The Entertainment Knowledge Graph

An IR project based on Knowledge Graph Structures for movies and actors/actresses analysis.

Group Members

1. Abhinav Talari (S20180010003)
2. Utkarsh Ajay Aditya (S20180010182)

Project Details

Project Title: The Entertainment Knowledge Graph

Domain of Project: Entertainment (Movies, television programs and celebrity content

Project Description

This project focuses on end-to-end utilization of knowledge graphs and semantics in the field of entertainment. Our entities include movies, tv shows, actors/actresses and all their attributes and relationships. Our project is based on entity ranking, entity linking, entity-based retrieval models, entity recommendation, document filtering and knowledge graph population. The knowledge graph will represent movies and tv shows as well as actors, genres and the complex interrelationships among them, which will help us retrieve more informational value compared to free text.

Algorithms/Approaches Used

Individual Breakdown for each member.

1. Abhinav Talari(S20180010003) Tasks

A. Web Scraping(Actors/Actresses data)

Associated Files: Scraper_For_Actors.py

This python code file is used for scraping actor data from Wikipedia and storing it.
Scraped Data from Wikipedia by Even going through the underlying links which increased the corpus exhaustively

The Functions Used :
1. wiki_scrape(page)
This Looks for a page on Wikipedia
extracts it,parses it and the underlying link and keeps on fetching data from those links.

For Example for a query as wiki_scrape("Robert Downey Jr.") Made it crawl 800+ links

2. wiki_page(page)
This just fetches the data from the very first hit it gets on Wikipedia
the query for this function needs to be very precise

For Example wiki_page("The Avengers") will not fetch the movie synopsis of the MCU's Avengers but just links and names of other movies/entities with the same name or context.

