Popular Movies and Streaming

Carlos Gomes, Eduardo Silva, Joana Silva and Joana Ramos
MIEIC, FEUP, U.Porto
Porto, Portugal

***Abstract*—Information retrieval technologies have had a tremendous growth over the past decades, driven primarily by the large document collections that are constantly becoming available through the Internet. On the other hand, Semantic Web provides a common framework that enables data to be shared and used in various types of applications. Inspired by these systems, it was proposed to develop both an information retrieval platform and an ontology centered on the movie and acting universe, built on top of several document collections and data sets, aiming to fulfill a user's information needs on related topics. The information retrieval system, although with some limitations due to the bottleneck strategy used to reduced the datasets, allowed to fulfill information needs with acceptable precision levels. On the other hand, Protegé, which was used to develop an appropriate ontology, had considerable limitations that did not allow the use of a significant number of examples. For that reason, the developed system is more of a proof of concept, deeming the ontology as effective and expressive enough for the proposed domain.**

I. Introduction

In recent decades, the Web has made available vast amounts of information to the common user. Search Engines like Bing, DuckDuckGo and Google allow the intuitive search of these document collections with an incredible speed and accuracy. Inspired by these developments, this paper describes the creation of an information retrieval (IR) system for movies, its details, crew and current streaming platforms. Other essentials tasks for similar systems are the description, organization and the establishment of relations between data artifacts. RDF and OWL are formal ways to do these tasks, allowing to connect web data and standardise structures for it. As such, this document also describes a movie-related ontology created for these purposes. The present document is structured as follows: in chapter II the dataset used is presented as well as the pre-processing operations performed on it. Part III describes the IR system built. Part IV analyses the creation of an ontology capable of describing our domain and possibly others, with the detailed description of each performed step. The results and technology evaluation of both the IR system and the ontology are described in the respective sections.

II. The Dataset

A. Obtaining and Preparing the Datasets

The data that will be used is gathered from 4 different datasets. Further details about them are presented below.

1) IMDb Official Dataset: The IMDb official dataset [1] is a **structured dataset** in TSV format that contains all of the information regarding movies and TV series that can be found on IMDb's website. The dataset is composed of 7 different files which together add up to a **volume** of 4.6 GB, a size that increases daily as the files are updated. All of the 7 files present a very similar **structure** but contain different relevant information:

- **title.akas.tsv.gz** file - contains important information concerning titles, such as a movie's region, title and language;
- **title.basics.tsv.gz** file - has relevant information such as the release year of a movie, its run-time in minutes, the genres associated with the film and its original title;
- **title.crew.tsv.gz** file - this file contains the details about a movie's writers and directors;
- **title.principals.tsv.gz** file - contains a movie's main cast and crew details, such as the category of the job being executed, the specific job title and the characters played in case of being an actor;
- **title.ratings.tsv.gz** file - has IMDb's votes and rating information for titles, containing the number of votes and the average rating;
- **name.basics.tsv.gz** file - significant information about each person, for instance the name for which the person is most credited, birth year and death year (when applicable), its top three professions and the titles for which is most known;
- **title.episode.tsv.gz** file - this file will not be used, as it consists of information related to TV series.

In what concerns the **licence** of the dataset, IMDb, as the **source authority**, provides a limited non-commercial use of their data if all of the stipulated conditions [2] are met. Additionally, a commercial use licence can also be obtained [3] , if the need arises.

2) IMDb Scraped Dataset: This dataset [4] is a **structured dataset** with movie information retrieved through scraping of IMBb's website [5]. The dataset comprises all the movies with more than 100 votes as of 01/01/2020, amounting to 85,855 movies. The information is represented in 4 CSV (Comma Separated Values) files in UTF-8 encoding, which total 230 MB. This dataset is obtained from Kaggle [6] under the CC0: Public Domain **license**. Since the **authority** of this data repository might be questionable, this dataset is complementary and it's correctness can be assessed. It contains the following information:

- **IMDb movies.csv** file - contains movie related information like the title, genre, language, budget, director and actors. This file contains the movie's IMDb ID;
- **IMDb names.csv** file - has information regarding people

involved in movies (directors, writers, actors etc), namely their name, bio, birth date and other personal details;

- **IMDb ratings.csv** file - a collection of ratings for each movie, including demographic information (e.g. percentage of female voters);
- **IMDb title_principals.csv** file - associations between the movies and the names files

3) Streaming Dataset: The Streaming Dataset [7] contains information about 16744 movies from 4 different platforms: Netflix, PrimeVideo, Hulu and Disney+. The data was last updated on May 2020. This dataset is **structured** and extracted from Kaggle [6] under the **license** CC0: Public Domain. The principal characteristics of the dataset are:

- **Title (string)**: Title of the movie;
- **Year (int)**: Year when movie was released;
- **Netflix (int)**: 1 if the movie is available in Netflix, 0 otherwise;
- **Hulu (int)**: 1 if the movie is available in Hulu, 0 otherwise;
- **PrimeVideo (int)**: 1 if the movie is available in Prime Video, 0 otherwise;
- **Disney+ (int)**: 1 if the movie is available in Disney+, 0 otherwise;

However, due to inconsistencies in certain attributes, in relation to the other datasets, and the lack of an IMDb ID, only 15531 of the available titles are used in the following stages.

4) IMDb Movie Pages: The information regarding the synopses of the movies is directly scraped from the respective IMDb [5] pages. This data is composed of free text and is **unstructured** by nature, with the total volume being limited by that of the other datasets.

5) Data Pipeline Process: The data pipeline is illustrated in Figure 9. The matching between all the datasets is done through a common IMDb IDs. All of the datasets posses this attribute except for the streaming one. In this case, a new column `imdb_id` is added with this value. The ID's are obtained through a match between titles, year of release and the IMDb classification (`short`, `movie`, `tvmovie`, `tvshort` or `video`). It is also worth noting that due to the smaller volume of this particular dataset, it acts as a bottleneck for the other 3.

After all the data is matched and aggregated, cleaning operations are performed, namely duplicate value removal and title and date normalization.

Finally, the data is then stored in an SQL relational database, ready for further exploration.

B. Datasets Characterization

The number of movies per streaming platform is represented in Figure 1. Clearly, Prime Video dominates the dataset, with Netflix coming in second while Hulu and Disney+ are more or less equivalent.

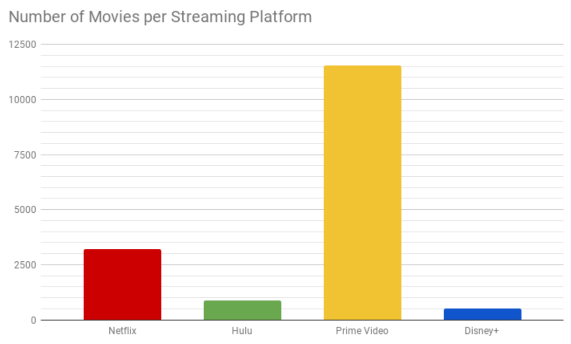


Fig. 1. Number of Movies per Streaming Platform

Figure 10 illustrates the distribution of movies on streaming platforms, with relation to the movie's release year. It's interesting to notice that while Prime Video [8] has a bigger movie collection overall, Netflix [9] hosts more recently released movies. In spite of this, we should keep in mind that this is a sample, and might not be representative of the whole set.

The average movie rating by year can be seen in figure 11, with the ratings from IMDb [5] and Rotten Tomatoes [10] represented. We can verify that in Rotten Tomatoes the rating is almost always lower.

Regarding the textual information, the average word count of synopses and biographies can be seen in Figure 12. It seems that biographies tend to be bigger than synopses.

C. Domain Conceptual Model

The conceptual model of the domain is illustrated in Figure 13 and presents the **main entities** of the domain, which are movies and people involved in them, such as actors, directors, writers, among others.

A **movie** can have as attributes its title, release date, runtime in minutes and synopsis. Moreover, a movie can also be from a certain country and have many different languages associated with it, as well as ratings from different sources like IMDb [5], Rotten Tomatoes [10] and even Metacritic [11] (metascore). Furthermore, a movie can also be in different streaming platforms such as Hulu [12], Disney+ [13], Netflix [9] and PrimeVideo [8].

A **person** has as attributes its name, date of birth, gender and biography. In addition, a person can also have one or more job titles in a movie, being for example an actor, a writer or director, and, in the case of being an actor, the character or list of characters that it plays in the movie.

D. Data and Information Retrieval Tasks

The focus of the data retrieval tasks are movies and the people involved in them. Some possible queries are:

1) Retrieve all movies in which an actor appears;
2) Retrieve all movies by director;
3) Retrieve high rated movies for each genre;
4) Retrieve high rated movies for each year;
5) Retrieve high rated movies for each language;

6) Retrieve movies that fit into a text description (e.g. "second world war movies");
7) Retrieve people that fit into a text description (e.g. "young puerto rican actor").
8) Retrieve movies set in a given time period.
9) Retrieve actors who have won some kind of award.
10) Retrieve controversial or polarizing movies.

Some of these can already be performed in other tools or applications, however, currently, there is no single solution that offers all of the possibilities. For example, queries 1, 2, 3 and 4 can be performed in the IMDb official website [5], but with only their own rating being considered, and not those of other platforms. For query 6, similar results can be obtained in WhatIsMyMovie [14] through the use of Machine Learning.

III. Information Retrieval System

In order to establish the base for the information retrieval process and evaluate the results produced, **Apache Solr** [15] was chosen as the indexing and retrieval tool. The main reasons for this are that Solr is an industry standard in information retrieval, having extensive documentation available as well as good performance, both while indexing and retrieving results. In addition to this, it also features an administrator user interface, something welcomed by new users. Despite this, there are some drawbacks as well, most notably a steep learning curve when it comes to the initial configuration of the tool.

A. Document Collection

The collection used for indexing is comprised of 4 different datasets, as mentioned in Section II-A: the IMDb Official Dataset, the IMDb Scraped Dataset, the Streaming Dataset and IMDb Movie Pages. With the joint information of all of the datasets, two documents were defined: a document for **Movies** and another for **People** who work and are related to these movies. In total, 105,975 documents were indexed, being 15,531 of them Movies (14.7%) and 90,444 People (85.3%).

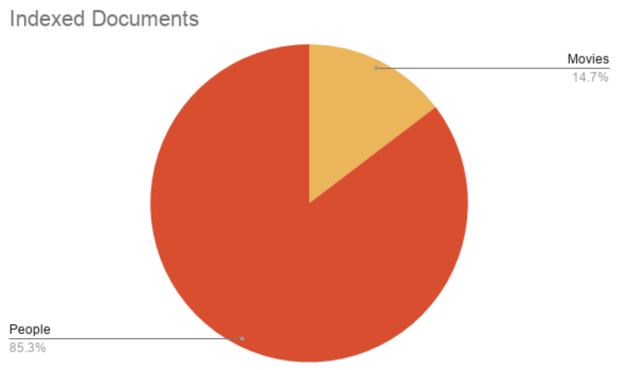


Fig. 2. Number of Indexed Documents

B. Document Indexing

Both documents (Movies and People) were indexed in the same core, with a schema being created containing the fields that can be seen in Tables I and II. All fields present in the original datasets were indexed.

For the movies, all relevant information from the previously mentioned datasets was combined in a single CSV file, `movies.csv`, using Python's library Pandas [16]. The same was done for the people information, in a single CSV file, `people.csv`. The relation between these two files is as follows:

- Each row in the movies file can have multiple corresponding rows in the people file (one for each person involved in that movie);
- Each row in the people file has exactly one corresponding row in the movies file.

To index these files, the following commands were used:

```
bin/post -c movies -params "f.genres.
    split=true&f.genres.separator=,"
    movies.csv

bin/post -c movies people.csv
```

Another indexing strategy was also tried: indexing the two different types of documents in two cores. This made it possible to retrieve the different type of documents in a query without mentioning fields from both of them explicitly. For instance, when searching for a movie (e.g.: Inception) the results list included the actors that played a role in this movie (e.g.: Leonardo DiCaprio).

Looking at these strategies, both are acceptable approaches. The difference between them consists in the schema. Since there is only one core, the fields of the schema cannot be set to *required*, to allow the indexing of both types of documents. For the two cores approach, it is easier to join the common fields between the documents. However, to search for just one type of document the user needs to be aware that each core corresponds to a single type. For the one core approach, there is no need to change between cores. So to retrieve different documents the query has to mention the specific field(s) of the document that the user wants to retrieve. To join documents the idea is the same. In order to keep it simple, we decided to go for the one core approach: we do not need to create another core, having the same results in either solution.

