Overview of the Function of the Code

My project is a search engine and recommender for film festival films. In the film industry, there are two critical pieces of information that professionals need to know: 1) what festivals a film has screened at. 2) Films made by a certain director (autor/ filmmaker).

The application that I built fills this need by scraping data from film festival wikipedia pages and then allows the user to search this data and learn what festivals a film has screened at and all the films a director has made. I was able to scrape a sizable amount of data: ~1700 films from the three most highly renowned film festivals (Cannes, Berlin, and Venice) spanning 70 years (1947 - 2018).

My application also has a recommender system which uses the scraped festival data to make it's recommendations. This is quite unique, because to my knowledge there are no movie recommender systems that recommend only high quality, high art, film festival films. Most movie recommenders are built into things like Netflix and Amazon Prime; they work well, but the users of my application (people in the film and film festival industry) usually aren't interested in these kinds of films.

Documentation of How the Software is Implemented

General Search

When the user is trying to search the data to find the name of a film or director, I use the Vector Space Model (VSM) for general searching. Users can search using either or both terms (film title or director) to help them locate their desired film. In this process the search engine returns a ranked list of the top results to the user.

Currently with my implementation of **VSM**, I am performing **Document length Normalization** and **IDF Weighting** (only for Title queries though because there are no "more important" names of people.) I am not, however, using any **term frequency (TF)** weighting (rewarding more matches) because that is not applicable when searching for a title or a name.

I was able to build a fast and scalable searching framework by doing some pre-computation (indexing) tasks on the data. For example, after the film data is scraped I run the indexer. The indexer creates and saves a vocabulary for all of the titles of the films and for all of the director names. The indexer also creates and saves the document vectors for all of the titles of films and directors of films. And since document vector are usually quite large vectors with only a few 1's and the rest being 0's, I compress these vectors by storing only the length of the vectors and the index's of the 1's.

Festival Appearance and Director's Films Search

These two types of searches both look through the data trying to find exact film title or director name matches. If they do find exact matches, they save all the matches they get and display their particular information to the user.

In the case of the Festival Appearance Search, the app displays a list of the festivals the film screened at, allowing one to visualize the film festival network.

And in the case of the Director's Films search, the app displays a list of films the director has made, allowing you to learn more about the filmmaker.

However, if the user doesn't know the full name of the film or director, the app will recognize this and return a general search of the incomplete info given by the user. This will help them locate the actual film/director they are searching for and use this info the get an exact match (disregarding capitalization and punctuation).

Recommender System

My recommender system is a **content-based similarity recommender**. It basically takes the films the user likes, scores the similarity between each of them and each film in the database. Then it returns the top five. My similarity function uses the following criteria for similarity: 1) year the film was made 2) Director 3) Country 3) Festival. I would have really liked to use genre, but that info wasn't readily available in the wiki pages I was scraping.

Documentation of the Usage of the Software

My application has three main pieces of functionality, which are the following: 1) web-scraping and indexing of film festival Wikipedia pages 2) searching this data 3) film recommendations.

0. Installing Necessary Software

Language
Application built and run using: Python 2.7.10

Libraries
The only library required to run my application is BeautifulSoup, which I use when scraping Wikipedia pages.

If you don't have this library already installed, then install it by running:
```
pip install beautifulsoup4
```

1. Scraping Data

To scrape the film data, all you need to do is run the following command:

```
python webscraper.py
```

The app will then scrape data from 92 wiki links. After the scraping is done, check to ensure the “film_data.txt” file was created. If you’re curious about the pages I’m scraping data from, you can open “webscraper.py” and check them out.

2. Indexing Data

To index the film data, all you need to do is run the following command:

```
python indexer.py
```

The app will then index the data and create four files. Check to ensure they were all created, after the process ends: “title_vocab.txt”, “director_vocab.txt”, “title_vocab_vectors.txt”, and “director_vocab_vectors.txt”

3. Search

3.1 Search by Film to see film festival network

To execute a search by film, use the following command:

```
python search_app.py "<film title here>" "-"
```

This command will either return:

A) A list of the festivals the film has screened at.
OR
B) If the user types in a film title that doesn't get any exact matches, the search engine will then do a general search on the title to get the top matching films (helping the user find the film they are actually looking for).

