IR ASSIGNMENT-2

R S Sai Prashanth
2110110885

Choula Amarthya
2110110756

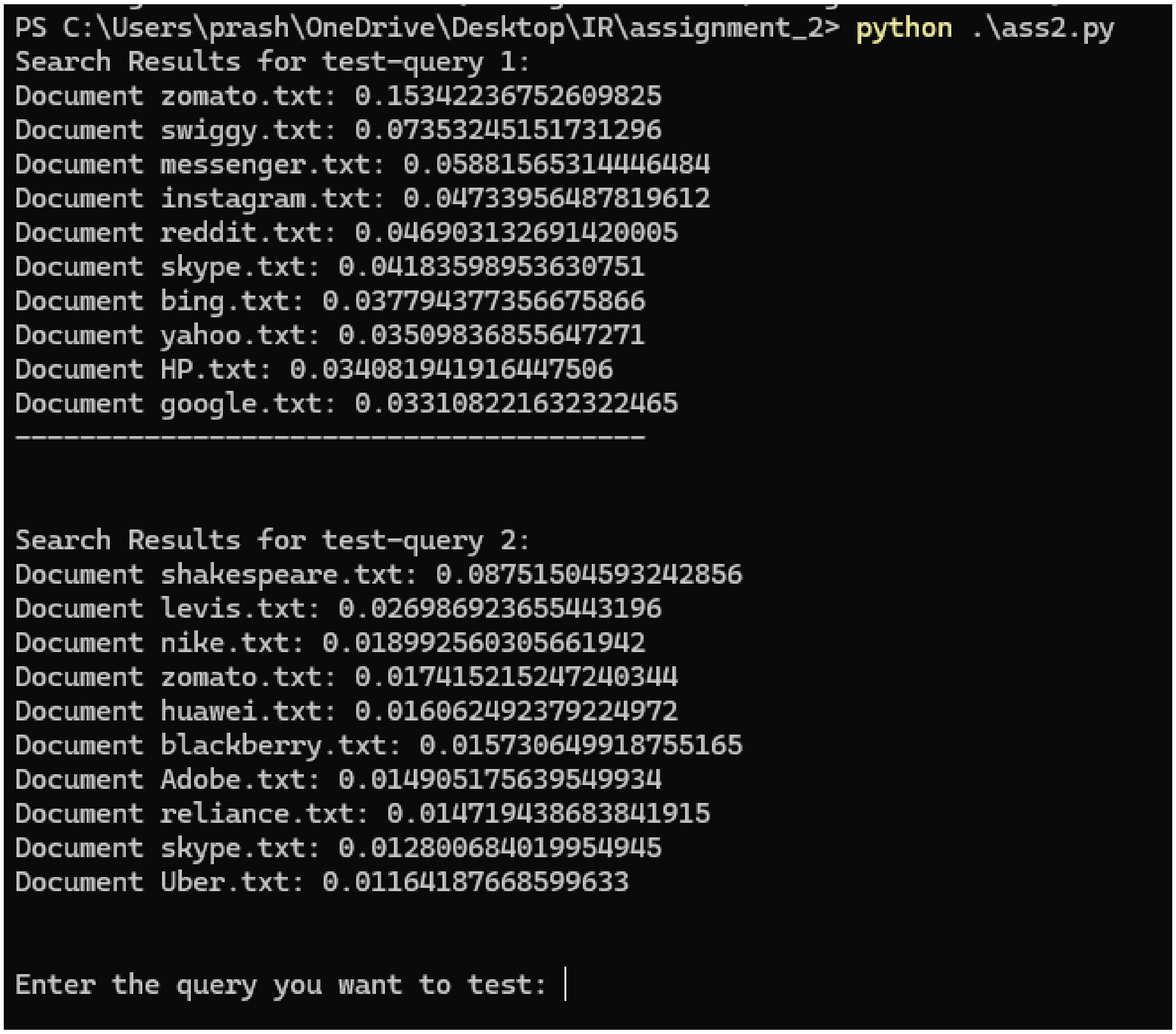
Github link: https://github.com/prash1reddy/IR_ass2

