Title: Assignment 4: **Build a simple Vector Space Information Retrieval System**

Author: Zhiheng Wang

Date: 3/23/2019

Description: Implement a simple vector space information retrieval system supporting disjunctive ("OR") queries over terms, and apply it to your wikipedia movie corpus. The system should consist of (1) an indexing module that constructs an inverted index, with term dictionary and postings lists for a corpus and (2) a run-time module that implements a Web UI for searching the corpus, returning a ranked result list and presenting selected documents.

Dependencies: Python 3.6.5

Before running you should have NLTK package (https://www.nltk.org/) installed. Besides you should install flask (http://flask.pocoo.org/)

The easiest way to install it is using pip: pip3 install nltk pip3 install flask **Build Instructions**:

You can install these packages in any sequences.

Run Instructions:

vs_index.py is used for building an inverted index for the database for future information retrieval. The outputs are inverted_index.db, normalized_length.db, and 2018_movie_database.db.

Run vs_index in terminal and you will get these outputs, after that for the fast access to the search engine, comment these lines back. It takes around 20 seconds to get the outputs depends on the machine you use.

vs_query.py has all the functions to do cosine similarity score, rank the results, and runs flask to call our search engine.

Modules:

load_json_file_to_dict(file_path) # load json format data into a dictionary

tokenize(text) # input string text and output a list of words without stopwords which is defined as NLTK stopwords list and string.punctuation

stemming(word) # apply NLTK PorterStemm on word # input string word # output stemmed word

build_inverted_index(file_path) # build inverted_index. Input file path output (inverted index, idf, tfidf) and normalized_length

save_to_shelve(file_path) # save inverted indices to shelve
save_2018_movie_json_to_shelve(file_path) #see name

find_doc_id_list(data, text, unknow_list) # Input a database, a row text, and a unknow_term list Return a set of related docId if found Return a empty set and add to unknown list if not found

dummy_movie_data(doc_id) # Return data fields for a movie.

dummy_movie_snippet(doc_id, query) # Return a snippet for the results page. It includes a title and a short description.

Testing: Test all possible edge cases. Also test on each method to make sure it works.

Text Normalization: tokenize each text string word_tokenizer via NLTK, and then do PorterStemm via NLTK stem.

Shelve Timing: approximately 20 seconds on my machine.

Test Queries Examples:

search: king; hero. Save the worldadfaaga. Have a good day.

Data: two corpus files (test_corpus.json, films_corpus.json).