Preprocessing:

Approach:

- Lowercasing: Convert words to lowercase.
- Stopword removal: Remove common stopwords
- Tokenization: Split the text into words
- Punctuation removal: Eliminate punctuation marks.
- Blank space removal: Remove any extra space between the words.
- Storing the data in a Dataset directory

Methodologies: Using nltk.tokenize.word_tokenize for tokenization, nltk.corpus.stopwords.words('english') for stop word removal, and basic string operations like lower() for converting lowercase, translate(str.maketrans("", "", string.punctuation)) for removing punctuation

Assumptions:

1. The Preprocessing have been done in above sequence , so in case where input data contains I've it will change to i ve , i is a stop word but it will not get removed since it is in the form of i've and we are tokenizing it later, if we tokenize before removal of stop word then some words like didn't will cause trouble, since it get tokenize into [did,n't] so it wouldn't get removed from stop word despite of the fact didn't is a stop word.
2. Words like "don't" and "don't." are treated as different word. And will remain because stopword will not remove "don't." but later on remove punctuation will remove "." from "don't."

Building the Inverted Index and Boolean queries:

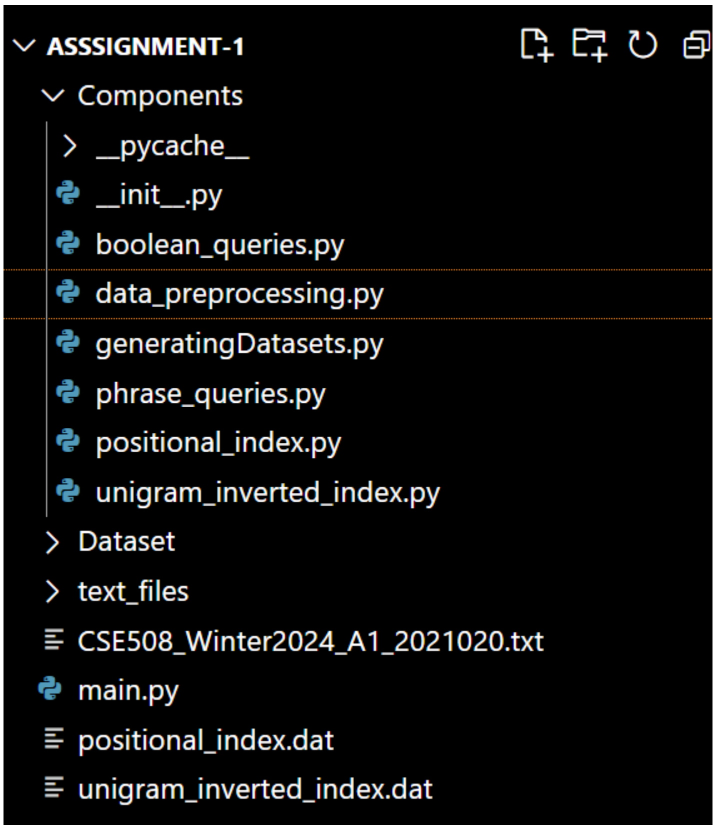
- Iterating through each document in the dataset.
- After that Updating the inverted index dictionary for each term encountered in the document.
- Using Python's pickle module to serialize and save the inverted index.
- Operations Supported:
  1. T1 AND T2
  2. T1 OR T2
  3. T1 AND NOT T2
  4. T1 OR NOT T2
- Input format:
  a. The first line contains N denoting the number of queries to execute
  b. The next 2N lines contain queries in the following format:
    i. Input sequence
    ii. Operations separated by comma
  Output Format:
  a. 3N lines consisting of the results in the following format:
    i. Query X
    ii. Number of documents retrieved for query X
    iii. Name of the documents retrieved for query X (as file1.txt)
- Input Sequence get preprocessed first and then query get formed to execute

- Assumption : If number operations is not equal of one less than length of input preprocessed sequence it may cause an error

Positional Index and Phrase Queries:

- Iterating through each document in the dataset.
- For each term in the preprocessed text, an entry is made in the dictionary with term as key if it does not exist, and the dictionary contain a set of document IDs along with the positions of occurrences of the term in that document.
- Using Python's pickle module to serialize and save the inverted index.
- Input Format:
    a. The first line contains N denoting the number of queries to execute
    b. The next N lines contain phrase queries
- Output Format:
    a. 2N lines consisting of the results in the following format:
        i. Number of documents retrieved for query X using positional index
        ii. Names of documents retrieved for query X using positional index
- Input Sequence get preprocessed first and then query get executed

Assignment Structure



```
ASSSIGNMENT-1
  Components
    __pycache__
    __init__.py
    boolean_queries.py
    data_preprocessing.py
    generatingDatasets.py
    phrase_queries.py
    positional_index.py
    unigram_inverted_index.py
  Dataset
  text_files
  CSE508_Winter2024_A1_2021020.txt
  main.py
  positional_index.dat
  unigram_inverted_index.dat
```

Main.py contains menu driven function for smoothly functioning of above tasks, all the necessary functions stored in the Component folder. text_files directory contains input files and Dataset directory contains preprocessed files.