Code:

```
import math
import os
from collections import defaultdict

class VSM:
    def __init__(self):
        # Dictionary to store term frequencies across all documents
        self.dictionary = {}
        # Postings list to store (doc_id, log_tf) for each term
        self.postings = defaultdict(list)
        # Store document lengths for normalization
        self.doc_lengths = {}
        # Total number of documents in the corpus
        self.N = 0
        # Mapping of document IDs to filenames
        self.doc_ids = {}

    def load_corpus(self, directory):
        """
        Load and index all text documents from the specified directory.
        """
        for filename in os.listdir(directory):
            if filename.endswith(".txt"):
                filepath = os.path.join(directory, filename)
                with open(filepath, 'r', encoding='utf-8') as file:
                    content = file.read()
                self.N += 1
                self.doc_ids[self.N] = filename
                self.index_document(self.N, content)

    def index_document(self, doc_id, content):
        """
        Index a single document: tokenize, compute term frequencies,
        and update the dictionary and postings list.
        """
        terms = self.tokenize(content)
        term_freq = defaultdict(int)
        for term in terms:
            term_freq[term] += 1

        doc_length = 0
        for term, tf in term_freq.items():
            # Update document frequency in the dictionary
            if term not in self.dictionary:
                self.dictionary[term] = 1
            else:
                self.dictionary[term] += 1

            # Compute log term frequency
            log_tf = 1 + math.log10(tf) if tf > 0 else 0
            self.postings[term].append((doc_id, log_tf))

            # Update document length for normalization
            doc_length += log_tf ** 2

        # Store the square root of the document length for cosine
normalization
        self.doc_lengths[doc_id] = math.sqrt(doc_length)

    def tokenize(self, text):
        """
        Basic tokenization: convert to lowercase and split on whitespace.
        """
        return text.lower().split()

    def search(self, query):
        """
        Perform a search using the Vector Space Model.
        Returns top 10 documents ranked by cosine similarity.
        """
        query_terms = self.tokenize(query)
        query_weights = defaultdict(float)
        query_length = 0

        # Compute query term weights (tf-idf)
        for term in query_terms:
            tf = 1 + math.log10(query_terms.count(term))
            if term in self.dictionary:
                df = self.dictionary[term]
                idf = math.log10(self.N / df)
                query_weights[term] = tf * idf
                query_length += query_weights[term] ** 2

        # Normalize query vector
        query_length = math.sqrt(query_length)

        # Compute document scores
        scores = defaultdict(float)
        for term, weight in query_weights.items():
            normalized_query_weight = weight / query_length
            for doc_id, log_tf in self.postings[term]:
                tfidf = log_tf * normalized_query_weight
                scores[doc_id] += tfidf

        # Normalize document scores (cosine similarity)
        for doc_id in scores:
            scores[doc_id] /= self.doc_lengths[doc_id]

        # Return top 10 results
        return sorted(scores.items(), key=lambda x: x[1],
reverse=True)[:10]

# Example usage
vsm = VSM()

# Load corpus from a directory
corpus_directory = "Corpus"
vsm.load_corpus(corpus_directory)

# Test queries
```

```python
query1 = """Developing your Zomato business account and profile is a great
way to boost your
restaurants online reputation"""
query2 = """Warwickshire, came from an ancient family and was the heiress
to
some land"""

result1 = vsm.search(query1)
result2 = vsm.search(query2)

print("Search Results for test-query 1:")
for doc_id, score in result1:
    print(f"Document {vsm.doc_ids[doc_id]}: {score}")

print("----------------------------------------\n\n")
print("Search Results for test-query 2:")
for doc_id, score in result2:
    print(f"Document {vsm.doc_ids[doc_id]}: {score}")

print("\n")
# Interactive search
query = input("Enter the query you want to test: ")
results = vsm.search(query)

print("Search Results:")
for doc_id, score in results:
    print(f"Document {vsm.doc_ids[doc_id]}: {score}")
```

Output:



```
PS C:\Users\prash\OneDrive\Desktop\IR\assignment_2> python .\ass2.py
Search Results for test-query 1:
Document zomato.txt: 0.15342236752609825
Document swiggy.txt: 0.07353245151731296
Document messenger.txt: 0.05881565314446484
Document instagram.txt: 0.04733956487819612
Document reddit.txt: 0.046903132691420005
Document skype.txt: 0.04183598953630751
Document bing.txt: 0.037794377356675866
Document yahoo.txt: 0.03509836855647271
Document HP.txt: 0.034081941916447506
Document google.txt: 0.033108221632322465
---------------------------------------


Search Results for test-query 2:
Document shakespeare.txt: 0.08751504593242856
Document levis.txt: 0.026986923655443196
Document nike.txt: 0.018992560305661942
Document zomato.txt: 0.017415215247240344
Document huawei.txt: 0.016062492379224972
Document blackberry.txt: 0.015730649918755165
Document Adobe.txt: 0.014905175639549934
Document reliance.txt: 0.014719438683841915
Document skype.txt: 0.012800684019954945
Document Uber.txt: 0.011641876685996633


Enter the query you want to test:
```