1) Movies: Movie documents contain certain details about a movie like its title, release date and synopsis. Each movie is identified by an unique ID, attributed by IMDb. The most relevant fields can be seen in Table I.

Field	**Description**
imdb_id	ID of the movie within the IMDb database
popularTitle	Title for which the movie is best known as
synopsis	Brief summary of the movie
runtimeMinutes	Duration of a movie in minutes
genres	Various genres of a movie (e.g.: action)
netflix, primevideo, disney, hulu	Streaming platforms in which the movie is available at

TABLE I
Most relevant fields for Movie documents

Other fields present in Movie documents are: startYear, originalTitle, isAdult.

2) People: Similarly to movies, People type documents have some descriptive fields. These documents are identified by a pair of IDs (both attributed by IMDb), one for the person and one for the associated movie. The most relevant fields can be seen in Table II.

Field	Description
imdb_id	ID of the movie within the IMDb database
imdb_name_id	ID of the person within the IMDb database
category	Type of job carried out (e.g.: actor, composer, etc.)
characters	Character(s) played out by the actor/actress in the movie (if applicable)
name	The person's name
bio	The person's biography

TABLE II
MOST RELEVANT FIELDS FOR PEOPLE DOCUMENTS

Other fields present in People documents are: date_of_birth, date_of_death, birth_name, reason_of_death, death_details, birth_details, children, height, divorces, place_of_birth, place_of_death, spouses and so on.

C. Filter Types

Besides the already existing Solr filter types, two new filter types were created in order to better accommodate the specific needs of this system. The first filter type, named **text_title**, was used in movie titles and movie character names, while the second, **custom_text**, was created for both movie synopsis and people's bios. Tables III and IV show the filters used in each of the new filter types.

Movie Title & Character Names	Index	Query
White Space Tokenizer	✓	✓
Standard Tokenizer	×	×
Lowercase Filter	✓	✓
Porter Stemming	✓	✓
Synonym Graph Filter	×	✓
Duplicate Removal	✓	×
English Possessive Filter	×	×
Stop Word Filter	×	×

TABLE III
FILTERS USED IN MOVIE TITLE & CHARACTER NAMES FILTER TYPE

Synopsis & Bio	Index	Query
White Space Tokenizer	×	×
Standard Tokenizer	✓	✓
Lowercase Filter	✓	✓
Porter Stemming	✓	✓
Synonym Graph Filter	×	✓
Duplicate Removal	✓	✓
English Possessive Filter	✓	✓
Stop Word Filter	✓	✓

TABLE IV
FILTERS USED IN SYNOPSIS & BIO FILTER TYPE

Regarding the stop word and synonym graph filters, the lists used for these were provided by Solr and manually assembled, respectively.

D. Retrieval Process

At this point, we have a Solr Schema defined with all the documents we need (subsection III-A and subsection III-B) to start the retrieval process. From the Solr interface we used the *Standard Query Parser* [17] together with *Common Query Parameters* [18] and explored the scope of the **eDisMax** (Extended DisMax) module [19].

E. Standard Query Parser

The Standard Query Parser is the basis of Solr search. It allows to insert the query that we want to search with as well as some operators to make the search more flexible.

Using only this item we can create some simple queries, using the query field, just by looking for something in a field:

$$synopsis : murder$$

This simple query returns documents that contain the field *synopsis*, which includes the token *murder* (consult the results in the Figure 14) and corresponds to the information need: Movies about murders. To join different fields or to look for similar words, the parser has some operators that help the user, such as AND (&&), OR (||), NOT (!), + and -.

- AND or && - Boolean operator that requires the presence of both terms
- OR or || - Boolean operator that requires either term (or both terms) to be present
- NOT or ! - Operator that doesn't let the term be present
- $-$ (minus sign) - Operator that has the same function as the *NOT* operator
- $+$ (plus sign) - Operator that requires the presence of a term

Joining these operators with wildcards searches ($*$ and ?), similar to regex expressions, we are able to represent the information need with more precision or, at least, with a wider range, in order to try to obtain the best results.

For the information need above we can develop a better query, for example:

$$popularTitle : (assassin\ murder\ kill*)\ genres : action$$

With this query we are able to have not only matches on the field *popularTitle* with the token *murder*, but also matches that contain related words such as *assassin*, *kill* and other words that have this last term as their root.

You can consult the results in Figure 15.

1) Common Query Parameters: From this functionality we used the following parameters:

- start - Offset into where to begin displaying results. The value used in this field was always "0", as we wanted to retrieve the first results
- rows - Number of results returned. Default number used was 10. The value was only changed sometimes to provide a greater context for tests
- fq - Filter where a query that can be used to restrict the super set of documents returned can be defined , without influencing the score.

The most important functionality to mention is the Filter Query (fq). This was used in the instance where we tested the use of two different cores, where one had the movies collection and the other the people collection. Here we are able to return, for instance, the results corresponding to the information need: actors who play a role in the movie Inception. To obtain the results for this information need, we can search for:

$$category : act*$$

and use the following filter query:

$$\{!join\ from = imdb\_id\ fromIndex = movies\\ to = imdb\_id\}\ popularTitle : Inception$$

It is important to keep $act*$ in order to include actresses as well. As it was intended to know which of the actors were in a certain movie, this query was conducted in the people's core. This solution performed quite well, returning all the respective actors and actresses present in the dataset (Figure 16).

2) Extended DisMax: eDisMax means *Extended DisMax Query Parser* and it is an improved version of the *DisMax* query parser, being more powerful. In order to make our search more robust, we used the parameter *stopwords* (only available on *eDisMax*) and the parameter *qf* (Query Field, already in DisMax). Moreover, as expected, the *q* (Query) parameter is also used.

- stopwords - Boolean parameter indicating if the StopFilterFactory configured in the query analyzer should be respected when parsing the query.
- qf - List of fields, each of which is assigned a boost factor to increase or decrease that particular field's importance in the query.
- q - The essence of the search. The parameter supports raw input strings provided by users with no special escaping.

To observe the power of this query parser, let's analyse the following query:

$$netflix : true\ AND\ (world\ war\ 2)$$

It is important to mention that this query could be also written as:

$$netflix : true\ AND\\ ((world\ war\ 2)\ OR\ WW2\ OR\ WWII)$$

Although, those terms are already considered by Solr due to the existence of the synonyms graph filter (see subsection III-C).

In both queries the information need is explicit: movies about World War 2 that are available on the Netflix platform. The *qf* was filled with:

$$synopsis^3\ popularTitle^2$$

With this, Solr retrieves most of the movies about World War 2 or at least movies that are in some manner related to it (results in Figure 17).

3) Raw Query Parameters: Solr has many features that are not available in the user interface. When someone wants to use one of those features, they can use the *Raw Query Parameters* field.

This field was used to find the British actors that have been nominated or won an Oscar award. Each People document has the information of someone and their role in a movie. So, if there's information on the dataset of an actor performing in 10 movies, there will be 10 retrieved documents. With the purpose of not getting all of these documents being retrieved from only an individual, we used the *Raw Query Parameters* to group the results by individual, using an unique id - *imdb_name_id*.

Thus, to translate the information need to the query format we used:

$$category : act*\ AND\ oscar\ AND\\ place\_of\_birth : engl*\ AND\ (winning\ nomina*)$$

And to group the results we just inserted the following command in the mentioned field:

$$group.field = imdb\_name\_id\&group = true$$

Therefore, each document retrieved regards a different person that supposedly matches the information need.

It is relevant to mention as well that the query parser used for the search was *eDisMax* with the string:

$$bio^2$$

Part of the results corresponding to this search are in Figure 18.

F. Platform Evaluation

1) Results: To perform a formal evaluation of the final system, two queries were executed, each one pertaining to a different type of document: Movies and People. The details of both are displayed below:

- Query #1
  - **Information Need:** Retrieve the British actors that have been nominated for or won an Oscar award.
  - **Query:** [category:act* AND oscar AND place_of_birth:engl* AND (winning nomina*)]
  - **Field weights used:** bio^2
- Query #2
  - **Information Need:** Retrieve the movies that are set during World War 2.
  - **Query:** [world war 2 nazi holocaust]
  - **Field weights used:** synopsis^3 popularTitle^2 genres

Regarding query #1, the reason why only the *bio* field was considered (and not the set of weights indicated in Section III-E2) is because in this case the other fields (excluding the ones indicated in the query itself) would only bring irrelevant information. For example, the system would return actors or characters named "Oscar", which doesn't fit the information need.

The top 10 results of both queries can be seen in Table V, as well as the respective precision-recall graph in Figure 3.

Result no.	1	2	3	4	5	6	7	8	9	10
Query #1	R	N	N	R	N	N	R	N	N	R
Query #2	R	N	R	R	R	N	R	R	N	R

TABLE V
RESULT RELEVANCE (R - RELEVANT; N - NON-RELEVANT)

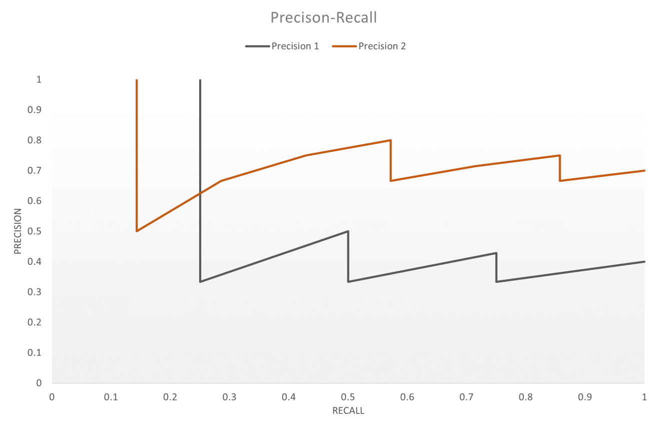


Fig. 3. Precision-Recall graph for the evaluation queries

The calculated average precision for query #1 and #2 are 0.582 and 0.769, respectively.

As we can observe, query #2 maintained a greater precision than #1 over all recall levels. This can be attributed to the nature of the queries themselves: it is more common for a movie which contains “world war 2” or a synonym in its synopsis or title to be set during that time period and not just tangentially related to the concept.

Nevertheless, the system obtained a mean average precision of 0.675 over both queries, which can be deemed acceptable.

2) *Information Retrieval Software:* In what comes to Solr as an information retrieval tool, there are some things to note. Firstly, while it exists a great deal of documentation available, this proved insufficient many times, slowing down development. More specifically, there was an almost complete lack of concrete examples for the implementation and usage of the various modules. In addition to this, while the administrator user interface is a helpful feature, it isn’t the most intuitive and it could benefit from more documentation as well. Finally, a dedicated mechanism for re-indexing is also missing, meaning that this process had to be done manually or through the use of self-developed scripts.

IV. SEMANTIC WEB

Semantic Web is referred to as a Web of Data, enabling people to create data stores, build vocabularies, and write rules for the data on the Web. It is an extension of the World Wide Web that has as its main purpose to provide a common framework to share and reuse data across various applications, using multiple technologies to ensure this such as **RDF**, **OWL**, and **SPARQL**.

RDF (Resource Description Framework) is a framework that expresses the details about resources, being these documents, physical objects, people, and even abstract concepts. It is related to the concept of Linked Data, which is the collection of interrelated datasets on the Web. RDF is used when the information within the Web needs to be processed by applications, making the information machine-readable and able to be used by different types of applications.

In Semantic Web is also very important to organize data and build vocabularies.For this, **OWL (Web Ontology Language)** is used, allowing to enrich the meaning of data. OWL is a Semantic Web language that is able to represent complex knowledge about information and its relations with other types of information, being an ontology a set of specific and descriptive statements about a certain domain of interest.

As Semantic Web can be seen as a global database, it is needed to have a query language that enables retrieving information from the data itself. **SPARQL** is the query language used in Semantic Web and is used to convey queries in diverse data sources, supporting aggregation, negation, subqueries, among other features, as well as displaying its results on sets or RDF graphs.

The next step in this project was to **develop an ontology for the project’s domain**, creating all of the needed entities, such as classes, object and data type properties, along with their relations and restrictions. Furthermore, the ontology was populated and queries were created using SPARQL to retrieve information. The ontology was developed using **Protégé** [20], a free and open-source ontology editor framework that provides a graphic interface and several plugins to define ontologies and manage a knowledge system.

