Assignment 4: Building a simple Vector Space IR System

Author: Ruobin Hu

Date: 4/3/2021

Description:

This program implements a simple vector space information retrieval system supporting disjunctive ("OR") queries over terms and apply it to the TREC corpus. The system consists of two key components:
(1) An indexing module that constructs an inverted index, with term dictionary and postings lists for a corpus.
(2) A run-time module that implements a Web UI for searching the corpus, returning a ranked result list and presenting selected documents.

The user of this application will be able to type in a text query and click the article title to view the whole document information.

Dependencies:

The software used to run the system is Python 3.7 and is obtained from: python.org.

The packages used in this system are Flask, nltk 3.5 and peewee~ 3.14.3.

Build Instructions:

1. For Windows users only: set up git bash: https://gitforwindows.org
2. Create a conda environment with Python 3.7

- Documentation https://docs.conda.io/projects/conda/en/latest/user-guide/
- Install Conda / Miniconda for Python 3.7: https://www.anaconda.com/distribution/
- Create an environment in bash shell: $ conda create -n cosi132a python=3.7
- Activate the environment: $ conda activate cosi132a

3. Activate the cosi132a conda environment and run `pip install -r requirements.txt`, using the file provided.

Run Instructions:

1. Run `python hw4.py --build [INDEX-NAME]` to create the inverted index.
2. Run `python hw4.py --run` in the conda environment and type http://127.0.0.1:5000/ in the browser to view and test the web application.
3. The user interface is the website used to search articles.
4. How to use it:

- Enter the query into the search bar and click search, then documents with matched content of at least one term in the query(non stopword) will be presented in the result page based on their cosine similarity score. The brief descriptions of these articles will also be presented below the title. The result page will also show the total number of hits.
- By clicking the article title, you will be guided to a doc page with the full contents of the article.
- You can go back to the home page by clicking the home button.
- In the result page, you can click the previous or next bottom to browse different results.
- If the input query has stop words or terms not in the dictionary, the result page will have prompts to indicate the user.

- Above the result page, the idf scores for each of the terms in the query will be displayed. And the cosine similarity score of each result will be shown on the left of each title in the result list. A list of query terms that were found in the document will be shown below each document snippet.

Testing:

1. The system was tested with 10 hand-made documents. I first searched some stopwords like “this is a” and the result will be none and these terms will be showed as the ignoring terms.
2. For unknown words, I just search some randomly typed words like “shuish shkh” and the result page will also be none and only showed these terms as the unknown words.
   In order to test for the test normalization, I created a document with the word application and I tried to search the words with the same stem as the application after the stemmer process. For instance, I search for “applicate, applicative, applicable, applicant, applicability” and all of these terms will lead to the result of the document which has the word application.
3. To test the idf score: I made all of the documents contain the word “title” and nine of them contain the word “content”. Then, when I searched the word “title”, it gives me the idf score of 0 and the idf score for searching “content” match the result of log2(10/9).
4. I tested my program by searching some edge cases on the search engine. For instance, I entered nothing and the result page also showed nothing.
5. To test the cosine similarity score: I searched for word “pig” and get one result with the cosine similarity score of 1.1073. Then I calculated the cosine similarity score by hand and get log2(10)/3 which also equals 1.1073.
6. I also searched terms with special symbols like “@#$%^&*><” and the result page will show these special will be unknown words.
7. The tested documents also have one documents that has the type other than the sanitized_html. Another document will also have the html tags in order to test if I successfully remove them.
8. The test result showed that I successfully remove the html tag and the document with other types will not have any contents.
9. I also have documents that contain more frequencies of a words than the others in order to see if it has a higher cosine similarity score.

General comments:

This assignment took me around 10 hours. I spent some time to write the method to calculate the cosine similarity score.