Example:

```
python search_app.py "the wages of" "-"
```

Returns:

```
(['1953', 'The Wages of Fear', 'Henri-Georges Clouzot', 'France', 'Cannes',
'Award'], 7.710809571830852)
(['1953', 'The Wages of Fear', 'Henri-Georges Clouzot', 'France', 'Berlin',
'Award'], 7.710809571830852)
```

```
(['1964', 'The Umbrellas of Cherbourg', 'Jacques Demy', 'France', 'Cannes',
'Award'], 5.140539714553902)
…
```

Since there was no exact match, the user sees the helpful results and remembers the fill title of the film they are searching for. User executes new search:

```
python search_app.py "the wages of Fear" "-"
```

Returns:

```
Berlin Film Festival

year:           1953
title:          The Wages of Fear
director:       Henri-Georges Clouzot
country:        France
festival:       Berlin
award?:         Award

Cannes Film Festival

year:           1953
title:          The Wages of Fear
director:       Henri-Georges Clouzot
country:        France
festival:       Cannes
award?:         Award
```

3.2 Search by Director to see their films

To execute a search by director, use the following command:

```
python search_app.py "-" "<film director here>"
```

This command will either return:

A) A list of the films the director has made.
OR
B) If the user types in a director name that doesn't get any exact matches, the search engine will then do a general search on the title to get the top matching films (helping the user find the director they are actually trying to search for).

Example:

```
python search_app.py "-" "wes anderson"
```

Returns:

```
Wes Anderson

year:          2012
title:         Moonrise Kingdom
director:      Wes Anderson
country:       United States
festival:      Cannes
award?:        _

year:          2014
title:         The Grand Budapest Hotel
director:      Wes Anderson
country:       United Kingdom, Germany
festival:      Berlin
award?:        _

year:          2005
title:         The Life Aquatic with Steve Zissou
director:      Wes Anderson
country:       USA
festival:      Berlin
award?:        _

year:          2002
title:         The Royal Tenenbaums
director:      Wes Anderson
country:       USA
festival:      Berlin
award?:        _

year:          2007
title:         The Darjeeling Limited
director:      Wes Anderson
country:       United States
festival:      Venice
award?:        _
```

2.3 General Film Search

To execute a general search, use the following command:

```
python search_app.py "<title OR empty>" "<director OR empty>"
```

Examples:

```
python search_app.py "the tree of" ""

python search_app.py "" "Terrence"

python search_app.py "the tree of" "Terrence"
```

These commands simply search the database and return a ranked list of the search results.

3. Recommendation

To get recommendations, you simply need to input films that you have liked using the following command:

```
python recommendation_app.py “film_1 title” “film_2 title” …
“film_n title”
```

Example:

```
python recommendation_app.py "magnolia" "sex, lies, and
videotapes"
```

This command would then return:

Recommended Films

```
Recommended Films


score: 1.0296474359
year:          2013
title:         Behind the Candelabra
director:      Steven Soderbergh
country:       United States
festival:      Cannes
award?:        _

score: 1.0296474359
year:          2013
title:         Side Effects
```

```
director:       Steven Soderbergh
country:        United States
festival:       Berlin
award?:         _

score: 1.025
year:           1990
title:          Wild at Heart
director:       David Lynch
country:        United States
festival:       Cannes
award?:         Award

score: 1.025
year:           1990
title:          Music Box
director:       Costa-Gavras
country:        United States
festival:       Berlin
award?:         Award

score: 1.025
year:           1999
title:          The Thin Red Line
director:       Terrence Malick
country:        United States
festival:       Berlin
award?:         Award
```

Which, in my opinion is a decent recommendation list for a few reasons:

1) It recommends, some but not ALL films by the same director as “sex, lies, and videotapes” (Steven Soderbergh)
2) It has some 90’s films, which is the year and feel both of the liked movies has.
3) Three of the five recommended are award winners
4) The genre of the recommended movies is actually very similar to the liked ones.

GitHub Link

https://github.com/Athena96/Film-Festival-Search-Engine