response_received_count': 458,
 'robotstxt/forbidden': 9,
 'robotstxt/request_count': 3,
 'robotstxt/response_count': 3,
 'robotstxt/response_status_count/200': 3,
 'scheduler/dequeued': 477,
 'scheduler/dequeued/memory': 477,
 'scheduler/enqueued': 477,
 'scheduler/enqueued/memory': 477,
 'spider_exceptions/ValueError': 1,
 'start_time': datetime.datetime(2024, 4, 23, 2, 8, 45, 254417)}
2024-04-22 21:09:09 [scrapy.core.engine] INFO: Spider closed (finished)
```

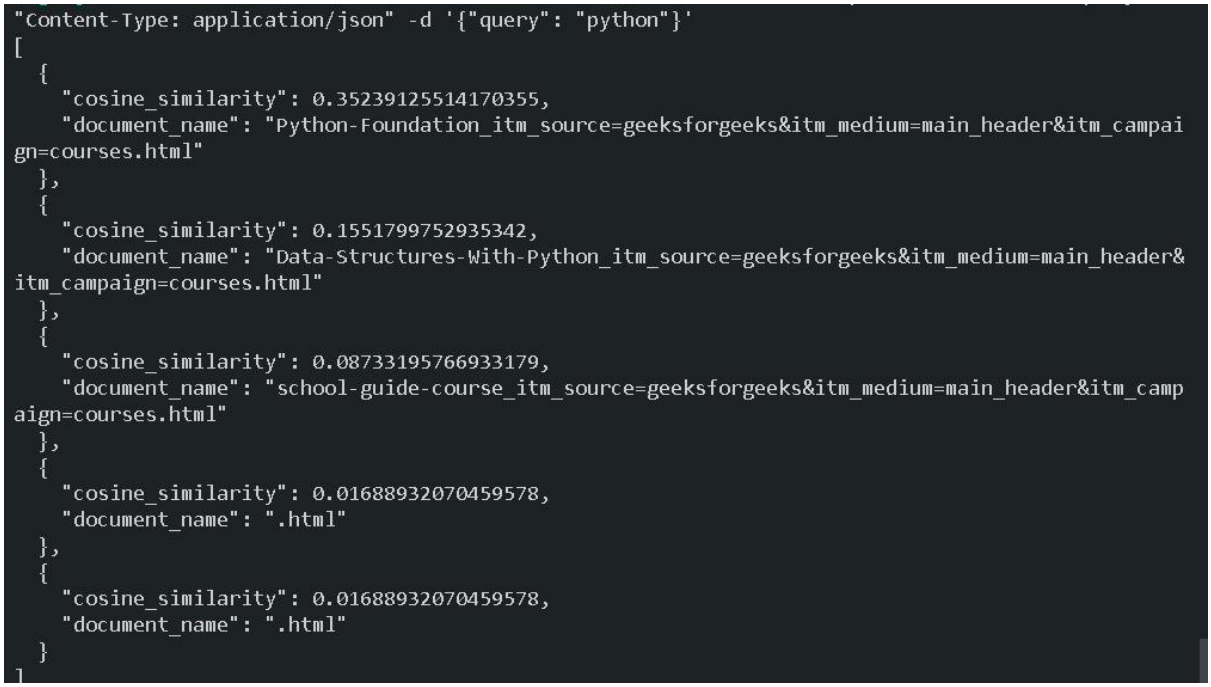
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