A. *Related Ontologies*

During the research for related ontologies, in order to better understand the work that already had been implemented in the film industry domain, **two ontologies** were found. The first ontology named **MO - The Movie Ontology** [21] was created by the Department of Informatics of the University of Zurich and intended to provide a user-friendly presentation of movie descriptions with a vast vocabulary. The second ontology, **”A Creative Works Ontology for the Film and Television Industry”** [22], was created by **MovieLabs** [23], an independent non-profit organization founded by several well-known companies like Disney, Paramount, Twentieth Century Fox, Sony Pictures, Universal, and Warner Bros. studios, with the goal of advancing the research and development in motion picture distribution and protection.

When comparing both ontologies, the second one was more interesting to explore as it is more related to this project and its entities seemed to be better organized and described. Regarding the entities present in the ontology, they are the following:

- **Creative Works -** films or television series;

- **People -** Contributors in creative works (e.g., producers, actors, directors);
- **Locations -** Real or fictional location (e.g., filming locations, production country, setting, the birthplace of a person);
- **Groups -** a collection of creative works, where creative works are grouped by a number of things (e.g., franchise, universe, character);
- **Awards -** Information related to movie and television awards and nominations.

This ontology appears to be general enough to cover most of the scenarios in this area yet detailed enough to represent this type of domain. Moreover, some entities and attributes of this ontology relate to our project, particularly the Creative Works and People entities, which can represent various attributes present in our schema. Despite of all of these positive characteristics, this ontology is not currently being used in any application or project, and its documentation, although not bad, it is not the best as it is only available a pdf with its details and not the actual ontology. Therefore, it was decided to build an ontology from the beginning, only taking some inspiration from the previous ontologies mentioned.

B. Web Ontology

In this section, the most important entities of the created ontology are described.

1) Classes: The following classes were created in order to model the domain:

- **Movie:** to represent movies;
- **Person:** people involved in the movie industry;
  - **PersonInMovie:** to capture the association between a person in a particular movie (e.g., their role in that movie);
- **Genre:** film genres;
- **Location:** geographic locations;
- **Character:** movie characters.

2) Object Properties: The object properties in Table VI were created to best model the associations between the different classes. These associations can also be seen in Figure 4.

Object Properties	Domain	Range
playsAs	PersonInMovie	Character
hasParticipated	PersonInMovie	Movie
appearsIn	Character	Movie
wasBornIn	Person	Location
hasGenre	Movie	Genre
isPerson	PersonInMovie	Person

TABLE VI
OBJECT PROPERTIES

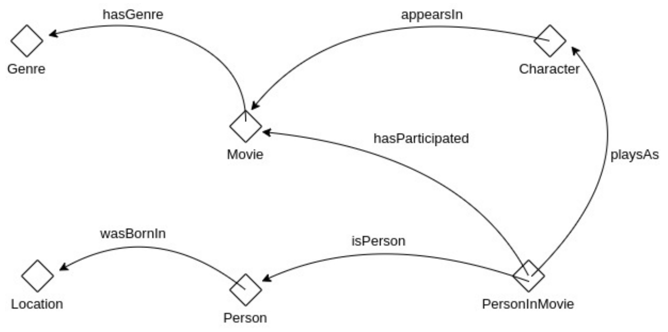


Fig. 4. Diagram modeling the object properties

3) Data Properties: To be able to store various information about the different classes, data properties were created, which can be seen in Table VII.

Domain	Data Property	Range
Movie	isStreamedOnNetflix	xsd:boolean
	isStreamedOnHulu	xsd:boolean
	isStreamedOnPrime	xsd:boolean
	isStreamedOnDisney	xsd:boolean
	Adult	xsd:boolean
	Synopsis	xsd:string
	RunTime	xsd:integer
Person	IMDbPersonID	xsd:string
	Bio	xsd:string
	BirthDetails	xsd:string
	Children	xsd:integer
	Divorces	xsd:integer
	Height	xsd:double
	Spouses	xsd:integer
PersonInMovie	Role	xsd:string
-	Name	xsd:string
Movie, PersonInMovie	IMDbMovieID	xsd:string
Movie, Person	Date	xsd:string

TABLE VII
DATA PROPERTIES

C. Ontology Population

To populate the ontology the **Cellfie plugin** [25] was used since it already comes bundled with Protégé by default. To that purpose, the CSV files containing the data were agregated in a single XLSX workbook split in two different spreadsheets (movies and people). However, only a few hundred rows per sheet were kept in the final document because Protégé doesn't handle large amounts of data particularly well. This data reduction was first applied to the movies file, keeping only the first 400 rows. As for the *People* information, a Python [26] script was created so as to keep only the rows with information connected with the movies that were kept after the reduction.

In addition to this, an extra column was created in the *People* file containing the person's name followed by the association's IMDb movie ID. This was done so as to act as an identifier for the individuals of the PersonInMovie class.

The transformation rules used to import the data can be seen in Figure 19. Something worth noting is that the identifier for the *Movie* individuals isn't the movie's title but their IMDb ID due to the presence of some characters in this field which aren't accepted in Protégé in identifier names.

D. SPARQL Queries and Result Discussion

The SPARQL queries performed can be seen in figures 5 to 8, with the respective results displayed in figures 20 to 23. These queries were performed using the SnapQL [27] plugin instead of Protégé's default SPARQL mechanism due to the better performance and responsiveness of the application in the first case.

In the first query, the movies with a director associated are listed, ordered by the latter's name. It can be observed that only 43 results were returned despite the 400 movies present the data. This is because the remaining movies don't have a director associated or their role is described in some other way.

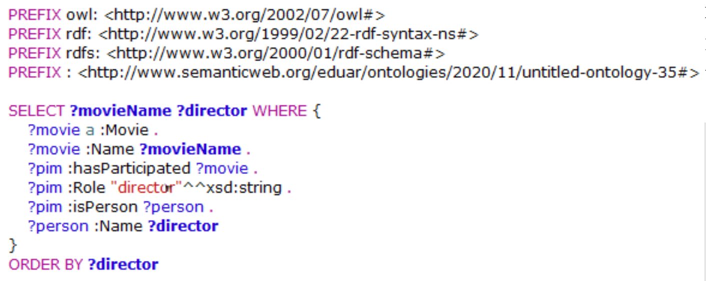

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://www.semanticweb.org/eduar/ontologies/2020/11/untitled-ontology-35#>

SELECT ?movieName ?director WHERE {
   ?movie a :Movie .
   ?movie :Name ?movieName .
   ?pim :hasParticipated ?movie .
   ?pim :Role "director"^^xsd:string .
   ?pim :isPerson ?person .
   ?person :Name ?director
}
ORDER BY ?director
```

Fig. 5. Query 1

A more realistic example is illustrated in query number 2, where all the movies with a run time under two hours currently streaming on Amazon Prime Video are listed. The number of results here are already much higher than in the previous query.

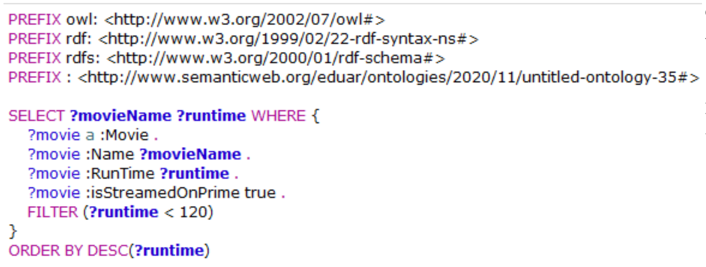

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://www.semanticweb.org/eduar/ontologies/2020/11/untitled-ontology-35#>

SELECT ?movieName ?runtime WHERE {
   ?movie a :Movie .
   ?movie :Name ?movieName .
   ?movie :RunTime ?runtime .
   ?movie :isStreamedOnPrime true .
   FILTER (?runtime < 120)
}
ORDER BY DESC(?runtime)
```

Fig. 6. Query 2

Query number 3 resembles one of queries performed to test the information retrieval system previously presented, with the information need of the latter being ”movies set during World War II”. However, since there are several ways to describe this period, it becomes somewhat unfeasible to list them all in a SPARQL query, unlike in Solr, where the multitude of filters available make this process much easier. As such, it was opted to only list movies about war in general, however, not all of the results returned are as strongly connected to this theme as one would hope.

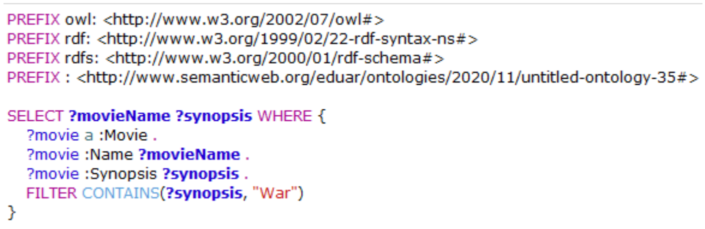

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://www.semanticweb.org/eduar/ontologies/2020/11/untitled-ontology-35#>

SELECT ?movieName ?synopsis WHERE {
   ?movie a :Movie .
   ?movie :Name ?movieName .
   ?movie :Synopsis ?synopsis .
   FILTER CONTAINS(?synopsis, "War")
}
```

Fig. 7. Query 3

Finally, query number 4 represents the most popular actors, in this case, the actors which have participated in the most number of movies, something which could be used in a trivia website, for example. Initially, the total number of characters portrayed by the actor was also displayed, but it was soon realized that most actors only play one character in each movie, which meant that these values would be the same. Due to this, it was decided to remove this variable from the query.

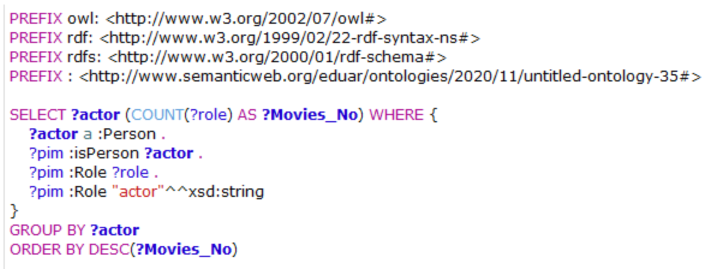

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://www.semanticweb.org/eduar/ontologies/2020/11/untitled-ontology-35#>

SELECT ?actor (COUNT(?role) AS ?Movies_No) WHERE {
   ?actor a :Person .
   ?pim :isPerson ?actor .
   ?pim :Role ?role .
   ?pim :Role "actor"^^xsd:string
}
GROUP BY ?actor
ORDER BY DESC(?Movies_No)
```

Fig. 8. Query 4

E. Protégé vs Solr

Protégé is a free, open-source platform that provides a growing user community with a suite of tools to construct domain models and knowledge-based applications with ontologies [20]. Protege-OWL is an extension to Protégé that supports the Web Ontology Language (OWL) [24] and with it help, we would able to create our ontology in a interactive way with a very useful interface.

In addiction, Protégé and Web Semantic tools allow:

- Easy retrieval of related knowledge;
- Use of reasoners to infer unspecified knowledge;
- Use of a wide array of helpful plugins.

However, SparQL is the protocol used to query this type of data means the user must have technical knowledge and knows somewhat precisely what to look for. Other problems regarding Protégé were detected:

- It consumes lots of memory and cpu;
- Unknown crashes during the development;
- Impossible to load a large set of data to populate the ontology.

In the other hand, Information Retrieval Tools is better when the objective is search in unstructured data. It allows a use of a powerful set of filters and weighting system retrieving a ranking of results related to the search easing the found of relevant results. Solr has the possibility of performing informal queries, i.e, queries without technical expertise, what is a huge advantage to not expertise users. Yet, Solr needs a schema to work, a schema that will allow the indexing of data and it is not trivial to build a schema with the best solution to this type of system.

Using the search about war in both system 7, 22 and 17, we can check that the results retrieved by the IR System are ordered by a value that tries to specify the relevance of the document, giving us movies about war, and not only for one occurrence of the word "war". However, the results from Protégé, despite having a synopsis that contains war, theres is no different relevance between them. For example, a synopsis that talks about a war inside a family will appear in our search, however this do not satisfies the search for movies about war.

F. Applications

The created ontology could potentially be useful for enhancing databases related to the movie industry, as it provides a simple and efficient way of representing this domain and allows an easy integration with other systems. Additionally, this ontology also combines the information present in movie databases, for example IMDb, with streaming platform details, which could be interesting for a future consumer application.

Regarding other topics apart from movie world, our ontology can be useful for the other topics such us book. The classes defined can be matched and used in another ontology using the command `owl:sameAs`.

V. Conclusions

With the information retrieval platform presented, a slightly experienced user can have most of their (theme related) information needs fulfilled with moderate precision levels.

