MOVIE DATA ANALYSIS AND VISUALIZATION USING SEMANTIC WEB ENGINEERING

Krishna Chandu Akula, Mariya Varghese, Hari Krishnan Puthiya Veetil, Naga Sai Krishna Gurram

***Abstract*— Now a days Movie is one of the trending topics everywhere. With the developed technology people these days are watching international language films in their home itself. With the increase in the streaming platforms like Netflix, Amazon Prime, there has been a rapid growth in the people who watch movies and other digital content. Before watching a movie, it is necessary to know the ratings and reviews along with the cast and crew of the movie. Knowing the movie before watching helps the users to have an idea about it so that they can prepare their mind about what they are going to watch. Apart from this, having these kinds of analysis will help the film industry to know what viewers really want and interested in what kind of movies.**

***Keywords*—Movie, IMDB, Movie Ratings and Reviews, Cast and Crew.**

I. INTRODUCTION

Movies are one of the major aspects of entertainment that everyone will enjoy. It is important for the film makers to know what kind of movies people really like. It is very hard to predict the trend of genres of movies. Our analysis of Movies [1] and Visualization of the data about movies will help both viewers to have better experience before watching and film makers to know the people better in making new and entertaining movies.

The data required for the analysis involved in this project will be obtained from various Movie Databases such as IMDB [2], BookMyShow [3] and Kaggle [4]. All the data that has been gathered will be covered into (CSV / JSON) triple data format [5]. In this project protégé tool will be used to create a web ontology language OWL [6] to map the relations between the collected data from different resources. SPARQL [7] will be used as a query language to retrieve the data according to the end user selection and input. This application will have an user friendly Interface to fetch the inputs from the user and also to display the results and visualization of the data and its trends.

Every individual in the team will research and gather the required data from various web resources that are available. Once everyone gathers all the data required, everyone in the team will participate in a brainstorming session to finalize on dividing the gathered dataset into three parts format which are supposed to be used for this application. Ontology creation and data conversion will be done by Naga Sai Krishna and Mariya, linking of data and SPARQL queries designing will be done by Krishna Chandu and Hari Krishnan.

This project can be extended to predict the movie results which helps the film makers to make decisions. To help everyone to understand the trends of movies that are most viewed by people at that time or any genre. In addition to this, the visualization of data helps in better understanding of the trends that are being predicted. Maintaining the reviews and ratings also helps in developing patterns to help the critics for better understanding of the movies and their respective ratings.

II. PROBLEM DEFINITION

In the United States, Cinema is a Multi-Billion-dollar industry. 1000s of films will be release every year. Each film will have a budget of billions of dollars. In some cases, advertising makes a lot of portion in the movie budget. Sometimes the moviemakers might need to face a huge loss. It is said that the success of the movie depends on the cast sometimes and sometimes it is depended on the story and production values. Because of various factors moviemakers cannot predict the success of the movie before releasing it into the theatres. But using Semantic Web Engineering there is somehow a way to analyse all the movies and see the patterns or trends about the success of a movie. This might help the movie makers to produce the movie better and to the viewers in getting the content better. Having a huge amount of data is an advantage to analyse and visualize the data. Doing this work manually might be a tedious task.

