READ ME.

Q1. Data Preprocessing [20 Marks]

The following steps were taken in the data preprocessing phase:

a. Lowercase the text
b. Perform tokenization
c. Remove stopwords
d. Remove punctuations
e. Remove blank space tokens

Q2.Unigram Inverted Index and Boolean Queries [40 Marks]

Introduction:

This Python script is designed to execute queries on a dataset using an inverted index. The script preprocesses text, builds an inverted index, and executes queries against the index.

Dependencies:

Ensure you have the following dependencies installed:

Python 3.x
pickle module (usually included in Python standard library)
BeautifulSoup (bs4) module for HTML parsing
nltk (Natural Language Toolkit) for text preprocessing
You can install BeautifulSoup and nltk using pip:

Copy code
pip install beautifulsoup4
pip install nltk
You'll also need to download additional resources for nltk. Run Python and execute the following commands:

python
Copy code
import nltk
nltk.download('stopwords')

nltk.download('punkt')
Usage:

Clone or download the repository to your local machine.
Place your dataset in a directory.
Modify the dataset_path variable in the script to point to the directory containing your dataset.
Modify the queries variable to contain the queries you want to execute.
Run the script in your Python environment.
Input Format:

The queries list contains tuples, where each tuple consists of a query and a string of comma-separated operations.
Operations supported are 'AND', 'OR', and 'NOT'.
Output Format:

For each query, the script outputs the following:
Query text
Number of documents retrieved for the query
Names of documents retrieved for the query
Example:

Suppose you have a dataset containing text files and you want to execute queries on it. You can use this script by providing the path to your dataset and specifying the queries. The script will output the number of documents retrieved and their names for each query.

python
Copy code

Sample Test Case

queries = [
("Car bag in a canister", "OR, AND NOT"),
("Coffee brewing techniques in cookbook", "AND, OR NOT, OR")
]

dataset_path = './data/text_files'
output_path = './temp'

Execute queries

results = execute_query(queries, inverted_index)

Output results

for i, result in enumerate(results):
query, count, documents = result
print(f"Query {i+1}: {query}")
print(f"Number of documents retrieved for query {i+1}: {count}")
print(f"Names of the documents retrieved for query {i+1}: {' '.join(documents)}\n")

Q3. Positional Index and Phrase Queries [40 Marks]

Introduction:

This Python script is designed to create and utilize a positional index for processing phrase queries on a dataset. The script preprocesses text, builds a positional index, and executes phrase queries against the index.

Dependencies:

Ensure you have the following dependencies installed:

Python 3.x
pickle module (usually included in Python standard library)
BeautifulSoup (bs4) module for HTML parsing
nltk (Natural Language Toolkit) for text preprocessing
You can install BeautifulSoup and nltk using pip:

Copy code
pip install beautifulsoup4
pip install nltk
You'll also need to download additional resources for nltk. Run Python and execute the following commands:

python
Copy code
import nltk
nltk.download('stopwords')

nltk.download('punkt')
Usage:

Clone or download the repository to your local machine.
Place your dataset in a directory.
Modify the dataset_path variable in the script to point to the directory containing your dataset.
Run the script in your Python environment.
Input Format:

The first line contains N denoting the number of phrase queries to execute.
The next N lines contain phrase queries.
Output Format:

2N lines consisting of the results in the following format:
Number of documents retrieved for each query using the positional index.
Names of documents retrieved for each query using the positional index.
Example:

Suppose you have a dataset containing text files and you want to execute phrase queries. You can use this script by providing the path to your dataset and specifying the phrase queries. The script will output the number of documents retrieved and their names for each query.