However, the results returned are only as good as the collection available. Since in total around 100,000 documents are used for this purpose, there are of course limitations for the total amount retrieved. Furthermore, because the bottlenecking strategy doesn't take into account any specific attributes, it might lead to some odd situations such as, for example, when movies from almost 100 years ago are being shown to the user while other more recent ones are absent.

In the future, these situations could be mitigated, making it easier for less experienced users to make use of the platform by further optimizing the system's configurations.

In what concerns the ontology developed in the final stage of the project, although the number of examples had to be reduced due to Protégé's incapability of handling big datasets, and even with the information retrieval platform providing more useful filters and enabling result ranking and weighting, the results obtained were good. The created ontology is simple and easy to understand, having a well-defined structure that facilitates the querying process.

References

[1] IMDb Official Dataset. https://datasets.imdbws.com.Accessed in October, 2020.

[2] IMDb Software Integration Help Page. https://help.imdb.com/article/imdb/general-information/can-i-use-imdb-data-in-my-software/G5JTRESSHJBBHTGX#. Accessed in October, 2020.

[3] IMDb's Developer Page. https://developer.imdb.com/?ref_=helpms_ih_gi_developer. Accessed in October, 2020.

[4] Stefano Leone. IMDb Scraped Dataset.https://www.kaggle.com/stefanoleone992/imdb-extensive-dataset. Accessed in October, 2020.

[5] IMDb's Official Website.https://www.imdb.com/.Accessed in October, 2020.

[6] Kaggle. www.kaggle.com. Accessed in October, 2020.

[7] Ruchi Bhatia. Streaming Dataset. https://www.kaggle.com/ruchi798/movies-on-netflix-prime-video-hulu-and-disney. Accessed in October, 2020.

[8] Prime Video. https://www.primevideo.com/. Accessed in October, 2020.

[9] Netflix. https://www.netflix.com/. Accessed in October, 2020.

[10] Rotten Tomatoes. https://www.rottentomatoes.com/. Accessed in October, 2020.

[11] Metacritic. https://www.metacritic.com/. Accessed in October, 2020.

[12] Hulu. https://www.hulu.com. Accessed in October, 2020.

[13] Disney+. https://www.disneyplus.com Accessed in October, 2020.

[14] What is My Movie. https://www.whatismymovie.com/. Accessed in October, 2020.

[15] Apache Solr. https://lucene.apache.org/solr/. Accessed in December, 2020

[16] Pandas. https://pandas.pydata.org/. Accessed in November, 2020.

[17] The Standard Query Parser. https://lucene.apache.org/solr/guide/6_6/the-standard-query-parser.html. Accessed in December, 2020

[18] Common Query Parameters. https://lucene.apache.org/solr/guide/6_6/common-query-parameters.html. Accessed in December, 2020

[19] The Extended DisMax Query Parser. https://lucene.apache.org/solr/guide/6_6/the-extended-dismax-query-parser.html. Accessed in December, 2020

[20] Protege Wiki. https://protegewiki.stanford.edu/wiki/Protege. Accessed January, 2021.

[21] MO - the Movie Ontology. http://www.movieontology.org/. Accessed January, 2021.

[22] A Creative Works Ontology for the Film and Television Industry. https://movielabs.com/cwontology/A-Creative-Works-Ontology-for-the-Film-and-Television-Industry-Final-2018-9-24.pdf. Accessed January, 2021.

[23] MovieLabs. https://movielabs.com/who-we-are/. Accessed January, 2021.

[24] Protege-OWL Wiki. https://protegewiki.stanford.edu/wiki/Protege-OWL. Accessed January, 2021.

[25] Cellfie. https://github.com/protegeproject/cellfie-plugin. Accessed January, 2021.

[26] Python. https://www.python.org/. Accessed January, 2021.

[27] Snap-SPARQL: A Java Framework for Working with SPARQL and OWL. Matthew Horridge, Mark Alan Musen.

https://www.researchgate.net/publication/301539966_Snap-SPARQL_A_Java_Framework_for_Working_with_SPARQL_and_OWL. Accessed January, 2021.