The dataset that will be used in this project will be retrieved

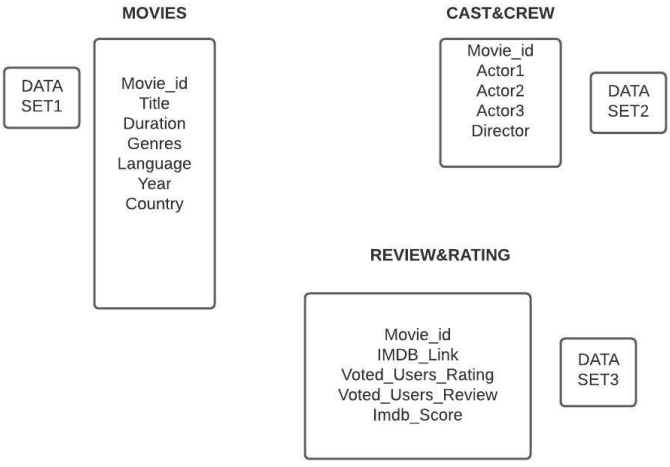


Figure 1: Vertically Fragmented Dataset

from the most popular Movie Database IMDB. It offers free API to everyone who needs the data. Thus, the dataset is free. It consists of 17 columns. The entire data will be divided into 3 fragments based on the Cast, Crew and Movie information as

shown in Figure 1. After fragmenting the data vertically, they will be transformed into OWL triple data format in order to provide a relationship between all the data. Once all the data is corelated to each other, queries will be designed to fetch the user required input using SPARQL.

III. Related Literature

This project will be almost like one of the most popular websites called IMDB [8]. This website provides insights about the movies and provides the viewers an opportunity to give ratings and reviews to the movies that they have watched. Gathering this kind of data from the users will help the viewers to know about the movie and helps the film makers to know what kind of movies were most liked by the viewers and critics. There are lost of projects based on this data to know the trends and patterns that have been formed to predict the success of a movie.

Another popular database for the movie data to play with will be The Movie Data Base (TMDB) [9]. Just like the IMBD this website also provides public data for analysis and prediction of the success of a movie. This Database is a crowd sourced and funded DB. So, any movie data collected here will not have any censorship. People can use them freely and can do anything with the available data. One of the projects [9] that are based on TMDB is to categorize the movies based on genre and year, revenue and genre, revenue and year, budget and genre. By creating visualizations like these the film makers will know how much percentage of the viewers liked what genre of films in any particular year.

One other interesting project [10] that can be discussed is combining IMDB and Rotten Tomatoes websites data. Combining data from two different websites can be interesting. Big data tools like Spark and Hadoop will be used to maintain and analyse the data. Creating patterns out the combined data will provide diverse and reliable outputs.

In this project a query technology called SPARQL will be used to fetch the data according to the user inputs. There is one similar project [11] that used the same technology to fetch the movie data. In this project the data is cleaned and analysed using Python and once the data is all set, then it will be queried based on the patterns that needs to be generated.

Ontology based Opinion mining [12] is one other related research to this project. Ontologies are created based on the movie data gathered from various sources. The approach will be similar to this research paper. But in this project the ontologies will be created to corelate and analyse the data.

IV. Approach

As our problem statement is to come up with an application that uses the existing semantic web engineering technologies and represents the movie data in a beautiful visualization based on the query user perform on the appication. Some of them are entering a movie name and getting the cast, crew of the movie and reviews and ratigs of the movie. Other being, querying the cast or crew to get the movies he acted and ratings of the corresponding movie. This helps for users who would like to watch movie of specific artists or with specific genres and reviews or ratings they would like to. We approached this problem in a Rational Problem Solving manner. We did this because Rational decision making leverages data, logic and analysis instead of subjectivity and intuition help solve a problem. But using Semantic Web Engineering there is somehow a way to analyse all the movies and see the patterns or trends about the success of a movie.

Data is always the crux of solving a problem in any context. So we explored several data sets from various websites, database servers and other sources. As we finally found some data sets which best suits our needs and matches our problem definition, we continued solving our problem in ontology context. As the main key technologies of semantic web engineering are defininig resources and presenting them in Ontologies. We came up with a structured way of defining the classes , object properties , data properties and tried to map with the data set we have for better compatibility. We used protégé tool for designing and creating classes, object properties, data propertites, relation ships etc. Some tools such as WebVOWL helped us in visualising the owl by providing graphical depictions for elements of the OWL that are combined to a force-directed graph layout representing the ontology.