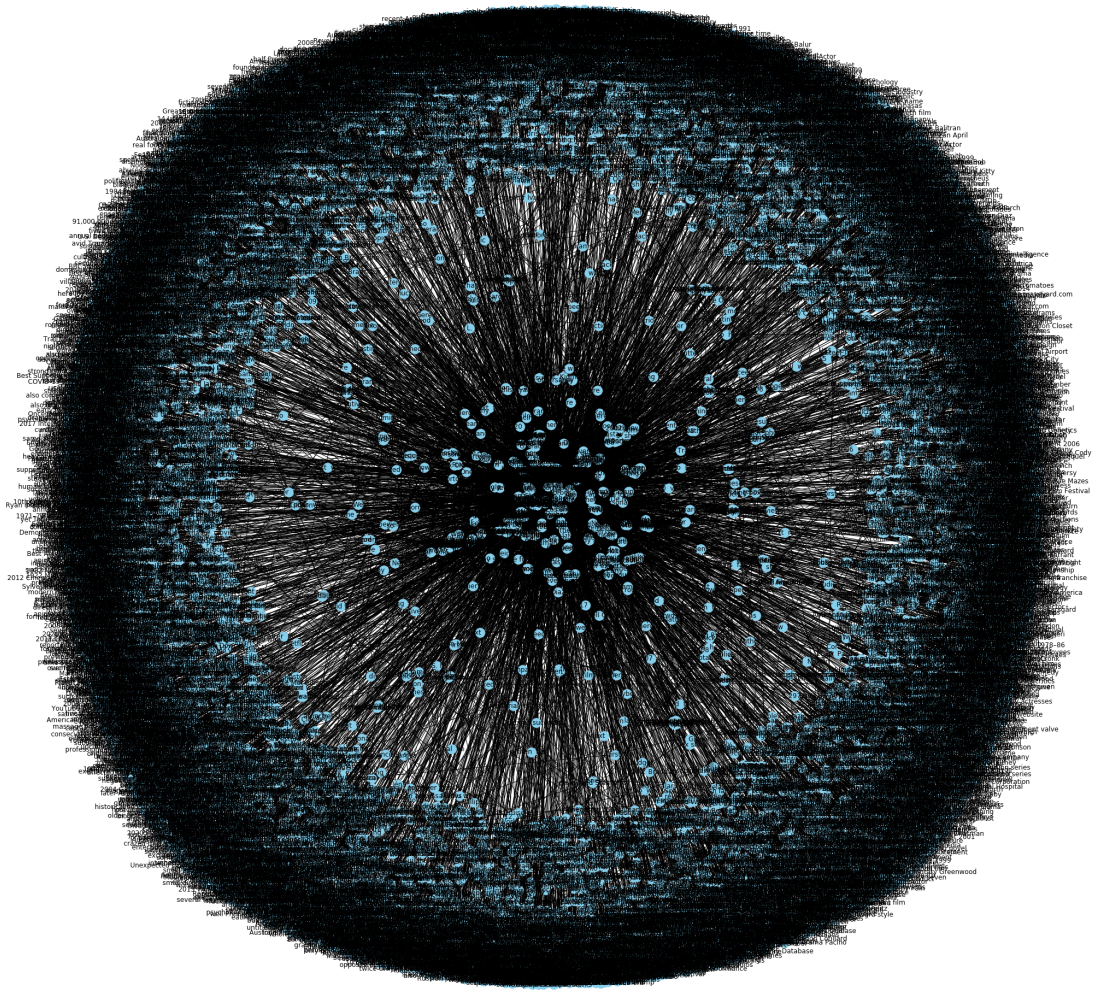
B. Knowledge Graph for Actors

Associated Files: Knowledge_Graph_for_Actors.py

Based on the data crawled and collected for Actors and Movies , I created a knowledge graph to perform "Entity-Entity pair based on Relation" Query.

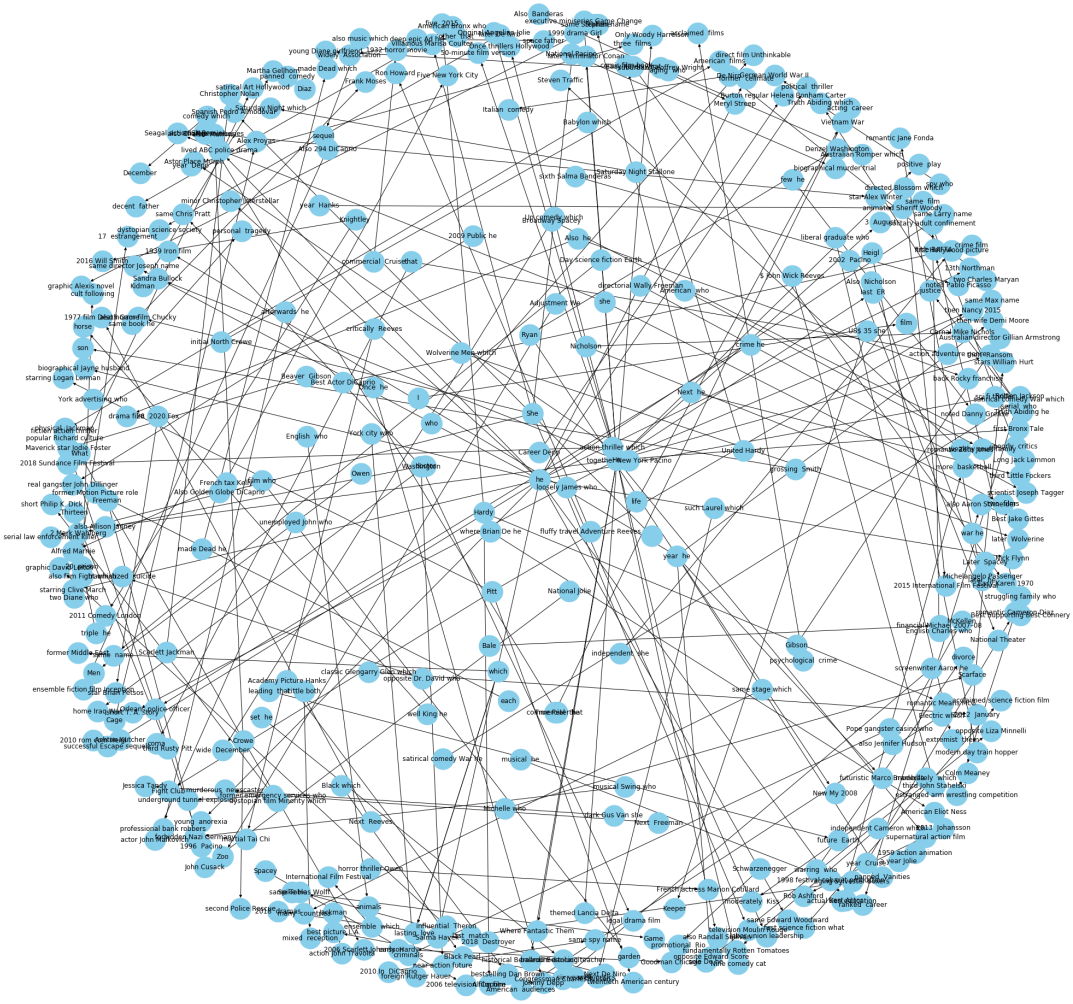
The sentences were tokenized and the Entity-Relation-Entity were identified and put into the Knowledge Graph
The knowledge graph was visualized using Networkx