VI. Annex

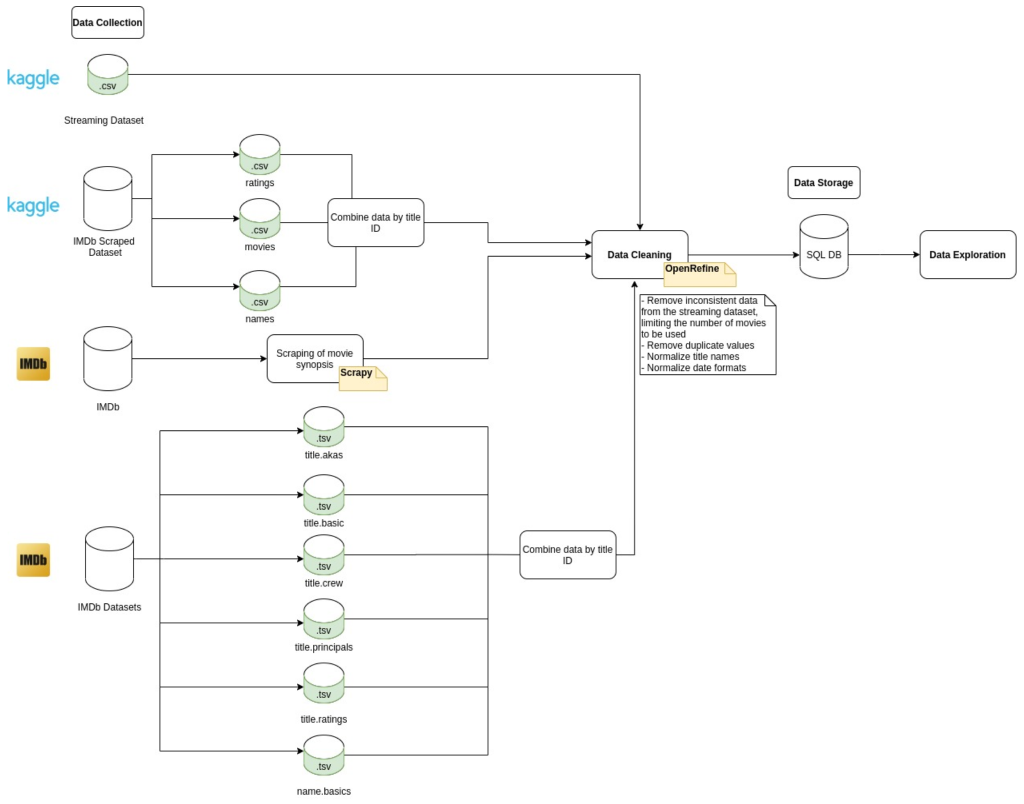


Fig. 9. Data Pipeline Diagram

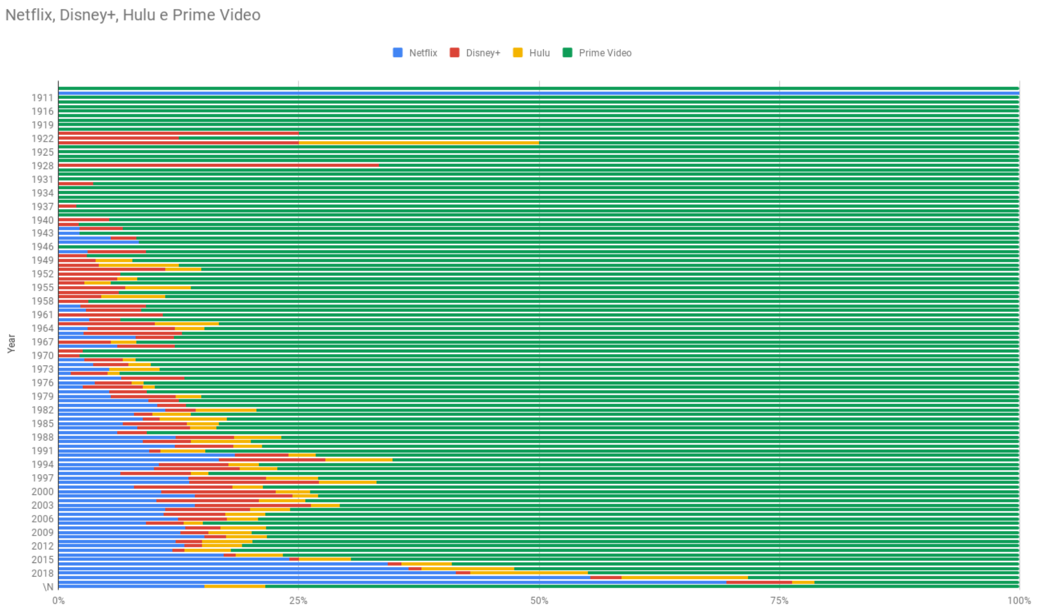


Fig. 10. Distribution of Movies on Streaming Platforms

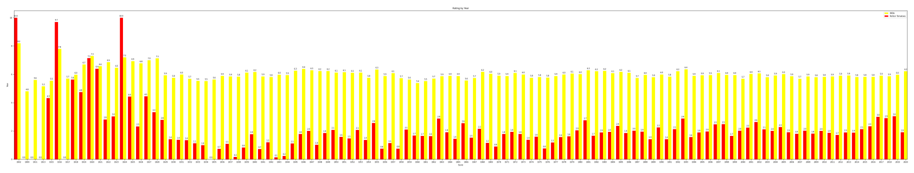


Fig. 11. Average Movie Rating by Year

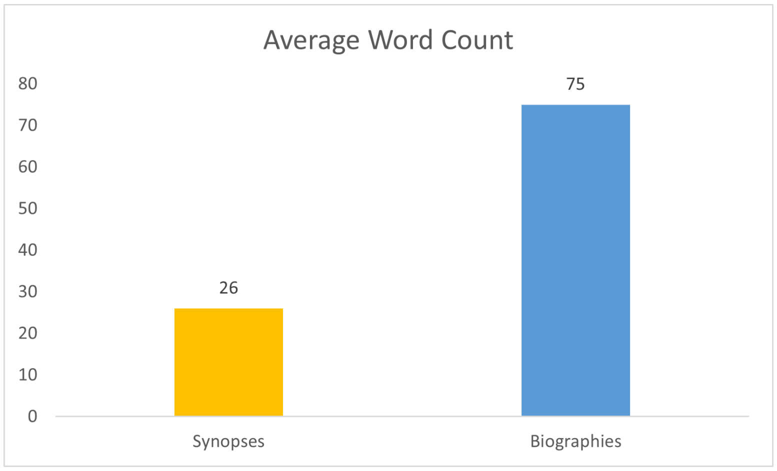


Fig. 12. Average Word Count of Synopses and Biographies

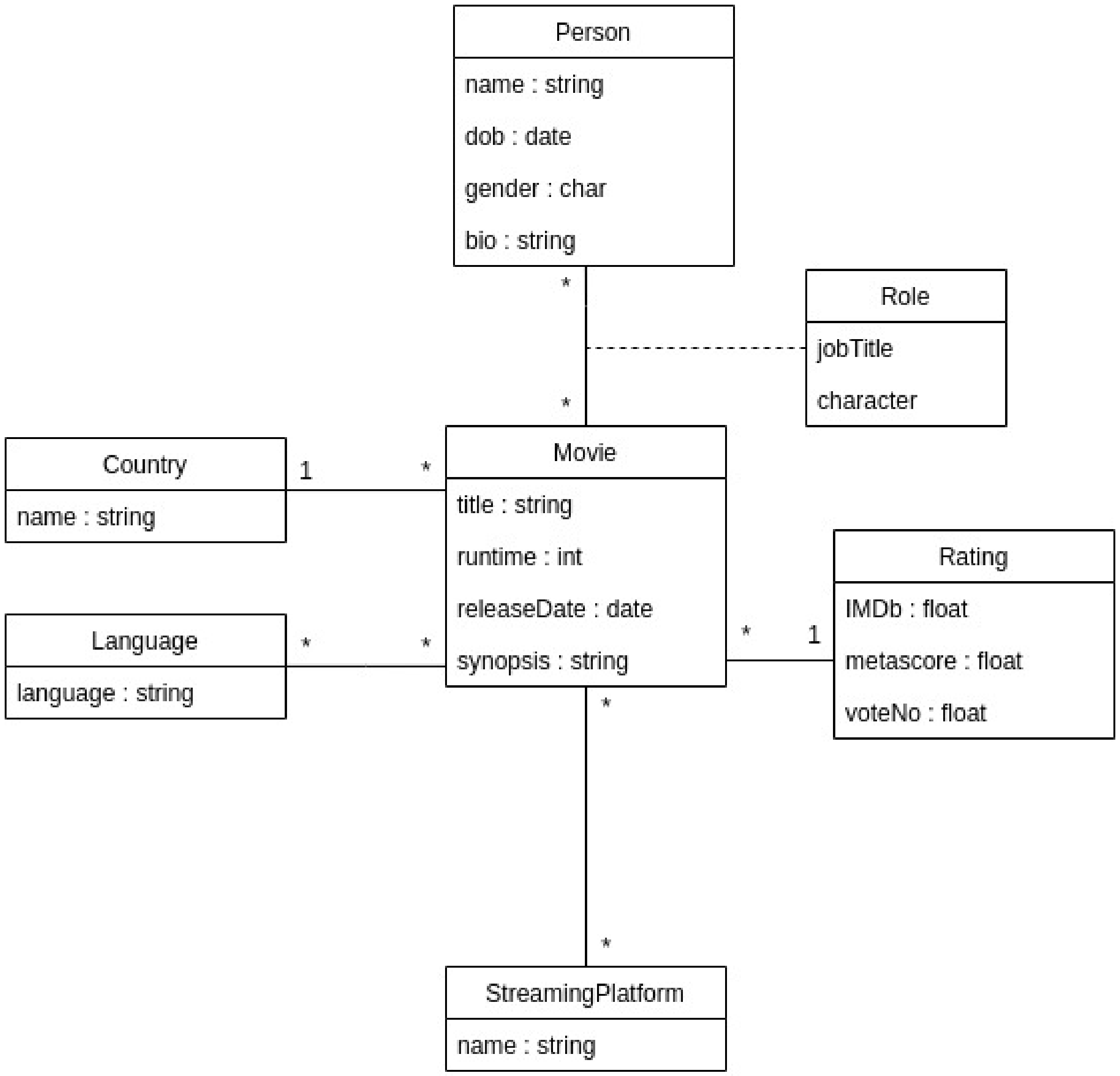
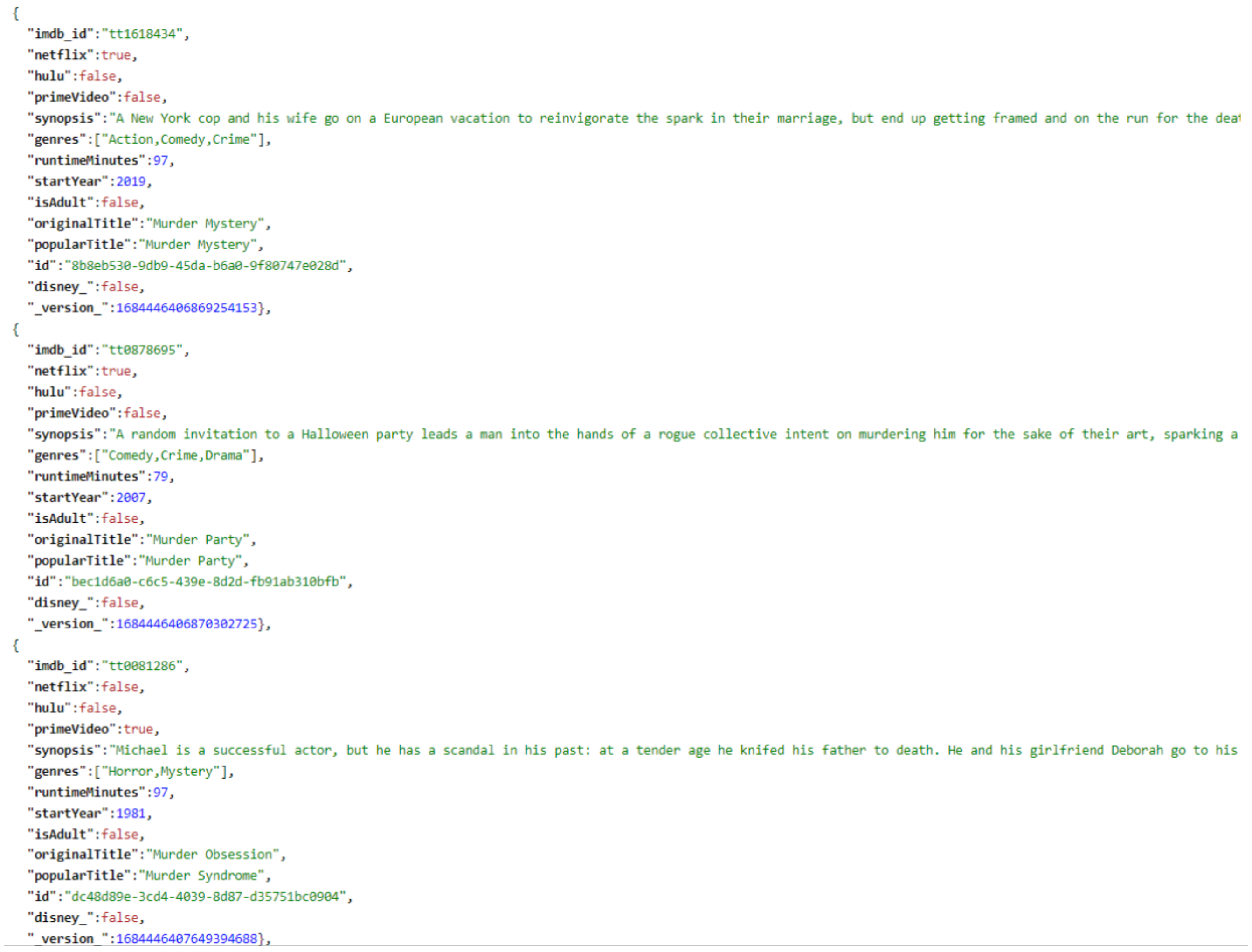
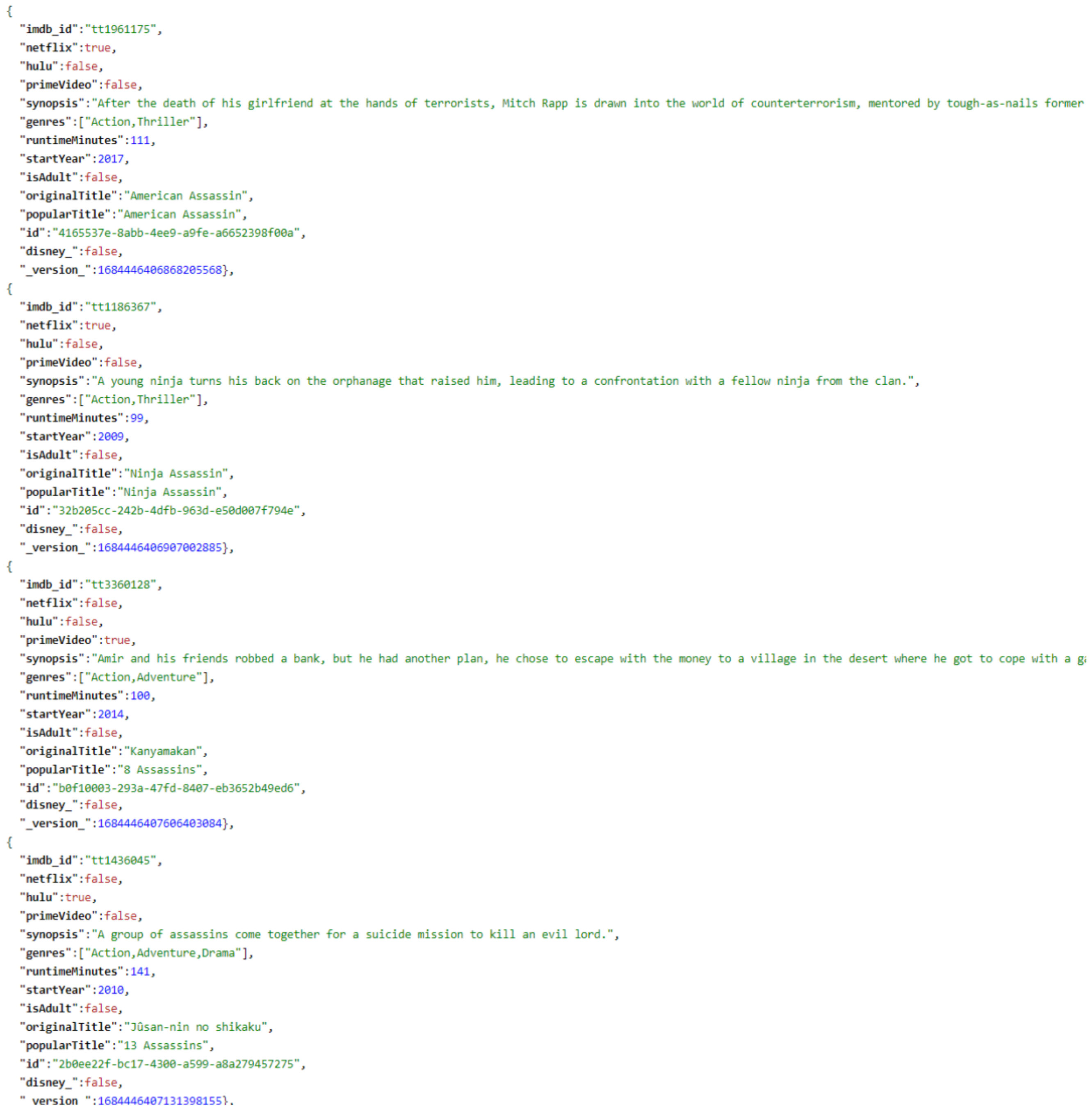


Fig. 13. Conceptual Model of the Domain



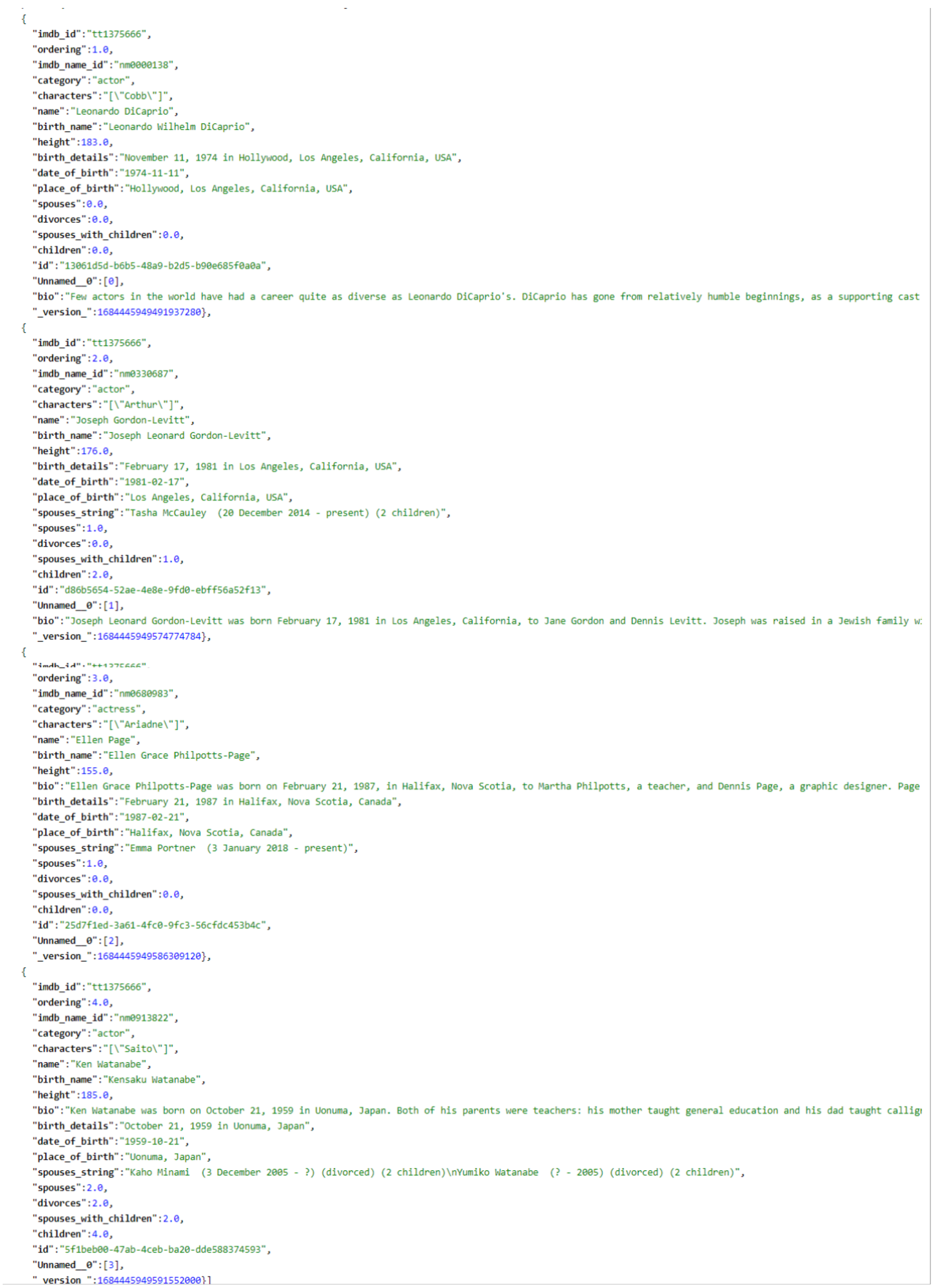
```
{
  "imdb_id":"tt1618434",
  "netflix":true,
  "hulu":false,
  "primeVideo":false,
  "synopsis":"A New York cop and his wife go on a European vacation to reinvigorate the spark in their marriage, but end up getting framed and on the run for the dea
  "genres":["Action,Comedy,Crime"],
  "runtimeMinutes":97,
  "startYear":2019,
  "isAdult":false,
  "originalTitle":"Murder Mystery",
  "popularTitle":"Murder Mystery",
  "id":"8b8eb530-9db9-45da-b6a0-9f80747e028d",
  "disney_":false,
  "_version_":1684446406869254153},
{
  "imdb_id":"tt0878695",
  "netflix":true,
  "hulu":false,
  "primeVideo":false,
  "synopsis":"A random invitation to a Halloween party leads a man into the hands of a rogue collective intent on murdering him for the sake of their art, sparking a
  "genres":["Comedy,Crime,Drama"],
  "runtimeMinutes":79,
  "startYear":2007,
  "isAdult":false,
  "originalTitle":"Murder Party",
  "popularTitle":"Murder Party",
  "id":"bec1d6a0-c6c5-439e-8d2d-fb91ab310bfb",
  "disney_":false,
  "_version_":1684446406870302725},
{
  "imdb_id":"tt0081286",
  "netflix":false,
  "hulu":false,
  "primeVideo":true,
  "synopsis":"Michael is a successful actor, but he has a scandal in his past: at a tender age he knifed his father to death. He and his girlfriend Deborah go to his
  "genres":["Horror,Mystery"],
  "runtimeMinutes":97,
  "startYear":1981,
  "isAdult":false,
  "originalTitle":"Murder Obsession",
  "popularTitle":"Murder Syndrome",
  "id":"dc48d89e-3cd4-4039-8d87-d35751bc0904",
  "disney_":false,
  "_version_":1684446407649394688},
```

