Topic: IMDb Actors Search Based on Movie Topics
Team: Chris Nolan's Fans Club
Members: Anasthasia Amelia (aas13), Aravind Pillai (pillai5), Mike Zhou (mikez2)

Goal:
The topic is scraping the IMDb website to capture the types of films individual actors act in and allow searching for actors by arbitrary topics and showing a list of movies matching those topics.

IMDb Actors Search Application Link:
http://54.176.35.246:8080/

Source code:
https://github.com/amelia2801/CourseProject

How to install and run the application:
https://github.com/amelia2801/CourseProject/blob/main/actor_search_engine/Readme.md

Video presentation:
https://drive.google.com/file/d/1P6hGsJ0Lwrk1OmJexMmp0VXBKdpC-GhF/view?usp=sharing

Data Preparation

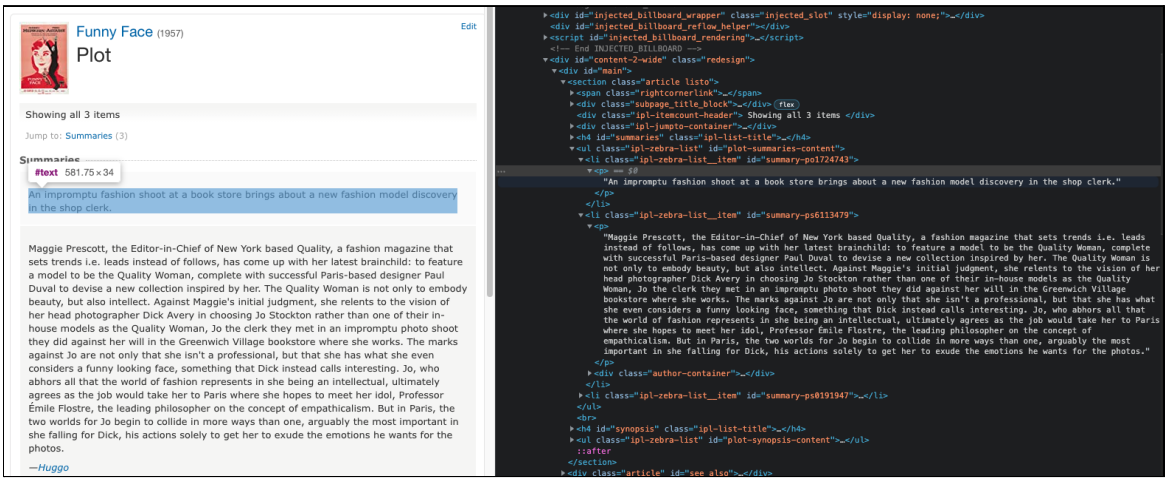
There are 4 main steps in preparing the data:

1. Process all the available data from IMDb
   There are 5 files downloaded from IMDb: *name.basics.tsv, title.principals.tsv, title.akas.tsv, title.basics.tsv and title.ratings.tsv*
   In this process, all the data taken from the above source are filtered to make the datasets more relevant. The filters applied to the data are:
   - actors must be born after year 1940
   - actors must at least be associated with 4 movies
   - each titles should be in 'movie' type
   - region must be US
   - movie should have received at least 2000 user votes

   After the necessary processes are applied to the data, the output is exported into a CSV file that can be found in `*../files/actorsOrig.csv*' which contains *nconst* (actorId), *primaryName* (actor's name), and *tconst* (movieId).
   For more details, read the code in
   `actor_search_engine/scripts/0_prepareMovieActorsFile.py`

2. Scrape each movie plot summary
   Based on the processed data in *actorsOrig.csv*, scrape all the plot summaries for each unique movie titleId (*tconst*). Scraping is done by using the BeautifulSoup library.
   URL: *https://www.imdb.com/title/<tconst>/plotsummary*
   Each scraped movie plot summary is exported to a `.txt` file that is stored in

`../movieFile/` folder. Actors data then filtered out to only those that have corresponding plot summary, then exported to CSV file *`../files/actors.csv`*.
For more details, read the code in
`actor_search_engine/scripts/1_ScrapeMoviesActors.py`