Output:Below is the knowledge graph representing top 250 actors along with their extracted entities (subject-object pairs) and the predicates (relation between entities).



The networkx library was used to create a network from this data frame. The nodes will represent the entities and the edges or connections between the nodes will represent the relations between the nodes.

It is going to be a directed graph. In other words, the relation between any connected node pair is not two-way, it is only from one node to another. For example, “John eats pasta”



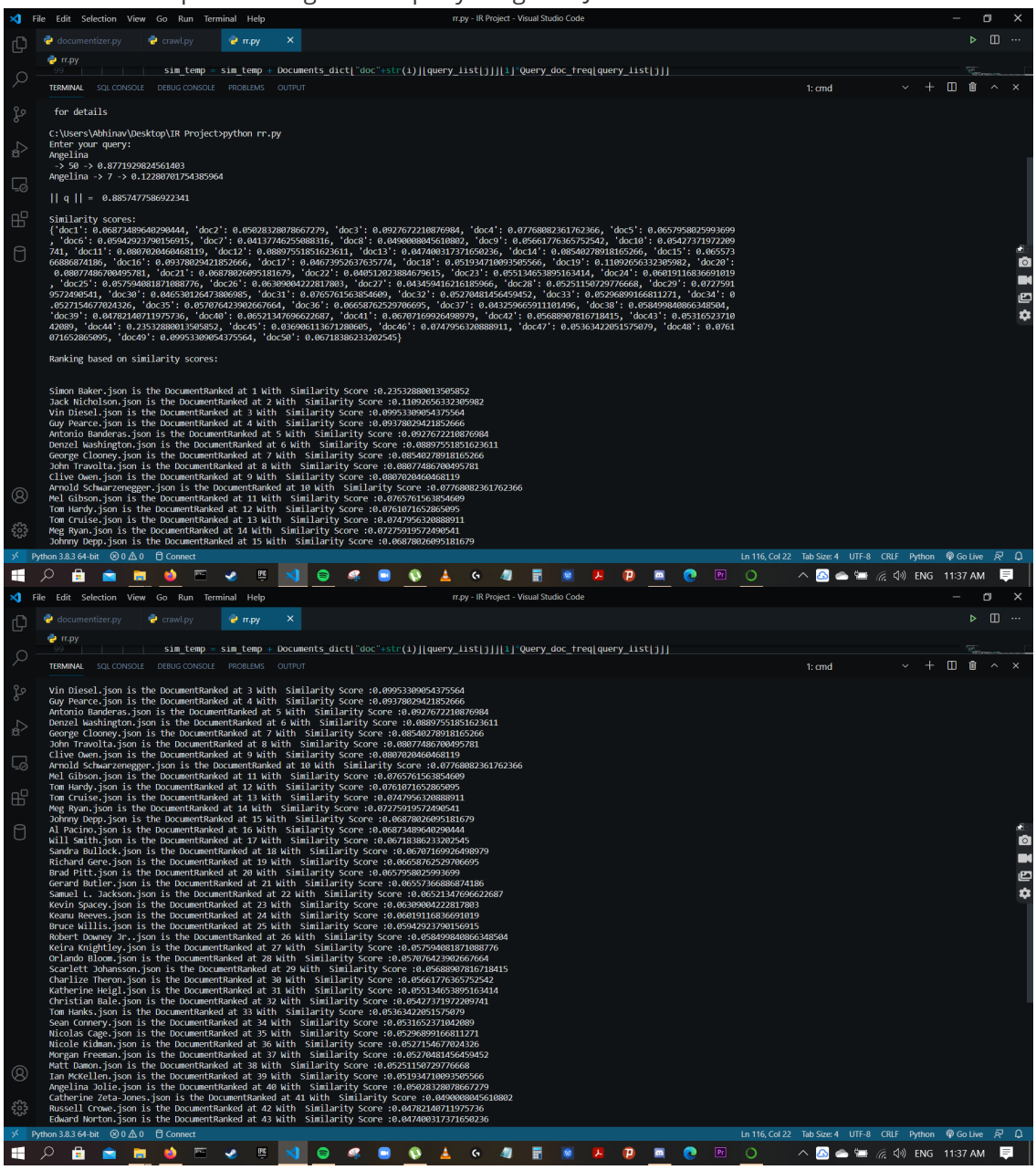
The above figure represents a sub-knowledge graph based on the relation "starred in" for actors.