As we are very concrete in our problem definition and have the data sets which suits the problem definition which was enhanced by creating OWL files, the next steps of approaching would be linking these data sets with OWL files and jumping over to deploying these instances in server which supports the SPARQL Queries. At the end these queries will be triggered from the queries given by user through the Web GUI Application and the results from the queries are captured by the application and presented in several visusalisation formats best suited depending on the context.

V. HIGH LEVEL DESIGN

The work flow of system at high level is mentioned below In the Figure 1. This Application contains of three high level components Front End , Back End and Cloud servers such as Micro soft Azure, Google Cloud and Amazon AWS.

Front End:
Front End consists of GUI Application which enables the user to enter inputs such as City, Actor or Ratings based on which user can filter of.

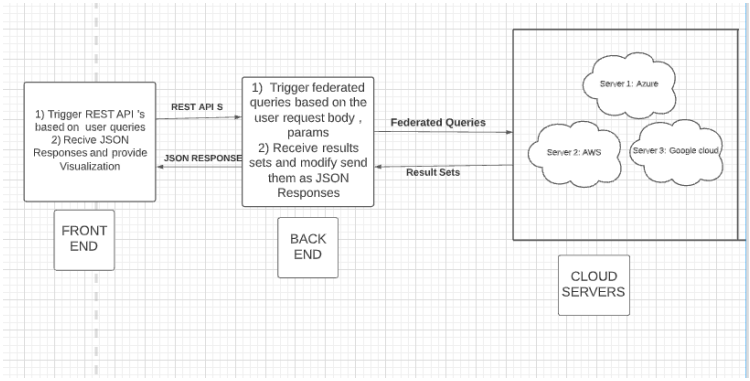


Figure 3: High level System Design

Once the user enters the inputs, the corresponding REST API's which are already deployed on backend and running on server will gets triggered. Once the front end recieves the responses it visualizes the data in several formats such as graphs, charts which represent the data in better format.

Back End:
The backend will contain a set of REST API'S which are responsible to get routed by specific user requests from front end. These REST API contains several definitions which will trigger the federated queries to several cloud servers such as AWS, Azure, Google Cloud

Cloud Servers:
Linked Data will be deployed in several cloud servers such as AWS, Google cloud, Azure. The linked data generated here is fusion of owl files and instances of all the three data sets we use for this project.

Three cloud servers expose API's which will be used by the backend code to trigger the corresponding federated query to the respected cloud server. This distribution into different cloud servers helps us to maintain our linked data in different distributed locations based closest to the best queries from corresponding locations

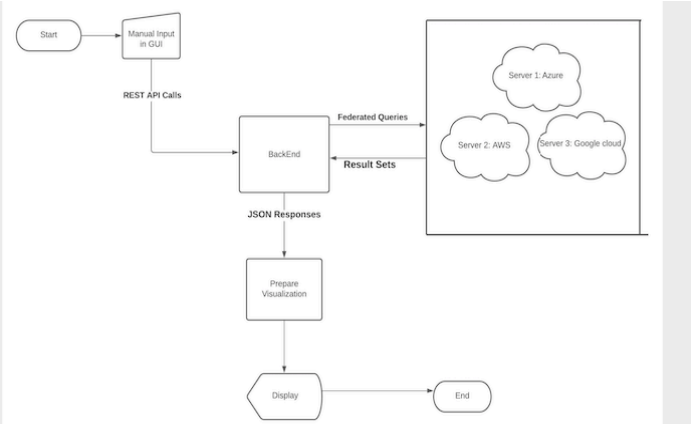


Figure 2: Overall System Work Flow

The above diagram depicts the over all system work flow starting from the Front end all the way through back end , cloud servers and backwards.