Fig. 14. Results of the Query *popularTitle:murder*



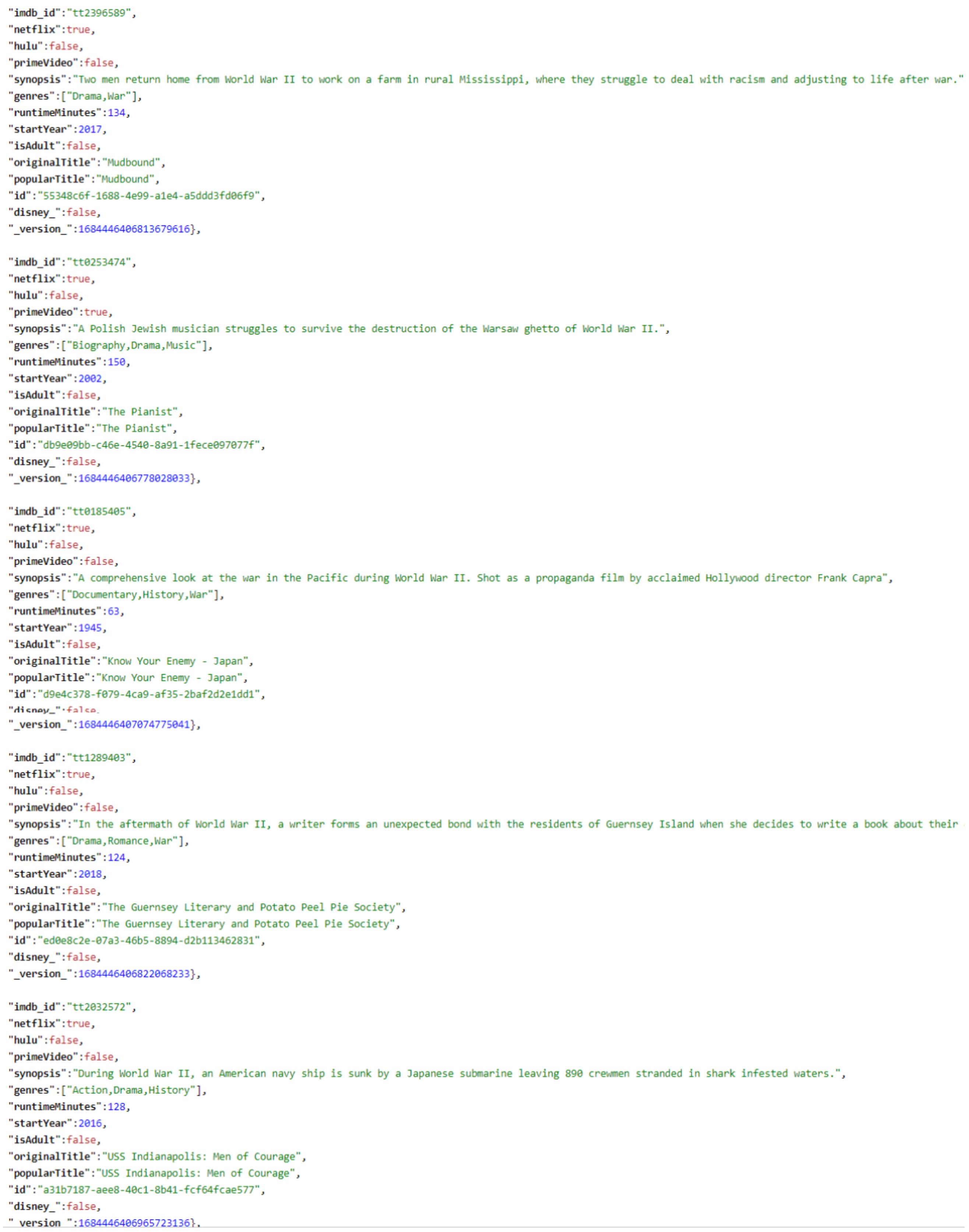
```
{
  "imdb_id":"tt1961175",
  "netflix":true,
  "hulu":false,
  "primeVideo":false,
  "synopsis":"After the death of his girlfriend at the hands of terrorists, Mitch Rapp is drawn into the world of counterterrorism, mentored by tough-as-nails former
  "genres":["Action,Thriller"],
  "runtimeMinutes":111,
  "startYear":2017,
  "isAdult":false,
  "originalTitle":"American Assassin",
  "popularTitle":"American Assassin",
  "id":"4165537e-8abb-4ee9-a9fe-a6652398f00a",
  "disney_":false,
  "_version_":1684446406868205568},
{
  "imdb_id":"tt1186367",
  "netflix":true,
  "hulu":false,
  "primeVideo":false,
  "synopsis":"A young ninja turns his back on the orphanage that raised him, leading to a confrontation with a fellow ninja from the clan.",
  "genres":["Action,Thriller"],
  "runtimeMinutes":99,
  "startYear":2009,
  "isAdult":false,
  "originalTitle":"Ninja Assassin",
  "popularTitle":"Ninja Assassin",
  "id":"32b205cc-242b-4dfb-963d-e50d007f794e",
  "disney_":false,
  "_version_":1684446406907002885},
{
  "imdb_id":"tt3360128",
  "netflix":false,
  "hulu":false,
  "primeVideo":true,
  "synopsis":"Amir and his friends robbed a bank, but he had another plan, he chose to escape with the money to a village in the desert where he got to cope with a g
  "genres":["Action,Adventure"],
  "runtimeMinutes":100,
  "startYear":2014,
  "isAdult":false,
  "originalTitle":"Kanyamakan",
  "popularTitle":"8 Assassins",
  "id":"b0f10003-293a-47fd-8407-eb3652b49ed6",
  "disney_":false,
  "_version_":1684446407606403084},
{
  "imdb_id":"tt1436045",
  "netflix":false,
  "hulu":true,
  "primeVideo":false,
  "synopsis":"A group of assassins come together for a suicide mission to kill an evil lord.",
  "genres":["Action,Adventure,Drama"],
  "runtimeMinutes":141,
  "startYear":2010,
  "isAdult":false,
  "originalTitle":"Jûsan-nin no shikaku",
  "popularTitle":"13 Assassins",
  "id":"2b0ee22f-bc17-4300-a599-a8a279457275",
  "disney_":false,
  " version ":1684446407131398155}.
```