C. Ranked Retrieval model(Actors)

Associated Files: RankedRetrieval_actors.py
Based on the Data Scraped I created a Ranked Retrieval System using tf-idf ranking.
Since the Boolean Model only fetches complete matches, it doesn’t address the problem of the documents being partially matched. The Vector Space Model solves this problem by introducing vectors of index items each assigned with weights. The weights are ranged from positive (if matched completely or to some extent) to negative (if unmatched or completely oppositely matched) if documents are present. Term Frequency - Inverse Document Frequency (tf-idf) is one of the most popular techniques where weights are terms (e.g. words, keywords, phrases etc.) and dimensions is number of words inside corpus.
Therefore I created a ranked retrieval based on tf-idf ranking

Output

1. Below is the output showing for the query "Angelina Jolie"



2.Utkarsh Ajay Aditya(S20180010182) Tasks

A. Web Scraping(Movies)

Associated Files: imdb_moviedata_scrape.py

The above code is used for scraping movie details of over 83,000 movies and storing it in final_dataset.csv

The collected fields for a given movie were:

Title, Date, Run Time, Genre, Rating Score, Description, Director, Stars, VotesGross

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	0	title	original_title	year	date_publis	genre	duration	country	language	director	writer	production_	actors	description	avg_vote	votes
2	1	Miss Jerry	Miss Jerry	1894	1894-10-09	Romance	45	USA	None	Alexander B	Alexander B	Alexander B	Blanche Bayliss, Will	The adventures of a fema	5.9	154
3	2	The Story of the Kelly Gang	The Story of the Ke	1906	26-12-1906	Biography, Crime, I	70	Australia	None	Charles Tait	Charles Tait	J. and N. Ta	Elizabeth Tait, John	True story of notorious A	6.1	589
4	3	Den sorte drøm	Den sorte drøm	1911	19-08-1911	Drama	53	Germany, Denmark		Urban Gad	Urban Gad,	Fotorama	Asta Nielsen, Valden	Two men of high rank are	5.8	188
5	4	Cleopatra	Cleopatra	1912	13-11-1912	Drama, History	100	USA	English	Charles L. G	Victorien Sa	Helen Gardr	Helen Gardner, Pear	The fabled queen of Egyp	5.2	446
6	5	L'Inferno	L'Inferno	1911	06-03-1911	Adventure, Drama,	68	Italy	Italian	Francesco B	Dante Alighi	Milano Film	Salvatore Papa, Artu	Loosely adapted from Da	7	2237