VI. ONTOLOGY DESIGN

During the Ontology design phase , the data from the existing cleaned data in spreadsheets was converted to meaningful triples. First the entire data was analyzed, so as to what rules and restrictions must be applied and what is the best way they can be linked which meets the end goal of the project. These restrictions and rules, validations are represented for creating ontology in Owl. This forumulation is used to create 3 different Owl files (1 for each spreadsheet) by using the tool Protégé. Each column from every spreadsheet was defined as class in Owl. For these classes data properties and object properties were defined with necessary restrictions. Hermit reasoner was used to validate all the rules and restrictions on the data in these Owl files was semantically logical and possible.

Based on the dataset that is collected, the ontology is designed. For example, the ontology for dataset-2 will have the classes Movie_id, Actor_1, Actor_2, Actor_3, Director. There are some Object properties defined which are is_id_of, has_id, is_actor_1_of, is_actor_2_of, is_actor_3_of, is_director_of, has_actor_1, has_actor_2, has_actor_3, has_director. Each property will have Domain and Range. For Instance is_actor_1_of will have Movie_id class as domain and Actor_1 class as Range. Apart from this some data properties as well. The data properties include is_actor_1, is_actor_2, is_actor_3, is_director. These data properties will have Domain and Range

as well. For instance is_actor_1 property will have Domain as Actor_1 class and Range as xsd:string.

VII. DATA COLLECTION AND PROCESSING

The raw data was collected from the website 'data.world' titled "imdb-movie dataset.csv" which has arounf 5000 rows and 15 columns. This raw data had statistics regarding the movie details such as movie name, cast and crew, rviews and ratings. These raw statistics gives us a good foundation for further manipulation required in this project.

So ,the data was initially distributed over 3 spreadsheets with columns divided as mentioned in the schme earlier (in figure 1) based on the logical data set they belong to . These spreadsheets were first merged into a single spreadsheet to get a picture at the size of the data. Then this data was divided into 3 spreadsheets for reducing redundancy and classified based on the schemas.
Each dataset is explained below:

Dataset-1:

This dataset has 8 columns which are movie_id, movie_title, duration, genres, language, title_year, country. This dataset will have the details about the movie itself. This doesn't include any cast or crew.

Dataset-2:

This dataset has 4 columns movie_id, actor_1_name, actor_2_name, actor_3_name, director_name. This dataset consists of only cast and crew details and nothing else.

Dataset-3:

The final dataset that is divided will have 5 columns which are movie_id, movie_imdb_link, num_user_for_reviews, num_voted_users, imdb_score. This dataset holds all the information of the movie after release such ars inks reviews and ratings.

One thing that is observed in these datasets are there is one common column which is movie_id which acts as a primary key and foreign key column. The detailed view of how the ontologies have been divided and designed will be shown in the next section.

VIII. VISUALIZATION OF THE ONTOLOGY

As discussed in the previous section, the dataset has been divided into 3 parts. The Figure 4 shows how the divided part of the dataset is turned into an ontology. In this first part of the ontology we have classes Movie_id, Title, Duration, Genres, Language, Title, Year, Country.

