CSCI 4141 Information Retrieval
Assignment 1: Document retrieval with inverted index

May 17, 2024

- Programing language: Python (Jupyter IPython Environment)
- Due Date: Posted in Syllabus
- Join the Assignment Repo on GitHub - *Join GitHub Classroom Assignment 1*

Marking scheme and requirements: Full marks will be given for (1) working, readable, reasonably efficient, documented code that achieves the assignment goals, (2) for providing appropriate answers to the questions in the Jupyter notebook pushed to the student's assignment repository.

Please cite all the resources used – the websites with source code you used should be listed in the submitted Jupyter notebook.

What/how to submit your work:

1. All your code should be included in a notebook that is provided in the assignment repository.
2. For questions that require you to write textual answers use the Markdown option provided by Jupiter Notebook. Ensure that all answers are appropriately numbered as per their corresponding questions.
3. **Push your code to the Github repo and Submit the repo link to Brightspace.** ***Note: Code not pushed on Github before the deadline, along with its link submitted to Brightspace, will not be considered for evaluation.***

Task Overview

This assignment focuses on exploring and implementing fundamental concepts and techniques in information retrieval. The primary objective is to develop a recipe retrieval system that can efficiently search and retrieve relevant recipes based on a given set of ingredients as a query. This would involve implementing term-document incidence matrix, inverted index, Jaccard similarity, and TF-IDF retrieval techniques learnt in class lectures.

***Note*: Please create your own queries to test the retrieval system. To ensure a fair performance comparison across different techniques, you should use the same search queries consistently throughout the assignment. This will allow you to accurately evaluate and compare the effectiveness and efficiency of each approach.**

Q1. Setting up the libraries and environment

To get started with this assignment, you will first need to set up your development environment. This includes cloning the GitHub repository containing the necessary files and installing the required libraries in your Jupyter notebook. This task would help you with that. Installing dependencies is a continuous task that would span across the questions; however, in the end, you should have all the installation commands in the notebook itself and, hence, should have the answer for this.

(a) Clone the GitHub repository (0.5 marks)

(b) Install the required libraries in the notebook. Your assignment notebook should be able to install all the libraries using the commands in the notebook. (0.5 marks)

Marks Distribution: 1/30

Q2. Data Pre-processing

With your environment set up, the next step is to load and pre-process the dataset you will be working with. Data pre-processing is a crucial step that involves cleaning, transforming, and preparing the raw data into a suitable format for analysis. You will need to describe the pre-processing steps you took and display a sample of the processed data.

(a) Load the dataset. (0.5 marks)

(b) Pre-process it. Describe the steps you have taken to pre-process the data and why you have taken those steps. (1 mark)

(c) Display the last 5 rows. (0.5 marks)

Marks Distribution: 2/30

Q3. Term-Document Incidence Matrix

One fundamental concept in information retrieval is the term-document incidence matrix. This matrix represents the occurrence of terms in a collection of documents. In this question, you will create an incidence matrix using an external library and also reproduce a barebone implementation from the tutorial. By comparing the two implementations, you will gain insights into their performance and effectiveness.

(a) Create a incidence matrix using any library (You are free to do your own research and choose the best possible library). (1 mark)

(b) Reproduce the barebone implementation of incidence matrix using the code from the Tutorial. (0.5 mark)

(c) Display the incidence matrix for both. (0.5 mark)

(d) Use the Boolean search implementation from the tutorial to search on both the matrices(One using external library and other is the barebone implementation from the Tutorial). (0.5 marks)

(e) Compare the results, and check if one implementation of incidence matrix performs better than the other one(One using external library and other is the barebone implementation from the Tutorial) and justify your findings. (0.5 marks)

Marks Distribution: 3/30

Q4. Inverted Index

Building upon the concept of the term-document incidence matrix, an inverted index is a data structure commonly used in information retrieval systems. It allows for efficient searching of terms across a large collection of documents. You will implement the core components of an inverted index using the provided skeleton methods and use it to search for a query in the documents.