3. Scrape keywords related to each movie title
   Similar to step 2, scrape the related keywords for each movie title. This data is available from the IMDb website.
   URL: *https://www.imdb.com/title/<tconst>/keywords*
   Scraped keywords for each movie are exported to a `.txt` file that is stored in `../movieFileTags/` folder.
   For more details, read the code in
   `actor_search_engine/scripts/2_ScrapeMoviesTags.py`

4. Build corpus
   Build corpus, a file that contains all documents. Each document is represented in one line. It contains information of *`tconst`* (titleId) and *`mov_text`* (movie plot summary).
   In this step, we generate 2 files:
   - `corpusList.csv`:
     This file contains data of titleId and a short plot summary (like a preview), that we will use to display in the search results.
   - `corpus.dat`:
     This is the corpus that we use to build indexes and run queries on.

   For more details, read the code in
   `actor_search_engine/scripts/4_buildCorpus.py`

Building Search Index

The program uses the metapy function `make_inverted_index` to build the inverted index. A configuration file `config.toml` is used when calling this function. The built index will be stored in a folder called `idx`, as mentioned in the configuration file. If such an index already exists, metapy will load that index.
The inverted index is created using certain features that are specified in the configuration file `config.toml`. A combination of unigram, bigram, trigram, and bigram part-of-speech tags features are used, which are defined in each `[[analyzers]]` block.
Code: `/actor_search_engine/buildSearchIndex.py`
Configuration file: `/actor_search_engine/config.toml`

Running Queries & Display Results

Code: `/actor_search_engine/movieApp.py`

The program accepts a query that the user types. The query is passed to the ranker scoring function. The ranker used in this program is OkapiBM25 with parameters k1=1.2, b=0.75, and k3=500. The scoring function will return a search result that contains the index of the data and its corresponding score.

The program uses a file `docList_df.csv` that contains a list of `tconst` (titleId) which are sorted in the same way as the corpus. It uses this list to map the document score with its corresponding `tconst`.

	tconst	doc_scores
0	tt0317705	7.301030
1	tt1001526	6.960334
2	tt0465624	6.935660
3	tt0467110	6.919347
4	tt0448157	6.848383
..	...	...
95	tt0114614	4.862828
96	tt0478970	4.856150
97	tt0100758	4.850182
98	tt0800369	4.827347
99	tt0262432	4.810512

The program then uses the data in `corpusList.csv` to join its data and the mapping produced in the previous step, such as the `nconst` (movieId), `primaryName` (actor's name), and `mov_text` (short plot summary). And then, movie link and actor link are also added for each result. `cum_score` is the cumulative score for an actor that has returned more than one movie in the results. `cum_rank` is the rank of movies for an actor (one actor might be involved in more than one movie).

In the image below, Amy Adams is returned in more than one movie: Underdog (2007), Batman v Superman: Dawn of Justice (2016), Zack Snyder's Justice League (2021), and Man of Steel (2013). Each of this movie plot summary has its corresponding document score. `cum_score` is the sum of all those document scores for movies that Amy Adams acted in, while `cum_rank` is the rank for each of her movies in this list.

