Natural Language Processing and Information Retrieval

Team 10 SoulOfQueries

Subject: How to retrieve a list of movies with a given query (e.g., "love" for movie1, movie2, etc) by searching the movie script dataset from https://imsdb.com/

The project

We are making a search engine to retrieve scripts containing a certain keyword(s) in the IMSDb.com database, in addition we can narrow down the results by specifying a genre(s), a writer(s) or a date.

Teamwork

Adrien Moreau 50231580
Scraping script to retrieve the full list of movie titles, writers, scripts, genres and dates of IMSDb.com.

Damien Juan 50231589
Search engine algorithm and python UI.
Redaction of the final report and presentation ppt.

Valentin Sauvier 50231622
Evaluation of results.

Key features

- Search by Word: Users can search for movies containing a specific word in the script.
- Filter by Genre: Users can filter movies by selecting a genre from a dropdown combobox.
- Filter by Writer: Users can filter movies by typing a writer's name, with an autocomplete feature for suggestions.
- Filter by Date Range: Users can filter movies by providing a start and/or end year.
- Open Movie Script Page: Clicking on a movie title in the results opens the corresponding page on IMSDb.com.

I. Data Retrieval

We are getting our data with a python script in order to be able to scrap the entire IMSDb.com film scripts.
In this project we are using several python packages to do so:

- Requests to send http requests and have access to IMSDb.com.
- BeautifulSoup for scraping the website by retrieving the data in the different divs.
- Panda to store and organize the results in a csv file.

How to access to IMSDb.com and access to every script page:

```
def get_movie_links():
   url = 'https://imsdb.com/all-scripts.html'
   response = requests.get(url)
   soup = BeautifulSoup(response.content, 'html.parser')
   links = ['https://imsdb.com' + a['href'] for a in soup.select('p
a[href^="/Movie Scripts/"]')]
   return links
```

How to get the writers:

```
writers = []
   movie_writers_div = soup.find('b', text='Writers')
   if movie_writers_div:
       for sibling in movie_writers_div.find_next_siblings():
           if sibling.name == 'a':
               writers.append(sibling.get_text(strip=True))
           elif sibling.name == 'br':
               continue
           else:
               break
```

How to store the results in a csv file:

```
df = pd.DataFrame(movies_data)
   df.to_csv('imsdb_movie_scripts.csv', index=False, encoding='utf-8')
```

II. Search engine

The movie data is loaded from a CSV file into a Pandas DataFrame. The CSV file contains the following columns:

- Title
- Writers
- Genres
- Script date
- Movie release date
- Script

We are doing basic keyword search functionalities with filtering mechanisms based on specific criteria (such as genre, writer, and date range).

This model can be considered a simplified form of the Boolean retrieval model because it has:

- Exact match: The search for words in the script and genres is based on finding exact matches using regular expressions.

- Implicit AND condition: The combination of different search criteria (word, genre, writer, date range) functions like an implicit AND condition, where all specified conditions must be met for a movie to be included in the results.

```
def search_movies(df, word=None, genre=None, writer=None,
start_year=None, end_year=None):
    result_df = df.copy()

    if word:
        result_df = search_by_word(result_df, word, ['Script',
'Genres'])
    if genre:
        result_df = search_by_word(result_df, genre, ['Genres'])
    if writer:
        result_df = filter_by_writer(result_df, writer)
    if start_year or end_year:
        result_df = filter_by_year(result_df, start_year, end_year)

    return result_df['Title'].unique()
```

For example if you want to have the full list of movies containing the word “Love” in their scripts you can and you will obtain a very big list of movies.

In addition you can select a genre to search the word “love” in the films only in the genre “Romance” to obtain more accurate results.

On top of that you can select a specific writer and a date range:

- if you specify only the start year you will obtain results from your specified date to today
- if you precise only the end year you will obtain the oldest results to your specified date
- if you precise both start year and end year you will obtain results in this date range

You can select multiple keywords, genre or writers but note that they will use the operator AND.

Our search engine proposes more genres than the IMSDb.com filtering system (they do not list them all on their page).

IMSDb.com genre list:

Genre		
Action	Adventure	Animation
Comedy	Crime	Drama
Family	Fantasy	Film-Noir
Horror	Musical	Mystery
Romance	Sci-Fi	Short
Thriller	War	Western

IMSDb.com film where we can see the genre Sport:

Speed Racer Script

IMSDb opinion
Couldn't stand it.