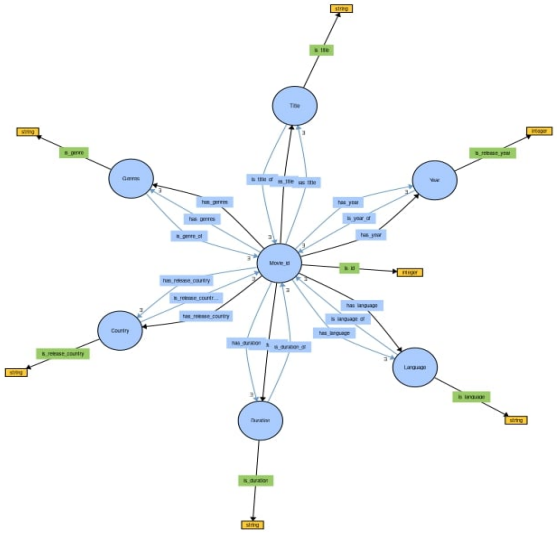


Figure 4: Visualization of Dataset-1 Ontology

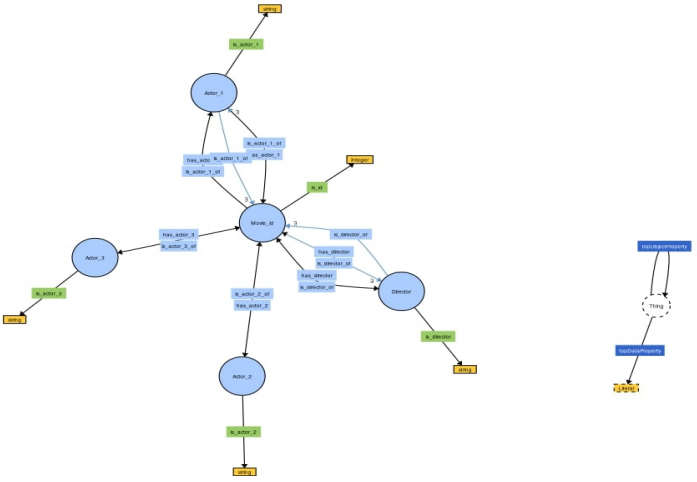


Figure 5: Visualization of Dataset-2 Ontology

The above figure depicts the ontology of the Dataset part 2. In this Data set classes such as Movie_id, Actor_1, Actor_2, Actor_3, Director were included.

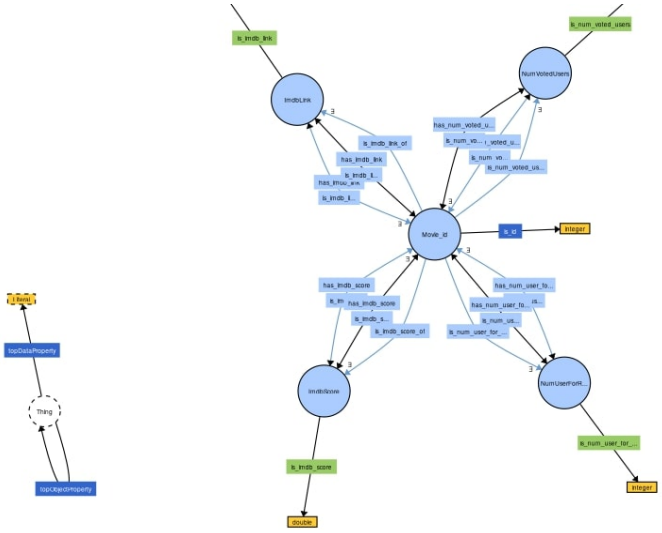


Figure 6: Visualization of Dataset-3 Ontology

This is the final part that is divided from the main Dataset that is being used in this project. This part has classes named Movie_id, ImdbLink, NumUserForReviews, NumVotedUsers, ImdbScore.

These three ontologies are linked together to form a new ontology having all the classes together. Links were made accordingly in the Protégé tool. Movie_id is the Class that is being linked in all the parts of the ontologies. This Movie_id class will have a property called Equivalent Class in all the Parts of the ontologies which makes links between then to form a new linked ontology.

IX. IMPLEMENTATION

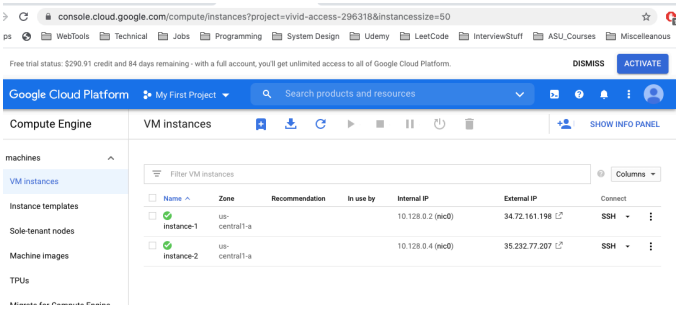
Data Generation and Mapping:

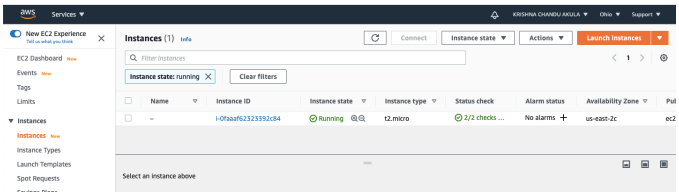
The data collected is available as Excel sheets and OWL files are created using Protégé tool. Before linking the OWL file and excel sheet, the data is analysed and cleaned to avoid the errors that might need to face in the future.

After cleaning the data, using Open refine tool both the OWL files and cleaned excel sheets were loaded. Using the open refine tool basing on the OWL class structure, data is mapped accordingly. All the classes, relations between them, object properties and data properties of each class is defined in the open refine tool. After linking all the raw data and OWL files, the open refine tool generated an RDF which is used in the cloud servers to query the data.

Cloud Servers:

Fuseki Servers are deployed on three different cloud virtual machines, two are setup on Google cloud compute engine and 1 is set on AWS console.





Construction of Application:

The application was build using MVC (Model-View-Controller) Architecture. The application architecture is explained in the Figure 3.

Backend:

Backend is built using the tech stack Java/Spring framework.We made it is a maven project for easy portability.We designed three REST API's

i) http://localhost:8080/getMovieInfo?name="MovieName"
ii) http://localhost:8080/getMovieWithMinRating?rating=Minrating
iii) http://localhost:8080/getDirectorMovieInfo?name=”MovieName”

RestAPI getMovieInfo is responsible for getting the movie info , rating info and the crew info . This API accepts the name as a parameter and triggers total of 3 queries out of which two are federated queries, where we get a movie id out of the name and use this id to retirieve details of both the reviews and crew info.

getMovieWithMinRating is responsible for getting the movie titles which has the minimum ratings x, where x is the rating

parameter value.It returns all the movie titles which has minimum ratins. For example if user selects min rating as 6, we display the 25 movies which have rating greater than 6.Apart from this this api also passes all the count s of movies which has minimum rating 6. In detail it passes the number of movies which has ratings 6-7, 7-8, 8-9, 9-10, This info will be represented as graph in frontend.

getDirectorMovieInfo is responsible for getting all he movies which are directed by a specific director.This endpoint accepts a query parameter called name which should be passsed with the value of director name. This returns all the movie names directed by the corresponding director.

Frontend:

Frondend is built on ReactJS and Material UI Framework. The entire UI consists of a radio button group having three options which are used in the search criteria. After selecting one of the options in the radio button group. The user needs to search the movie based on movie name or director or minimum rating. The search is an auto suggestive drop down, the user might not need to remember the entire search string. After querying the search box, backend service gives a response in JSON format which will be shown as tables and bar charts whenever required.

X. SPARQL QUERYING

As we have a total of 3 REST end points, we will be end up having complex queries for each end point.For the end point getMovieInfo , This single end point is combination of two federated queries which will run on the cloud server where the rating, crews rdf files are loaded onto. The query passed to this server contains endpoints of the other cloud server where movie rdf files are loaded. Below are the 2 federated queries , 1 normal query of which a single end point conatins of. *More queries can be found in the backend source code, Only 1 is presented here.*

Queries Prepared for /getMovieInfo EndPoint:

Federated Query 1:

This below federated query (Figure 7) is responsible for getting the mid1 by hitting the service at 18.191.196.300 and gets the mid1 which matches the movie name ‘Baby boy’ . Then we get rating info attributes imdblink, imdbscore, numusers etc based on the mid1 which is output of inner query.

```
PREFIX movie3: <http://www.semanticweb.org/imdb/movie3.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX movie: <http://www.semanticweb.org/imdb/mve.owl/>

SELECT ?mid1 ?imdblink  ?imdbscore ?numusers ?numvoted
  WHERE {
    ?movieid movie3:is_id ?mid1.
    ?movieid movie3:is_imdb_link ?imdblink.
    ?movieid movie3:is_imdb_score ?imdbscore.
    ?movieid movie3:is_num_user_for_reviews ?numusers.
    ?movieid movie3:is_num_voted_users ?numvoted.

      SERVICE <http://18.191.196.100:3030/Movie3/query>{
       SELECT ?mid1 WHERE {
            ?movieid movie:is_id ?mid1.
            ?movieid movie:is_title ?title1.
      FILTER (CONTAINS ((?title1), "Baby Boy"))
            }
  }
}
LIMIT 25
```

Figure 7: Fedreated Query 1

Federated Query 2:

This below federated query (Figure 8) is responsible for getting the mid1 by hitting the service at 18.191.196.300 and gets the mid1 which matches the movie name ‘Baby boy’ . Then we get crew info attributes actor1, actor2, actor3, director etc based on the mid1 which is output of inner query.

```
PREFIX movie1: <http://www.semanticweb.org/imdb/movie2.owl#>
PREFIX movie: <http://www.semanticweb.org/imdb/mve.owl/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>


SELECT ?actor1  ?actor2 ?actor3 ?director
  WHERE {
    ?movieid movie1:is_id ?mid1.
    ?movieid movie1:is_actor_1 ?actor1.
    ?movieid movie1:is_actor_2 ?actor2.
    ?movieid movie1:is_actor_3 ?actor3.
    ?movieid movie1:is_director ?director.

      SERVICE <http://18.191.196.100:3030/Movie3/query>{
       SELECT ?mid1 WHERE {
            ?movieid movie:is_id ?mid1.
            ?movieid movie:is_title ?title1.
      FILTER (CONTAINS ((?title1), "Baby Boy"))
            }
  }
}
      LIMIT 25
```

Figure 8: Fedreated Query 2

The below query (figure 9) is responsible for getting all the movie attributes given a movieid.

```
PREFIX movie1: <http://www.semanticweb.org/imdb/mve.owl/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <undefined>
SELECT ?title  ?duration ?mid ?genre ?country ?language
WHERE {
  ?movieid movie1:is_id ?mid1.
  ?movieid movie1:is_title ?title.
  ?movieid  movie1:is_duration ?duration.
  ?movieid movie1:is_genre ?genre.
  ?movieid movie1:is_language ?language.
  ?movieid movie1:is_release_year ?year.
  ?movieid movie1:is_release_country ?country.

         FILTER (CONTAINS((?title), "Turbo"))

  }
```

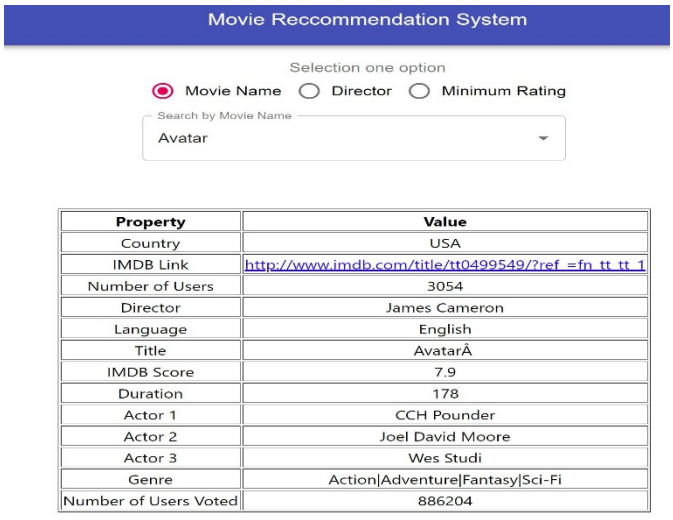
Figure 9: Query 3

All the query responses of Federated Query 1, Federated Query 2, Query 3 will be merged and will be made as a single json response which contains the entire movie info attributes from all the three different datasets such as movie_id, movie_name, actor_1, actor_2, director, imdbscore, numusers, numvoted etc.