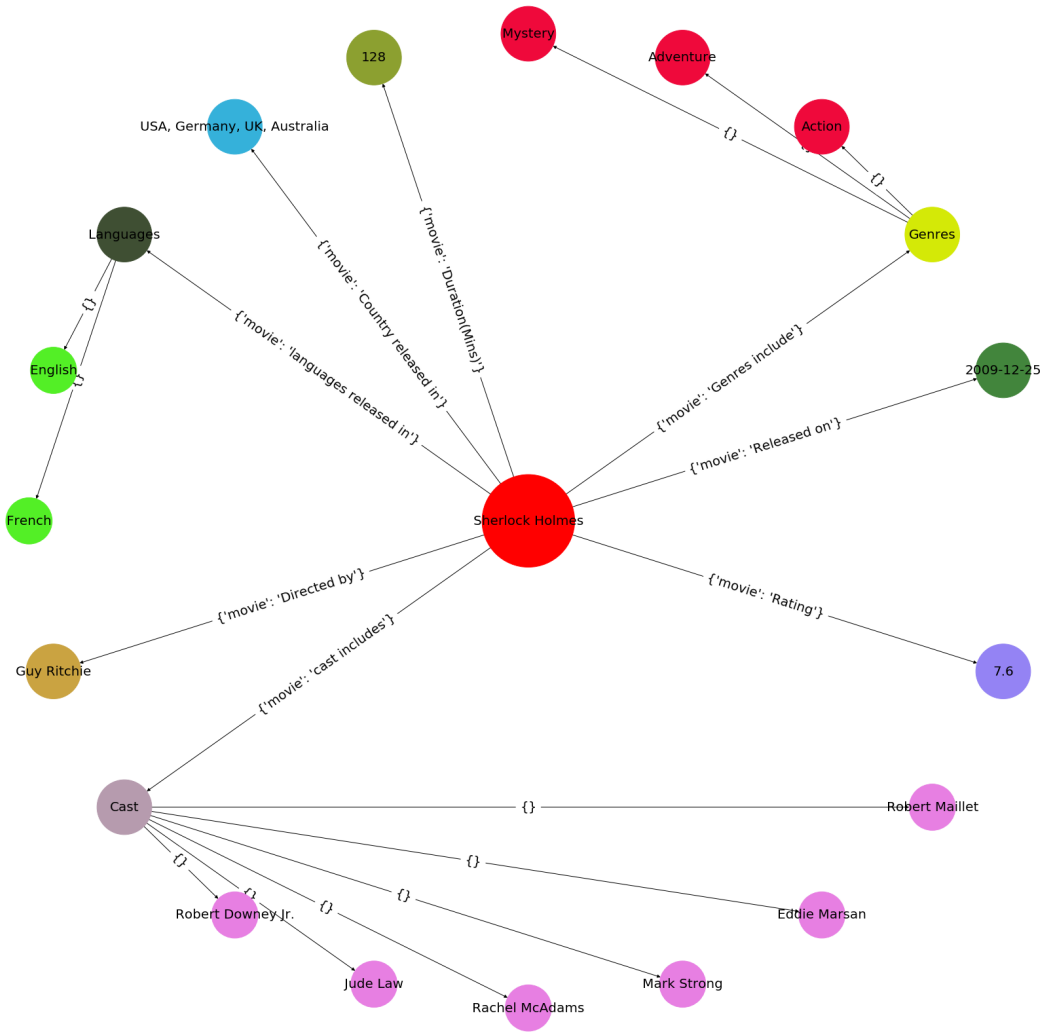
B. Knowledge Graph for Movies

Associated Files: KnowledgeGraph_movies_recommender_system.py

It works on the basis of final_dataset_imdb.csv.