```
Document at index 0:
{'document_name': '.html', 'document': ' geeksforgeeks a computer science portal for geeks skip to content
coursesdsa to developmentcoding for everyonefor working professionalsinterview 101 dsa system designdata sc
ience training programjava backend development livedevops engineering livesoftware testing automation lived
ata structures algorithms in pythonfor studentsinterview preparation coursedata science livedata structure
algorithmself paced cjavamaster competitive programming livefull stack development with react node js liveg
ate exam coursesgate cs itgate da aigate classroom course ncrfor school studentscbse class 12 computer scie
nceschool guidepython programming foundationall coursestutorialsdata structures algorithmsdsa for beginners
data structuresarraysmatrixstringslinked liststackqueuetreegeneric treebinary treebinary search treeavl tre
eb treeb treered black treetree data structure tutorialheaphashinggraphset data structuremap data structure
advanced data structuredata structures tutorialalgorithmsanalysis of algorithmssearching algorithmslinear s
earchbinary searchsearching algorithms tutorialsorting algorithmsselection sortbubble sortinsertion sortmer
ge sortquick sortheap sortcounting sortradix sortbucket sortsorting algorithms tutorialgreedy algorithmsdyn
amic programminggraph algorithmspattern searchingrecursionbacktrackingdivide and conquermathematical algori
thmsgeometric algorithmsbitwise algorithmsrandomized algorithmsbranch and boundalgorithms tutorialcomplete
dsa tutorialcompetitive programmingcompany wise sde sheetsfacebook sde sheetamazon sde sheetapple sde sheet
netflix sde sheetgoogle sde sheetwipro coding sheetinfosys coding sheettcs coding sheetcognizant coding she
ethcl coding sheetdsa cheat sheetsdsa sheet for beginnerssde sheetsfaang coding sheetlove babbaar sheetmass
 recruiter sheetproductbased coding sheetcompanywise preparation sheettop 100 dsa interview questions topic
wise100 days of codepythonpython tutorialpython exercisespython list exercisepython string exercisepython t
uple exercisepython dictionary exercisepython set exercisepython excercises top wisepython quizpython progr
amsadvanced python tutorialpython api tutorialpython database tutorialpython jsonpython cheat sheetpython p
rojectspython interview questionsml data sciencemachine learningmachine learning tutorialmaths for mlml pro
jects100 days of machine learningdata science tutorialdata science packagespandas tutorialnumpy tutorialdat
a visualizationdata visualization with pythondata visualization with rtableaupower bidata analysisdata anal
ysis with pythondata analysis with r100 days of data analyticsdeep learningnlp tutorialopencv tutorialinter
view questionsmachine learning interview questionsdeep learning interview questionsr interview questionssys
tem designsystem design tutorialsoftware design patternssystem design roadmaptop 10 system design interview
```

```
 * Serving Flask app 'webpage'
 * 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: 105-108-551
```

TERMINAL PORTS COMMENTS

```
/Documents/IR$ curl -X POST http://loca
lhost:5000/query -H "Content-Type: appl
ication/json" -d '{"query": "DSA"}'
[
  {
    "cosine_similarity": 0.081423179204
06708,
    "document_name": ".html"
  },
  {
    "cosine_similarity": 0.081423179204
06708,
    "document_name": ".html"
  },
  {
    "cosine_similarity": 0.06801095393735423,
    "document_name": "Data-Structures-With-Python_itm_source=geeksforgeeks&itm_medium=main_header&
itm_campaign=courses.html"
  },
  {
    "cosine_similarity": 0.05600681270701862,
    "document_name": "Python-Foundation_itm_source=geeksforgeeks&itm_medium=main_header&itm_campai
gn=courses.html"
  },
  {
    "cosine_similarity": 0.04614392983912188,
    "document_name": "competitive-programming-cp_itm_source=geeksforgeeks&itm_medium=main_header&i
tm_campaign=courses.html"
```



```
"Content-Type: application/json" -d '{"query": "python"}'
[
  {
    "cosine_similarity": 0.35239125514170355,
    "document_name": "Python-Foundation_itm_source=geeksforgeeks&itm_medium=main_header&itm_campai
gn=courses.html"
  },
  {
    "cosine_similarity": 0.1551799752935342,
    "document_name": "Data-Structures-With-Python_itm_source=geeksforgeeks&itm_medium=main_header&
itm_campaign=courses.html"
  },
  {
    "cosine_similarity": 0.08733195766933179,
    "document_name": "school-guide-course_itm_source=geeksforgeeks&itm_medium=main_header&itm_camp
aign=courses.html"
  },
  {
    "cosine_similarity": 0.01688932070459578,
    "document_name": ".html"
  },
  {
    "cosine_similarity": 0.01688932070459578,
    "document_name": ".html"
  }
]
```

Source Code - Listings, documentation, dependencies (open-source).

Data Sources

Scrapy - version 2.11.1
Beautiful Soup - version 4
Scikit-learn - version 1.4.2
Flask - version 3.0.3

Bibliography

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

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

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

https://scrapy.org/