with running instruction, details about the files in the folder, all text normalization details, shelve timing, packages used, test queries examples, and any other thoughts or concerns you like to share.

Running Instruction

You should run this project with following command

```
python boolean_index.py
python boolean_query.py
```

Files in the folder

boolean_index.py

The python file to build inverted index and provide search function for the website.

boolean_query.py

The python file to generate and start the backend python flask framework of this website for user to generate the website and interact with our search engine.

corpus.json corpus11.json

A ful film data original file and a test file.

corpus.db index_data.db stopWords.db

Database file for corpus, index_data and stopWords

Function in the program

loadJson

Load json file from original data

indexAllData

Function for build posting list

tokenize_stemming

Tokenize and stem input

storeStopWords

Down load stopwords from nltk and store it in our database

class SearchEngine

search

Input search query, return list of movie id.

preprocess

Check and delete the stopwords in the query. Check whether there is any unknown term in the query. return result of realquery, unknown term and stopwords.

findMovieId

Input a query (list of query words), output a list of movie id by intersecting posting list of all query words.

intersect

input two list, return the list of intersect movie id between two list.

get_movie_data

return the data of a movie by given id.

get_movie_snippet

return the title, text of a movie by given id.

shelve timing

Using time.clock() function to record the shelve building time in the main function.

```
if __name__ == '__main__':

    start_time = time.clock()
    print('Build Start!')
    data = loadJson(data_path)
    indexAllData(data)
    storeStopWords(stopWords_path)
    end_time = time.clock()
    print('Build End!')
    print('Build Time Use ' + str(end_time - start_time) + ' seconds')
```

Output after running

```
Build Start!
Build End!
Build Time Use 22.910499 seconds
```

Package Used

boolean_index.py

```
import shelve
import nltk
import json
from nltk.tokenize import RegexpTokenizer
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
import time
```

Here we use nltk.tokenize.RegexpTokenizer to tokenize all the words in the corpus(ignore all special mark '{',',' '.','?').

Use PorterStemmer as stemmer to stem words.

Download nltk.corpus.stopwords as our stopwords list

```
stemmer = PorterStemmer()
tokenizer = RegexpTokenizer(r'\w+')
```

```
def storeStopWords(stopWords_path):
    stopW_db = shelve.open(stopWords_path, writeback=False)
    stopW_db['stopWords'] = set(stopwords.words('english'))
    stopW_db.close()
```

boolean_query.py

```
from flask import *
```

Test queries examples

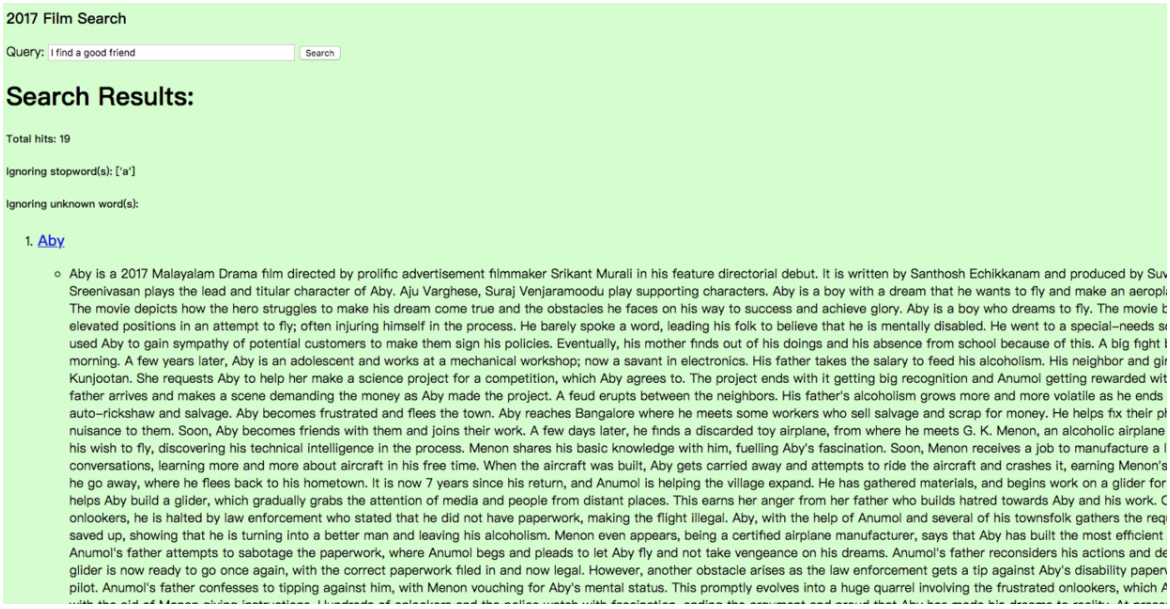
note: Here is some test query example to run in a full corpus database since I have already tested the correctness on my small test corpus.

As we can see, there are 467 hits in the search result.

Add some stopwords in it.

we can see it can ignore the stop words 'as' and 'you'.

testing on a longer query:



That seems great!

verify if intersecting the posting list from small to large

Inorder to verify we intersect the posting list from small to large in order to save time correctly, we add two print statement to print out the length of list we intersect in order in the following command.

```
def findMovieId(self, query):

movieId = []

if len(query) ==0 :

return movieId

posting_list = shelve.open(index_path,writeback=False)

new_query =sorted(query,key =lambda word :len(posting_list[word]))

movieId = posting_list[new_query[0]]

if len(query) ==1 :
```

```
return movieId

print(len(movieId))

for wordin new_query[1:]:

movieId =self.intersect(movieId, posting_list[word])

return movieId

def intersect(self, idList1, idList2):

print(len(idList2))

i =0

    j =0

    res = []

while (i

if idList1[i] < idList2[j]:

i = i +1

        elif idList1[i] > idList2[j]:

j = j +1

        else:

res.append(idList1[i])

i = i +1

            j = j +1

    return res
```

Output

```
127.0.0.1 - - [27/Feb/2018 14:53:56] "GET / HTTP/1.1" 200 -
214
220
467
512
127.0.0.1 - - [27/Feb/2018 14:54:07] "POST /results/1 HTTP/1.1" 200 -
```

That's correct!

Testing an unknown words

Input : I find a good friend asdfweq

After testing, we verify our search engine works correctly! That's great!!

Further thoughts

I think later we can do some advanced search about this movie corpus. Combine text and title, location. And also we can make a positional index system later to eable positional query in the later assignment.