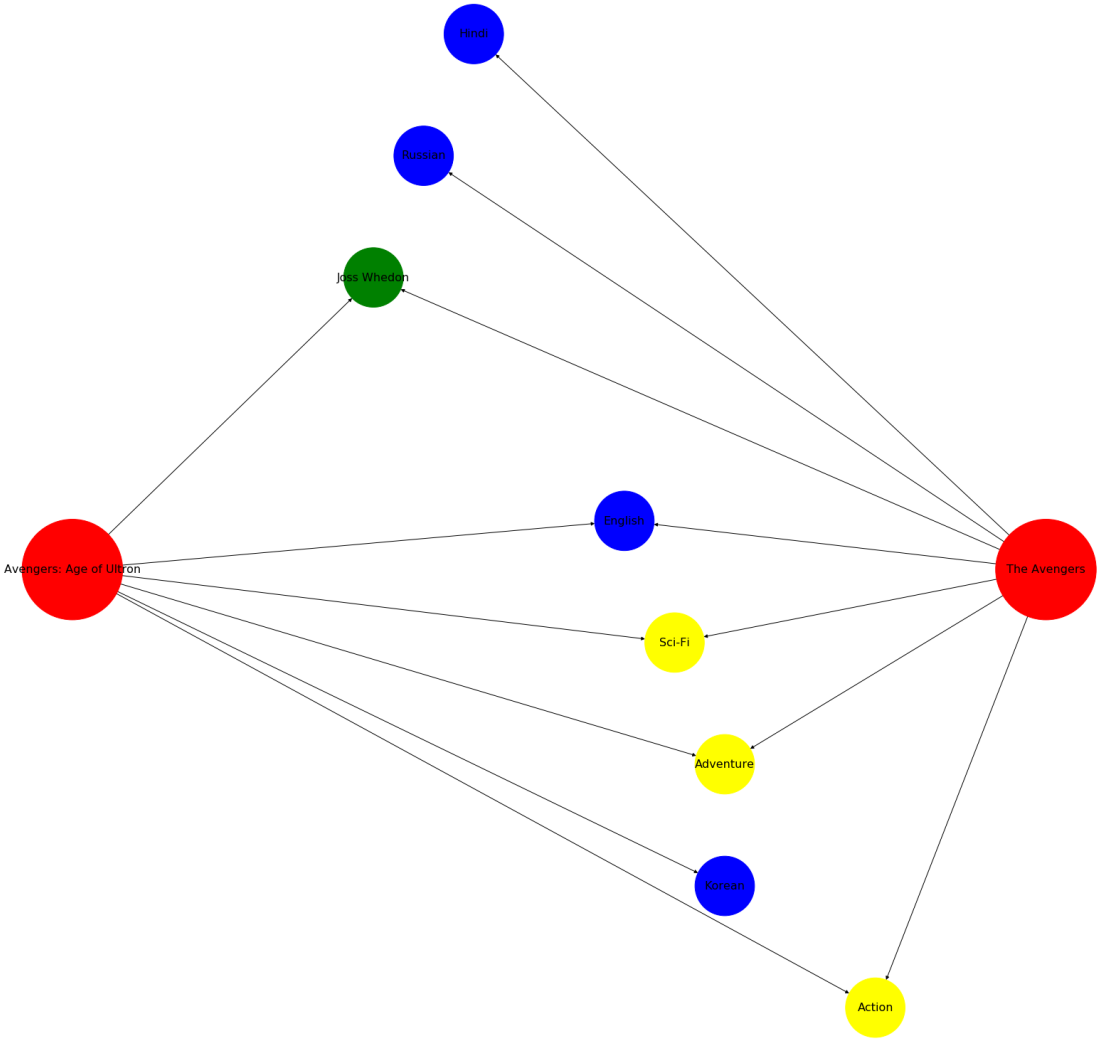
NOTE : All the graphs would be saved as .pdf file in the code directory. One has to view the pdf file where he can zoom and every node will be visible in HD resolution.
The above code file defines 3 different types of knowledge graph structures related to movies:

1. Movie detail graph for a single movie



The above figure shows the detailed view of a given movie name showing all its details like director, cast, release date, etc.

2. Movie Comparison graph between two movies.



The above figure shows the comparison between 2 input movie names on basis of genre, languages and director.

3.Knowledge graph of 1000 movies.

The blue nodes near the edges of the graph represent genres, edges of same color represent same genre movies.