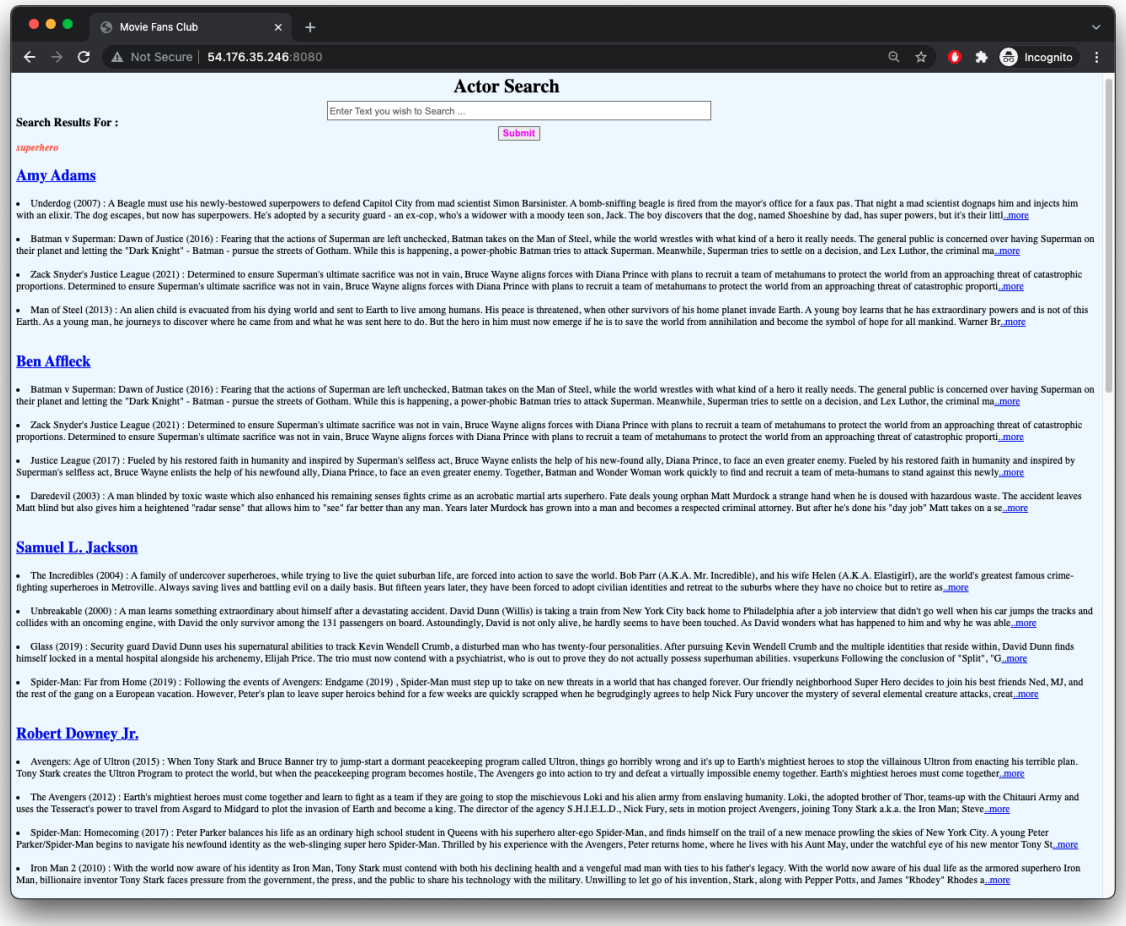
	index	nconst	primaryName	tconst	doc_scores	mov_text	cum_rank	cum_score
0	191	nm0010736	Amy Adams	tt0467110	6.919347	Underdog (2007) : A Beagle must use his newly-...	1.0	24.735582
1	115	nm0010736	Amy Adams	tt2975590	6.596645	Batman v Superman: Dawn of Justice (2016) : Fe...	2.0	24.735582
2	111	nm0010736	Amy Adams	tt12361974	6.216801	Zack Snyder's Justice League (2021) : Determin...	3.0	24.735582
3	64	nm0010736	Amy Adams	tt0770828	5.002789	Man of Steel (2013) : An alien child is evacua...	4.0	24.735582
4	114	nm0000255	Ben Affleck	tt2975590	6.596645	Batman v Superman: Dawn of Justice (2016) : Fe...	1.0	24.618237
5	110	nm0000255	Ben Affleck	tt12361974	6.216801	Zack Snyder's Justice League (2021) : Determin...	2.0	24.618237
6	106	nm0000255	Ben Affleck	tt0974015	6.200866	Justice League (2017) : Fueled by his restored...	3.0	24.618237
7	102	nm0000255	Ben Affleck	tt0287978	5.603925	Daredevil (2003) : A man blinded by toxic wast...	4.0	24.618237
8	47	nm0000168	Samuel L. Jackson	tt0317705	7.301030	The Incredibles (2004) : A family of undercove...	1.0	24.152946
9	43	nm0000168	Samuel L. Jackson	tt0217869	5.853653	Unbreakable (2000) : A man learns something ex...	2.0	24.152946
10	55	nm0000168	Samuel L. Jackson	tt6823368	5.768854	Glass (2019) : Security guard David Dunn uses ...	3.0	24.152946
11	51	nm0000168	Samuel L. Jackson	tt6320628	5.229409	Spider-Man: Far from Home (2019) : Following t...	4.0	24.152946
12	147	nm0000375	Robert Downey Jr.	tt2395427	6.436053	Avengers: Age of Ultron (2015) : When Tony Sta...	1.0	22.826375
13	139	nm0000375	Robert Downey Jr.	tt0848228	5.873378	The Avengers (2012) : Earth's mightiest heroes...	2.0	22.826375
14	143	nm0000375	Robert Downey Jr.	tt2250912	5.313183	Spider-Man: Homecoming (2017) : Peter Parker b...	3.0	22.826375
15	124	nm0000375	Robert Downey Jr.	tt1228705	5.203762	Iron Man 2 (2010) : With the world now aware o...	4.0	22.826375
16	148	nm0262635	Chris Evans	tt2395427	6.436053	Avengers: Age of Ultron (2015) : When Tony Sta...	1.0	22.765401
17	140	nm0262635	Chris Evans	tt0848228	5.873378	The Avengers (2012) : Earth's mightiest heroes...	2.0	22.765401
18	236	nm0262635	Chris Evans	tt0120667	5.345186	Fantastic Four (I) : 55) A group of astronauts...	3.0	22.765401
19	240	nm0262635	Chris Evans	tt0486576	5.110785	Fantastic 4: Rise of the Silver Surfer (2007) ...	4.0	22.765401