Fig. 15. Results for IR : Movies about Murders



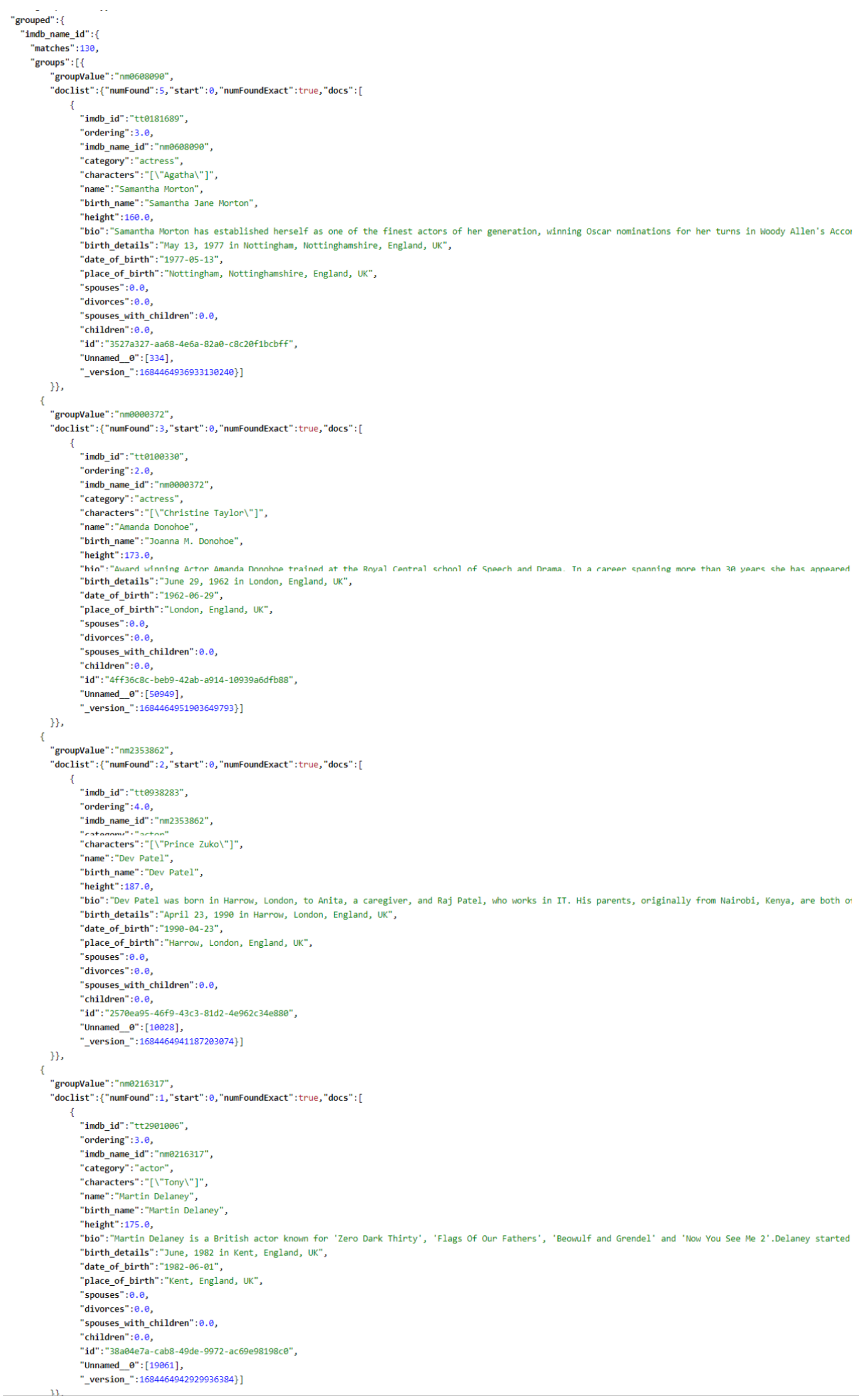
```
{
  "imdb_id":"tt1375666",
  "ordering":1.0,
  "imdb_name_id":"nm0000138",
  "category":"actor",
  "characters":"[\"Cobb\"]",
  "name":"Leonardo DiCaprio",
  "birth_name":"Leonardo Wilhelm DiCaprio",
  "height":183.0,
  "birth_details":"November 11, 1974 in Hollywood, Los Angeles, California, USA",
  "date_of_birth":"1974-11-11",
  "place_of_birth":"Hollywood, Los Angeles, California, USA",
  "spouses":0.0,
  "divorces":0.0,
  "spouses_with_children":0.0,
  "children":0.0,
  "id":"13061d5d-b6b5-48a9-b2d5-b90e685f0a0a",
  "Unnamed__0":[0],
  "bio":"Few actors in the world have had a career quite as diverse as Leonardo DiCaprio's. DiCaprio has gone from relatively humble beginnings, as a supporting cast
  "_version_":1684445949491937280},
{
  "imdb_id":"tt1375666",
  "ordering":2.0,
  "imdb_name_id":"nm0330687",
  "category":"actor",
  "characters":"[\"Arthur\"]",
  "name":"Joseph Gordon-Levitt",
  "birth_name":"Joseph Leonard Gordon-Levitt",
  "height":176.0,
  "birth_details":"February 17, 1981 in Los Angeles, California, USA",
  "date_of_birth":"1981-02-17",
  "place_of_birth":"Los Angeles, California, USA",
  "spouses_string":"Tasha McCauley  (20 December 2014 - present) (2 children)",
  "spouses":1.0,
  "divorces":0.0,
  "spouses_with_children":1.0,
  "children":2.0,
  "id":"d86b5654-52ae-4e8e-9fd0-ebff56a52f13",
  "Unnamed__0":[1],
  "bio":"Joseph Leonard Gordon-Levitt was born February 17, 1981 in Los Angeles, California, to Jane Gordon and Dennis Levitt. Joseph was raised in a Jewish family w
  "_version_":1684445949574774784},
{
  "imdb_id":"tt1375666",
  "ordering":3.0,
  "imdb_name_id":"nm0680983",
  "category":"actress",
  "characters":"[\"Ariadne\"]",
  "name":"Ellen Page",
  "birth_name":"Ellen Grace Philpotts-Page",
  "height":155.0,
  "bio":"Ellen Grace Philpotts-Page was born on February 21, 1987, in Halifax, Nova Scotia, to Martha Philpotts, a teacher, and Dennis Page, a graphic designer. Page
  "birth_details":"February 21, 1987 in Halifax, Nova Scotia, Canada",
  "date_of_birth":"1987-02-21",
  "place_of_birth":"Halifax, Nova Scotia, Canada",
  "spouses_string":"Emma Portner  (3 January 2018 - present)",
  "spouses":1.0,
  "divorces":0.0,
  "spouses_with_children":0.0,
  "children":0.0,
  "id":"25d7f1ed-3a61-4fc0-9fc3-56cfdc453b4c",
  "Unnamed__0":[2],
  "_version_":1684445949586309120},
{
  "imdb_id":"tt1375666",
  "ordering":4.0,
  "imdb_name_id":"nm0913822",
  "category":"actor",
  "characters":"[\"Saito\"]",
  "name":"Ken Watanabe",
  "birth_name":"Kensaku Watanabe",
  "height":185.0,
  "bio":"Ken Watanabe was born on October 21, 1959 in Uonuma, Japan. Both of his parents were teachers: his mother taught general education and his dad taught callig
  "birth_details":"October 21, 1959 in Uonuma, Japan",
  "date_of_birth":"1959-10-21",
  "place_of_birth":"Uonuma, Japan",
  "spouses_string":"Kaho Minami  (3 December 2005 - ?) (divorced) (2 children)\nYumiko Watanabe  (? - 2005) (divorced) (2 children)",
  "spouses":2.0,
  "divorces":2.0,
  "spouses_with_children":2.0,
  "children":4.0,
  "id":"5f1beb00-47ab-4ceb-ba20-dde588374593",
  "Unnamed__0":[3],
  "_version_":1684445949591552000}]
```

Fig. 16. Results From Using Two Different Cores



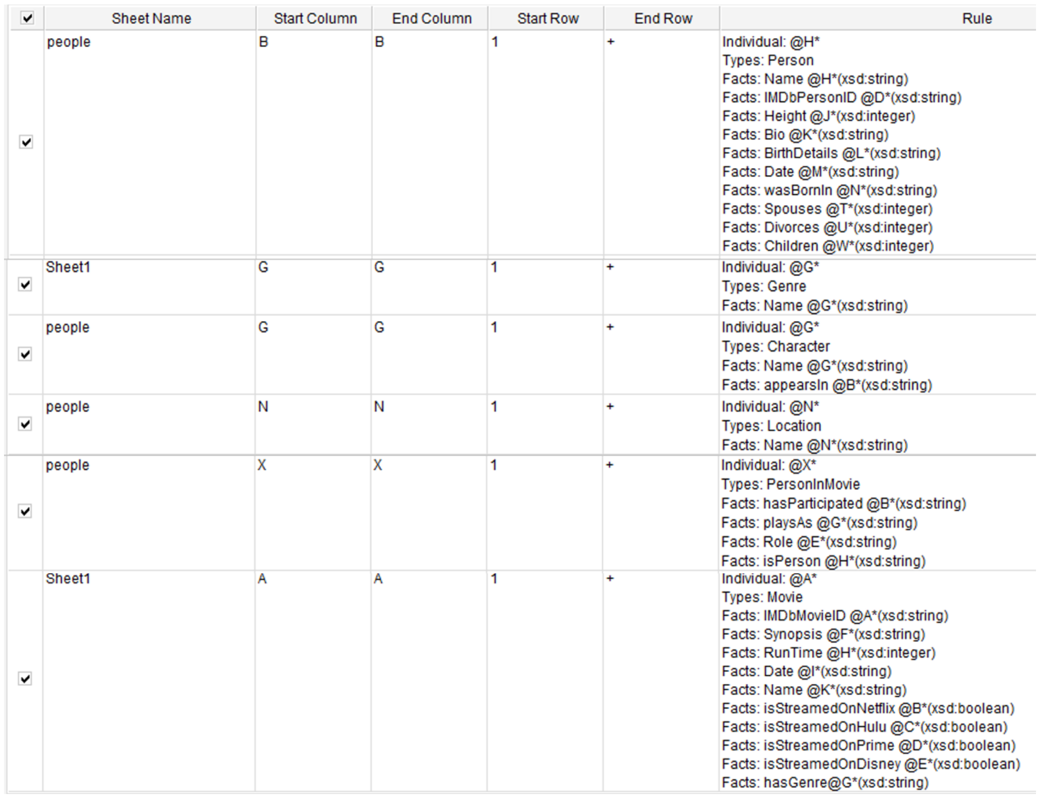
```
"imdb_id":"tt2396589",
"netflix":true,
"hulu":false,
"primeVideo":false,
"synopsis":"Two men return home from World War II to work on a farm in rural Mississippi, where they struggle to deal with racism and adjusting to life after war."
"genres":["Drama,War"],
"runtimeMinutes":134,
"startYear":2017,
"isAdult":false,
"originalTitle":"Mudbound",
"popularTitle":"Mudbound",
"id":"55348c6f-1688-4e99-a1e4-a5ddd3fd06f9",
"disney_":false,
"_version_":1684446406813679616},

"imdb_id":"tt0253474",
"netflix":true,
"hulu":false,
"primeVideo":true,
"synopsis":"A Polish Jewish musician struggles to survive the destruction of the Warsaw ghetto of World War II.",
"genres":["Biography,Drama,Music"],
"runtimeMinutes":150,
"startYear":2002,
"isAdult":false,
"originalTitle":"The Pianist",
"popularTitle":"The Pianist",
"id":"db9e09bb-c46e-4540-8a91-1fece097077f",
"disney_":false,
"_version_":1684446406778028033},

"imdb_id":"tt0185405",
"netflix":true,
"hulu":false,
"primeVideo":false,
"synopsis":"A comprehensive look at the war in the Pacific during World War II. Shot as a propaganda film by acclaimed Hollywood director Frank Capra",
"genres":["Documentary,History,War"],
"runtimeMinutes":63,
"startYear":1945,
"isAdult":false,
"originalTitle":"Know Your Enemy - Japan",
"popularTitle":"Know Your Enemy - Japan",
"id":"d9e4c378-f079-4ca9-af35-2baf2d2e1dd1",
"disney_":false,
"_version_":1684446407074775041},

"imdb_id":"tt1289403",
"netflix":true,
"hulu":false,
"primeVideo":false,
"synopsis":"In the aftermath of World War II, a writer forms an unexpected bond with the residents of Guernsey Island when she decides to write a book about their
"genres":["Drama,Romance,War"],
"runtimeMinutes":124,
"startYear":2018,
"isAdult":false,
"originalTitle":"The Guernsey Literary and Potato Peel Pie Society",
"popularTitle":"The Guernsey Literary and Potato Peel Pie Society",
"id":"ed0e8c2e-07a3-46b5-8894-d2b113462831",
"disney_":false,
"_version_":1684446406822068233},

"imdb_id":"tt2032572",
"netflix":true,
"hulu":false,
"primeVideo":false,
"synopsis":"During World War II, an American navy ship is sunk by a Japanese submarine leaving 890 crewmen stranded in shark infested waters.",
"genres":["Action,Drama,History"],
"runtimeMinutes":128,
"startYear":2016,
"isAdult":false,
"originalTitle":"USS Indianapolis: Men of Courage",
"popularTitle":"USS Indianapolis: Men of Courage",
"id":"a31b7187-aee8-40c1-8b41-fcf64fcae577",
"disney_":false,
"_version_":1684446406965723136},
```