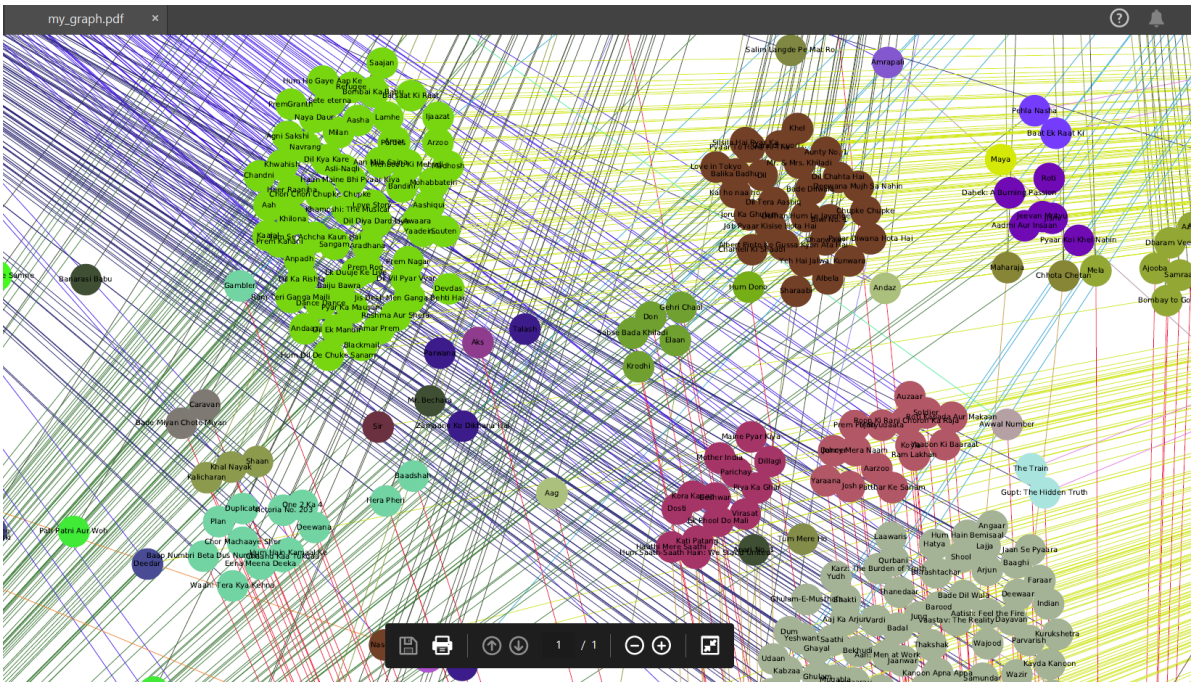
A movie node's color is derived by the combination of its genres color. Therefore movies with similar characteristics would be of same color and in the same color.

Properties of graph :

a. There is homogeneity between the movies belonging to the same color. (Represented by same color.)

b. There is heterogeneity among movies from different clusters. (Represented by different colors.)

Zoomed pic of a cluster.



Here you can see the movies which belong to different colors. Each color represents a color.

Such a graph provides us insight about what movie an user can like based on few of his judgments.

C. Movie Recommendation System

Associated Files: KnowledgeGraph_movies_recommender_system.py

Given a movie name it will return a list of top 10 similar movies along with their matching scores.

It is based on an empirically derived ranking system which takes into account all the movie details like genres, crew,

director, writer, year, other user reviews, etc.

Example : For movie - The Avengers

```
(base) PS C:\Users\utkar\Desktop\Untitled Folder> python '.\KnowledgeGraph (1).py'
1. Enter 1 for movie detail graph
2. Enter 2 for movie similarity graph between 2 movies.
3. Enter 3 for knowledge graph of 1000 movies.
4. Enter 4 for Movie Recommendations(Top 10 along with score)
5. Enter 0 to exit
4
Enter Movie Name(Case Sensitive): The Avengers
Rank  1 ==>  Avengers: Age of Ultron
Score:  65.00000000722684

Rank  2 ==>  Avengers: Infinity War
Score:  54.00000000796486

Rank  3 ==>  Iron Man 2
Score:  48.00000000706284

Rank  4 ==>  Captain America: Civil War
Score:  48.00000000644241

Rank  5 ==>  Avengers: Endgame
Score:  41.00000000754786

Rank  6 ==>  Iron Man
Score:  39.00000000920706

Rank  7 ==>  Captain America: The Winter Soldier
Score:  39.00000000719085

Rank  8 ==>  Iron Man Three
Score:  38.00000000739816

Rank  9 ==>  Captain America: The First Avenger
Score:  31.00000000723473

Rank  10 ==>  Spider-Man: Homecoming
Score:  31.0000000049406


Best movie match is : Avengers: Age of Ultron
```

