Assingment 4: Inverted Index

How to run

1. Run in terminal to create the inverted index and save in an output folder
   python3 main.py documents --output-dir ./out_dir --output InvertedIndex
2. Run the search interface in the terminal
   python3 searchInterface.py

Main.py

The mapper function *mapper_get_words* tokenizes the input data by line and then extracts each word from the line. The word is then cleaned by removing punctuation and numbers, stemmed using the SnowballStemmer, and filtered for stopwords. Each word is then emitted as a key-value pair with the stemmed word as the key and a dictionary with the file name and a count of 1 as the value. A vocabulary set is also created to keep track of the unique words encountered.

The reducer function *reducer_merge_counts* receives a list of dictionaries for each word key emitted by the mapper. The dictionaries contain the count of occurrences for each file. The function merges the counts for each file and yields the word key with a dictionary containing the merged counts for each file.

The output of this MapReduce job is an inverted index, where each word is a key and the value is a dictionary containing the count of occurrences for each file. The design choices for this implementation include the use of the SnowballStemmer to reduce words to their root form, the removal of punctuation and numbers, and the filtering of stopwords to reduce noise in the output. The use of a vocabulary set also ensures that each word is only included once in the output.

searchInterface.py:

This file has the code that defines functions to load and search an inverted index created from a set of documents. The inverted index is loaded from files in the 'InvertedIndex' directory (output from the main.py). The user can search for keywords in the index, and the code returns the top 10 documents that contain the keyword. It also uses the NLTK to remove stopwords and to stem keywords. If multiple keywords are entered, the code will return the top 10 documents based on the relevance score calculated from the TF-IDF scores of the keywords. The code also displays the preview of the document and its title.

For the preview, the function *display_search_results* extracts a preview of the relevant text from the top-10 documents that match the search query. It does this by finding the index of the first occurrence of the search keyword in each document and then extracting a snippet of text that includes the keyword and a few words before and after it. This helps the viewer the get a snapshot of the word in the document and the context around it. There is also a *printMenu* function that allows the user to print the menu or get help if they need.