Fig. 17. Results of a Search with Field Weights



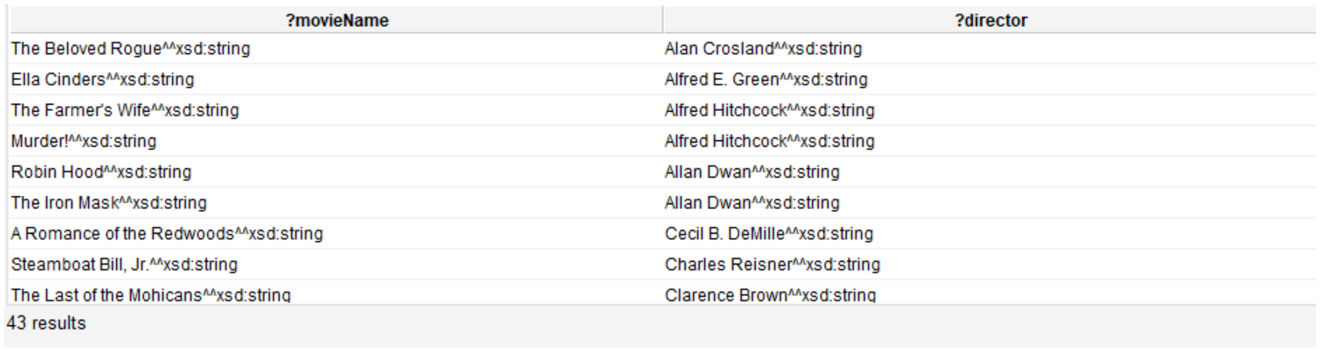
```
"grouped":{
  "imdb_name_id":{
    "matches":130,
    "groups":[{
        "groupValue":"nm0608090",
        "doclist":{"numFound":5,"start":0,"numFoundExact":true,"docs":[
            {
              "imdb_id":"tt0181689",
              "ordering":3.0,
              "imdb_name_id":"nm0608090",
              "category":"actress",
              "characters":"[\"Agatha\"]",
              "name":"Samantha Morton",
              "birth_name":"Samantha Jane Morton",
              "height":160.0,
              "bio":"Samantha Morton has established herself as one of the finest actors of her generation, winning Oscar nominations for her turns in Woody Allen's Acco
              "birth_details":"May 13, 1977 in Nottingham, Nottinghamshire, England, UK",
              "date_of_birth":"1977-05-13",
              "place_of_birth":"Nottingham, Nottinghamshire, England, UK",
              "spouses":0.0,
              "divorces":0.0,
              "spouses_with_children":0.0,
              "children":0.0,
              "id":"3527a327-aa68-4e6a-82a0-c8c20f1bcbff",
              "Unnamed__0":[334],
              "_version_":1684464936933130240}]
        }},
      {
        "groupValue":"nm0000372",
        "doclist":{"numFound":3,"start":0,"numFoundExact":true,"docs":[
            {
              "imdb_id":"tt0100330",
              "ordering":2.0,
              "imdb_name_id":"nm0000372",
              "category":"actress",
              "characters":"[\"Christine Taylor\"]",
              "name":"Amanda Donohoe",
              "birth_name":"Joanna M. Donohoe",
              "height":173.0,
              "bio":"Award winning Actor Amanda Donohoe trained at the Royal Central school of Speech and Drama. In a career spanning more than 30 years she has appeared
              "birth_details":"June 29, 1962 in London, England, UK",
              "date_of_birth":"1962-06-29",
              "place_of_birth":"London, England, UK",
              "spouses":0.0,
              "divorces":0.0,
              "spouses_with_children":0.0,
              "children":0.0,
              "id":"4ff36c8c-beb9-42ab-a914-10939a6dfb88",
              "Unnamed__0":[50949],
              "_version_":1684464951903649793}]
        }},
      {
        "groupValue":"nm2353862",
        "doclist":{"numFound":2,"start":0,"numFoundExact":true,"docs":[
            {
              "imdb_id":"tt0938283",
              "ordering":4.0,
              "imdb_name_id":"nm2353862",
              "category":"actor"
              "characters":"[\"Prince Zuko\"]",
              "name":"Dev Patel",
              "birth_name":"Dev Patel",
              "height":187.0,
              "bio":"Dev Patel was born in Harrow, London, to Anita, a caregiver, and Raj Patel, who works in IT. His parents, originally from Nairobi, Kenya, are both o
              "birth_details":"April 23, 1990 in Harrow, London, England, UK",
              "date_of_birth":"1990-04-23",
              "place_of_birth":"Harrow, London, England, UK",
              "spouses":0.0,
              "divorces":0.0,
              "spouses_with_children":0.0,
              "children":0.0,
              "id":"2570ea95-46f9-43c3-81d2-4e962c34e880",
              "Unnamed__0":[10028],
              "_version_":1684464941187203074}]
        }},
      {
        "groupValue":"nm0216317",
        "doclist":{"numFound":1,"start":0,"numFoundExact":true,"docs":[
            {
              "imdb_id":"tt2901006",
              "ordering":3.0,
              "imdb_name_id":"nm0216317",
              "category":"actor",
              "characters":"[\"Tony\"]",
              "name":"Martin Delaney",
              "birth_name":"Martin Delaney",
              "height":175.0,
              "bio":"Martin Delaney is a British actor known for 'Zero Dark Thirty', 'Flags Of Our Fathers', 'Beowulf and Grendel' and 'Now You See Me 2'.Delaney started
              "birth_details":"June, 1982 in Kent, England, UK",
              "date_of_birth":"1982-06-01",
              "place_of_birth":"Kent, England, UK",
              "spouses":0.0,
              "divorces":0.0,
              "spouses_with_children":0.0,
              "children":0.0,
              "id":"38a04e7a-cab8-49de-9972-ac69e98198c0",
              "Unnamed__0":[19061],
              "_version_":1684464942929936384}]
        }},
```

Fig. 18. Results from Search with Group Command



✓	Sheet Name	Start Column	End Column	Start Row	End Row	Rule
✓	people	B	B	1	+	Individual: @H* Types: Person Facts: Name @H*(xsd:string) Facts: IMDbPersonID @D*(xsd:string) Facts: Height @J*(xsd:integer) Facts: Bio @K*(xsd:string) Facts: BirthDetails @L*(xsd:string) Facts: Date @M*(xsd:string) Facts: wasBornIn @N*(xsd:string) Facts: Spouses @T*(xsd:integer) Facts: Divorces @U*(xsd:integer) Facts: Children @W*(xsd:integer)
✓	Sheet1	G	G	1	+	Individual: @G* Types: Genre Facts: Name @G*(xsd:string)
✓	people	G	G	1	+	Individual: @G* Types: Character Facts: Name @G*(xsd:string) Facts: appearsIn @B*(xsd:string)
✓	people	N	N	1	+	Individual: @N* Types: Location Facts: Name @N*(xsd:string)
✓	people	X	X	1	+	Individual: @X* Types: PersonInMovie Facts: hasParticipated @B*(xsd:string) Facts: playsAs @G*(xsd:string) Facts: Role @E*(xsd:string) Facts: isPerson @H*(xsd:string)
✓	Sheet1	A	A	1	+	Individual: @A* Types: Movie Facts: IMDbMovieID @A*(xsd:string) Facts: Synopsis @F*(xsd:string) Facts: RunTime @H*(xsd:integer) Facts: Date @I*(xsd:string) Facts: Name @K*(xsd:string) Facts: isStreamedOnNetflix @B*(xsd:boolean) Facts: isStreamedOnHulu @C*(xsd:boolean) Facts: isStreamedOnPrime @D*(xsd:boolean) Facts: isStreamedOnDisney @E*(xsd:boolean) Facts: hasGenre@G*(xsd:string)

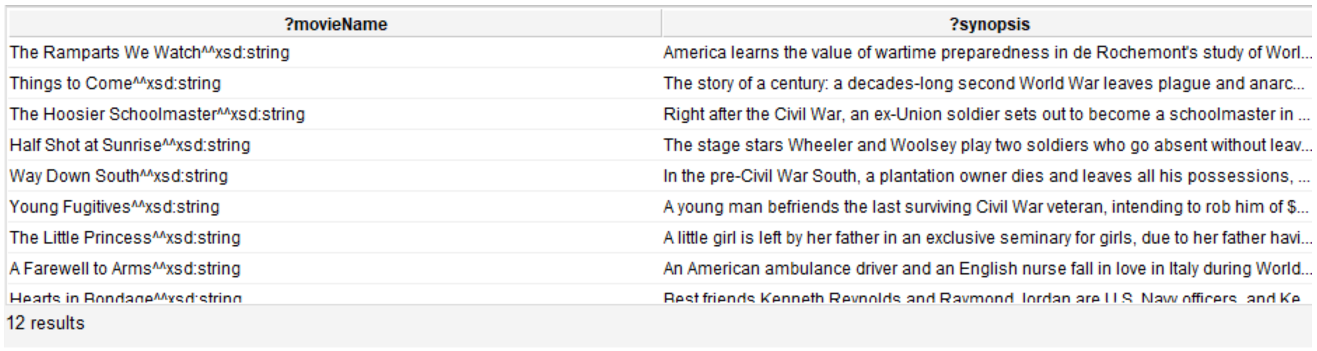
Fig. 19. Transformation Rules



?movieName	?director
The Beloved Rogue^^xsd:string	Alan Crosland^^xsd:string
Ella Cinders^^xsd:string	Alfred E. Green^^xsd:string
The Farmer's Wife^^xsd:string	Alfred Hitchcock^^xsd:string
Murder!^^xsd:string	Alfred Hitchcock^^xsd:string
Robin Hood^^xsd:string	Allan Dwan^^xsd:string
The Iron Mask^^xsd:string	Allan Dwan^^xsd:string
A Romance of the Redwoods^^xsd:string	Cecil B. DeMille^^xsd:string
Steamboat Bill, Jr.^^xsd:string	Charles Reisner^^xsd:string
The Last of the Mohicans^^xsd:string	Clarence Brown^^xsd:string

43 results

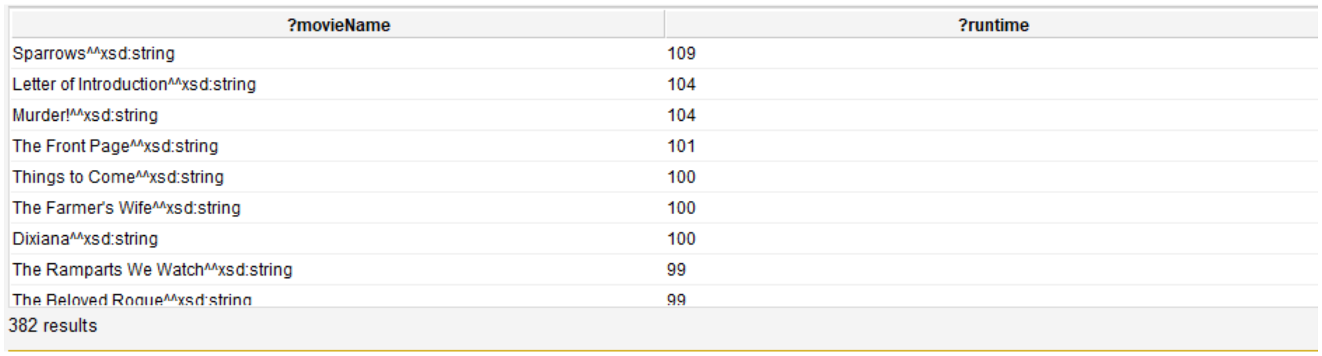
Fig. 20. Results of query 1



?movieName	?synopsis
The Ramparts We Watch^^xsd:string	America learns the value of wartime preparedness in de Rochemont's study of Worl...
Things to Come^^xsd:string	The story of a century: a decades-long second World War leaves plague and anarc...
The Hoosier Schoolmaster^^xsd:string	Right after the Civil War, an ex-Union soldier sets out to become a schoolmaster in ...
Half Shot at Sunrise^^xsd:string	The stage stars Wheeler and Woolsey play two soldiers who go absent without leav...
Way Down South^^xsd:string	In the pre-Civil War South, a plantation owner dies and leaves all his possessions, ...
Young Fugitives^^xsd:string	A young man befriends the last surviving Civil War veteran, intending to rob him of $...
The Little Princess^^xsd:string	A little girl is left by her father in an exclusive seminary for girls, due to her father havi...
A Farewell to Arms^^xsd:string	An American ambulance driver and an English nurse fall in love in Italy during World...
Hearts in Bondage^^xsd:string	Best friends Kenneth Reynolds and Raymond Jordan are U.S. Navy officers, and Ke...

12 results

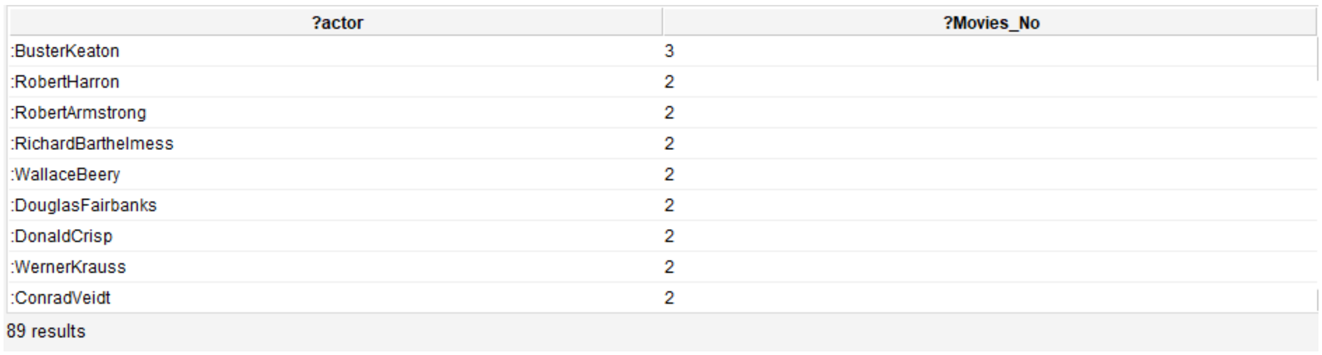
Fig. 21. Results of query 2



?movieName	?runtime
Sparrows^^xsd:string	109
Letter of Introduction^^xsd:string	104
Murder!^^xsd:string	104
The Front Page^^xsd:string	101
Things to Come^^xsd:string	100
The Farmer's Wife^^xsd:string	100
Dixiana^^xsd:string	100
The Ramparts We Watch^^xsd:string	99
The Beloved Rogue^^xsd:string	99

382 results

Fig. 22. Results of query 3



?actor	?Movies_No
:BusterKeaton	3
:RobertHarron	2
:RobertArmstrong	2
:RichardBarthelmess	2
:WallaceBeery	2
:DouglasFairbanks	2
:DonaldCrisp	2
:WernerKrauss	2
:ConradVeidt	2

89 results

Fig. 23. Results of query 4