(a) You have inverted-index-search method given, along with skeleton methods for map-function, reduce-function and create-inverted-index-map-reduce. Complete the implementation of the skeleton methods. Use the inverted-index-search method to search for the query in the documents. Print the results of the search. (3 marks)

Marks Distribution: 3/30

Q5. Inverted Index using Trees

In addition to the basic inverted index, you will explore an alternative implementation using tree data structures. Implementing the inverted index using trees from scratch will deepen your understanding of the underlying concepts. You will also measure the service latency to assess the performance of the tree-based approach.

(a) Implement inverted index using Trees from scratch (This means you cannot use any external library). (2 mark)

(b) Implement the search function to search the query in the documents. (0.5 mark)

(c) Print the results of the search. (0.5 mark)

(d) Calculate the Service Latency using "Averages" to search the query using Trees. (1 mark)

Marks Distribution: 4/30

Q6. Jaccard Similarity

Jaccard similarity is a measure used to quantify the similarity between two sets. In the context of information retrieval, it can be used to determine the similarity between a query and a document. You will implement the Jaccard similarity method and use it to search for a query in the documents. Additionally, you will calculate the service latency to evaluate its performance.

You have the implementation for "jaccard-similarity-search" given below along with the skeleton for the "jaccard-similarity" method.

(a) Complete the "jaccard-similarity" method. Use the "jaccard-similarity-search" method to search for the query in the documents. (1 mark)

(b) Calculate the Service Latency using "Averages" to search the query. (1 mark)

Marks Distribution: 2/30

Q7. TF-IDF

TF-IDF (Term Frequency-Inverse Document Frequency) is a widely used numerical statistic in information retrieval and text mining. It reflects the importance of a term in a document relative to the entire corpus. You will implement a TF-IDF based search using an external library and measure its service latency to assess its efficiency.

(a) Implement the TF-IDF search (You are free to use any external library here). Display the results of the search. (2 marks)

(b) Calculate the Service Latency using "Averages" to search the query using TF-IDF. (1 mark)

Marks Distribution: 3/30

Q8. Search index using Whoosh

At this point, you must have learned how the internals of IR modules work and interact amongst themselves. However, in reality, you may not be writing the barebone implementation and would use a pre-existing solution that helps your solution to scale to a larger corpus more efficiently for real-world use cases. This question would use one such library called *Whoosh* which is a library that allows you to develop custom search engines for your content.

(a) Create a schema for the index, defining the required fields. Use appropriate field types and indexing options. (1 mark)

(b) Create an index, add the recipe data to the index, and commit the changes(This is not a git commit; refer to the Whoosh documentation for more info). (2 marks)

(c) Implement a search function that takes a query string and returns the top 5 relevant results based on the recipe title and directions. Whoosh supports various scoring algorithms (*e.g. TF-IDF*). Use an appropriate scoring method from the available and justify the choice in a short paragraph. (3 marks)

(d) Perform searches using the existing queries and display the search results, including the recipe title, directions, and relevance scores. (2 marks)

(e) Evaluate the performance of the Whoosh-based searching:

- Calculate the average time for indexing and searching operations over multiple runs. (1 mark)
- Calculate and analyze the space complexity of the index directory on disk (This means calculating how much memory the created search index directory occupies in your file system and writing a short paragraph discussing different files in the index folder, their purpose, memory usage, etc.). (1 mark)

***Note**: Feel free to take 10% of the data for this question if the index search is taking a longer time.*
Marks Distribution: 10/30

Q9. Performance Discussion

(a) Compare the Service Latency of all the search methods and give a rationale on which one is more efficient. The rationale should include a short analysis of the latencies of all the search methods used in the assignment. (1 mark)

(b) Compare the search results and explain which one performed better and why. The explanation should include a short analysis of the results of all the search methods used in the assignment. (1 mark)

Marks Distribution: 2/30

Validation

(a) Make sure your notebook runs without generating exceptions by restarting it and running all code cells. This can be done by Choosing Kernel −> Restart & Run all. You should see no exceptions.

(b) Make sure to add relevant comments to your code

(c) Make sure to add the answers to short answer-type questions in the notebook.