(This is a preview of 20 first data)

Then it returns the top 12 results:

	primaryName	mov_text	actorLink	movLink
0	Amy Adams	[Underdog (2007) : A Beagle must use his newly...	https://www.imdb.com/name/nm0010736	[https://www.imdb.com/title/tt0467110/plotsumm...
1	Ben Affleck	[Batman v Superman: Dawn of Justice (2016) : F...	https://www.imdb.com/name/nm0000255	[https://www.imdb.com/title/tt2975590/plotsumm...
2	Samuel L. Jackson	[The Incredibles (2004) : A family of undercov...	https://www.imdb.com/name/nm0000168	[https://www.imdb.com/title/tt0317705/plotsumm...
3	Robert Downey Jr.	[Avengers: Age of Ultron (2015) : When Tony St...	https://www.imdb.com/name/nm0000375	[https://www.imdb.com/title/tt2395427/plotsumm...
4	Chris Evans	[Avengers: Age of Ultron (2015) : When Tony St...	https://www.imdb.com/name/nm0262635	[https://www.imdb.com/title/tt2395427/plotsumm...
5	Henry Cavill	[Batman v Superman: Dawn of Justice (2016) : F...	https://www.imdb.com/name/nm0147147	[https://www.imdb.com/title/tt2975590/plotsumm...
6	Elias Koteas	[Defendor (2009) : Arthur Poppington, a regula...	https://www.imdb.com/name/nm0000480	[https://www.imdb.com/title/tt1303828/plotsumm...
7	Christopher Reeve	[Superman II (1980) : Superman agrees to sacri...	https://www.imdb.com/name/nm0001659	[https://www.imdb.com/title/tt0081573/plotsumm...
8	Margot Kidder	[Superman II (1980) : Superman agrees to sacri...	https://www.imdb.com/name/nm0452288	[https://www.imdb.com/title/tt0081573/plotsumm...
9	Chris Hemsworth	[Avengers: Age of Ultron (2015) : When Tony St...	https://www.imdb.com/name/nm1165110	[https://www.imdb.com/title/tt2395427/plotsumm...
10	Michael Keaton	[Birdman or (The Unexpected Virtue of Ignoranc...	https://www.imdb.com/name/nm0000474	[https://www.imdb.com/title/tt2562232/plotsumm...
11	Jason Lee	[The Incredibles (2004) : A family of undercov...	https://www.imdb.com/name/nm0005134	[https://www.imdb.com/title/tt0317705/plotsumm...

Finally, the results are displayed in a web application. Here is a screenshot of search results for query ‘superhero’. It displays actors that are related to ‘superhero’ and their corresponding movies.



Workload & Contribution

- Environment setup (aas13, pillai5, mikez2)
- Explore the IMDb datasets (aas13, pillai5, mikez2)
- Discussion on how to structure the datasets (aas13, pillai5, mikez2)
- Initial data preparation & scrape movie plot summary (aas13)
- Scrape keywords for movies & build corpus (mikez2)
- Main program & user interface (pillai5)
- EC2 instance setup (pillai5)
- Testing (aas13, pillai5, mikez2)