Example : For movie mission impossible

```
Enter Movie Name(Case Sensitive): Mission: Impossible
Rank  1 ==>  Mission: Impossible III
Score:  34.0000000032431

Rank  2 ==>  Mission: Impossible - Fallout
Score:  34.00000000280981

Rank  3 ==>  Mission: Impossible - Rogue Nation
Score:  33.0000000033453

Rank  4 ==>  Mission: Impossible II
Score:  33.00000000308586

Rank  5 ==>  Mission: Impossible - Ghost Protocol
Score:  30.00000000449833

Rank  6 ==>  The A-Team
Score:  27.00000000243142

Rank  7 ==>  Largo Winch
Score:  27.00000000013296

Rank  8 ==>  The Hunt for Red October
Score:  26.00000000175033

Rank  9 ==>  Drop Zone
Score:  26.00000000018953

Rank  10 ==>  The Rock
Score:  25.00000000307112


Best movie match is : Mission: Impossible III
```

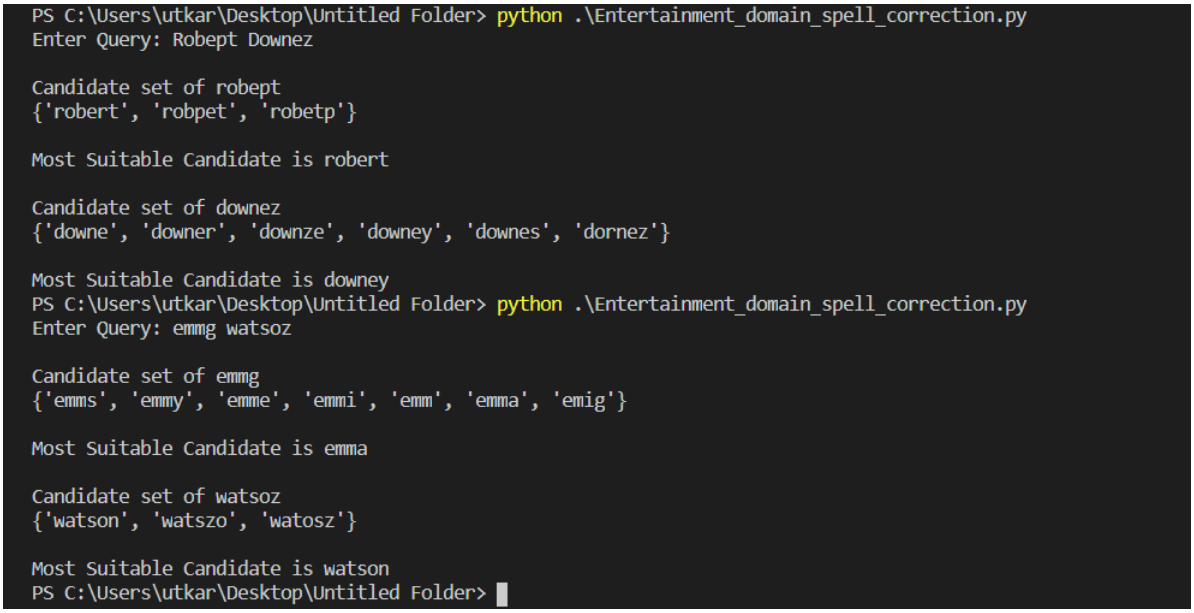
D. Spell Correction for the entertainment domain

Associated Files: KnowledgeGraph_movies_recommender_system.py

Entertainment_domain_spell_correction.py is an autocorrect program file to correct incorrect

a) movie name or TV series name
b) actors/actresses name
c) director/write name

Example For query "Robept Downez" and another query "emmg watsoz"



Implementation of spell checker

1) Store Dictionary in Trie
2) Query is processed word by word.
3) If word is in dictionary, move ahead.
4) If word is not in dictionary, display the candidate set.
5) Also display the post probable candidate by calculating the posterior probability

References

1. Wikipedia
2. IMDB
3. networkx
4. **Research Paper** : Knowledge Graphs: An Information Retrieval Perspective