IMSDb rating
(3 out of 10)

Average user rating
(10.00 out of 10)

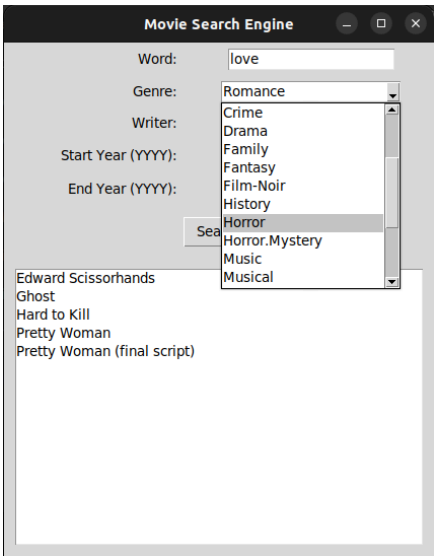
Writers
Larry Wachowski
Andy Wachowski

Genres
Action
Family
Sport

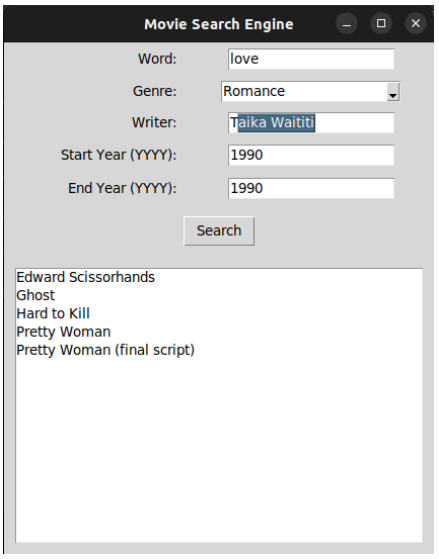
User interface

In order to use the search engine we have made a user interface using Tkinter (Python graphic library), it includes:

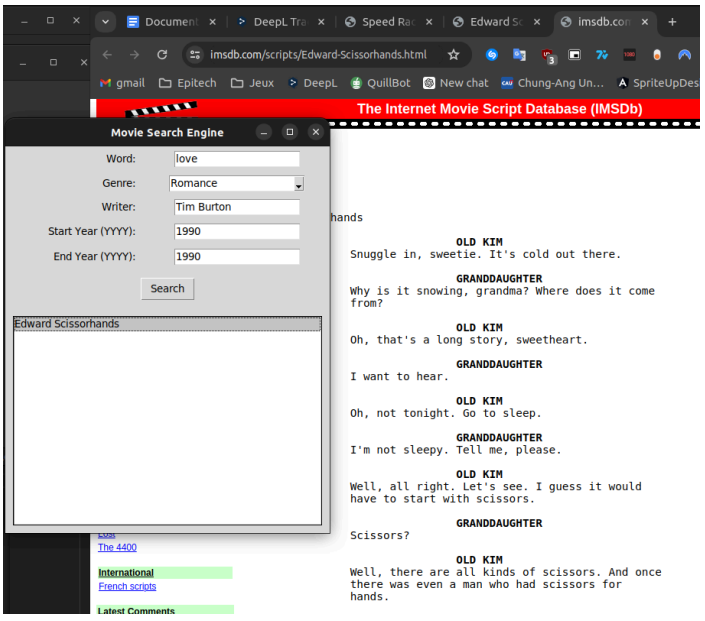
- Entry fields for the word, start year, and end year.
- Comboboxes for genre selection and writer input with autocomplete functionality.
- A listbox to display the search results.
- A search button to trigger the search.
- An event binding to open the corresponding IMSDb.com page when a movie title is selected from the results.



List of all genres



Writer Autosuggest



Access to the film script page by clicking the film title in the list.

III. Evalutation

To assess the effectiveness of our movie search engine, we calculated the precision, recall, and F1 score. These metrics help us understand how well our search engine performs in retrieving relevant results.

- Precision: The ratio of relevant results retrieved to the total results retrieved.
- Recall: The ratio of relevant results retrieved to the total relevant results available.
- F1 Score: The harmonic mean of precision and recall, providing a single measure of a test's accuracy.

Evaluation Method

We conducted a manual evaluation using the following steps:

1. Query: We used the search query “love” within the genre “western”.
2. Results: The search engine returned a list of 15 films.
3. Validation: We manually verified these 15 films by checking the scripts on IMSDb.com to see if they contained the word "love".

Results

All 15 films retrieved by the search engine contained the word "love" in their scripts.

Given these results, we calculated the following metrics:

- Precision: 1.00 (since all retrieved results were relevant)
- Recall: 1.00 (since all relevant results were retrieved)
- F1 Score: 1.00 (since both precision and recall are 1.00)

Running the evaluation script:

```
$ python3 effectiveness.py
Precision: 1.00
Recall: 1.00
F1 Score: 1.00
```

IV. Results Interpretation

The exceptionally high precision and recall rates can be attributed to the following factors:

1. Simple Keyword Search: Our search engine employs a straightforward keyword search combined with filters, effectively implementing a basic Boolean retrieval model. This model ensures that all results containing the keyword in the specified context are retrieved, leading to high precision and recall.
2. Comprehensive Filtering: The use of genre and keyword filters ensures that only relevant results are considered, further enhancing accuracy.

However, our search engine does have limitations:

- Incomplete Data: Some scripts on IMSDb.com may be incomplete, missing essential information such as writers or genres. These incomplete records are not retrieved by our search engine, as they do not meet the filter criteria. This can lead to missed relevant results.

Conclusion

While our movie search engine demonstrates perfect precision and recall in this specific evaluation, its performance is inherently tied to the completeness and accuracy of the underlying data. Incomplete data on IMSDb.com can limit the effectiveness of our search engine, highlighting the importance of comprehensive and accurate data sources.