XI. USE CASES FOR TESTING

There are total 4 use cases to test this application. Each use case is explained along with figures below.

Use case 1:



Property	Value
Country	USA
IMDB Link	http://www.imdb.com/title/tt0499549/?ref_=fn_tt_tt_1
Number of Users	3054
Director	James Cameron
Language	English
Title	AvatarÃ
IMDB Score	7.9
Duration	178
Actor 1	CCH Pounder
Actor 2	Joel David Moore
Actor 3	Wes Studi
Genre	Action|Adventure|Fantasy|Sci-Fi
Number of Users Voted	886204

Figure 10: Use case 1

In this use case the user searches for the movie information based on the movie name. When the user starts to type the movie name, the dropdown will be auto populated based on the search string. Once the query is successful the fetched data will be displayed in the form of a table as shown in the figure 10.

Use case 2:

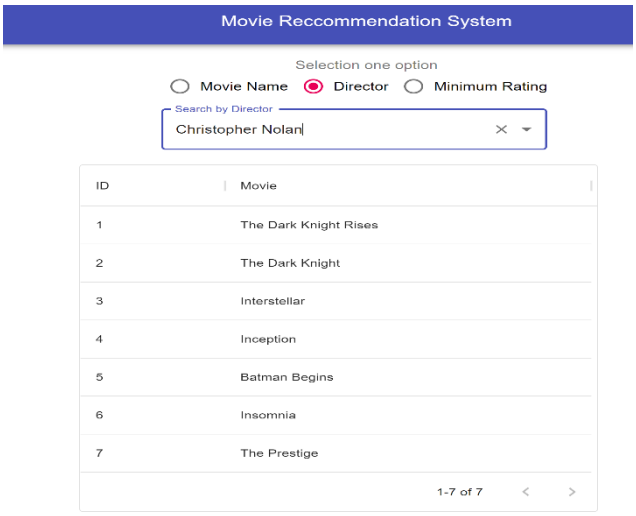


Figure 11: Use case 2

In this use case the user searches for list of movies based on the director name. When the user searches for a director it fetches the data from the fuseki servers and once the JSON data received successfully it displays all the movie names of that director. The table is sortable and paginated having 7 rows per page as shown in the figure 11.

Use case 3:

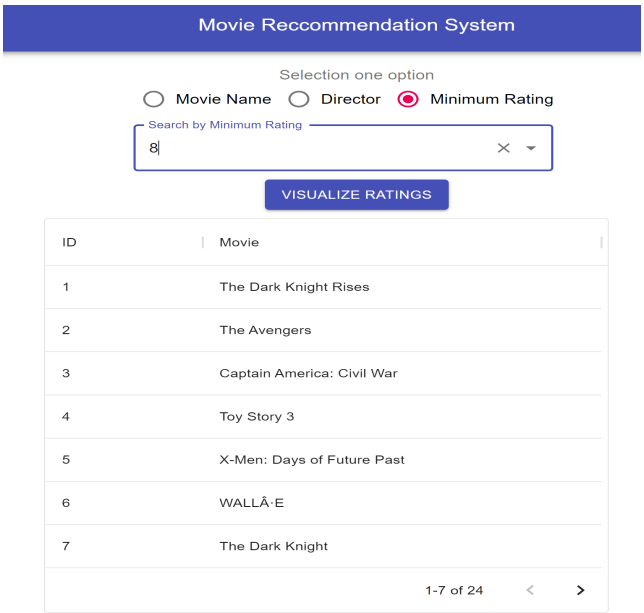


Figure 12: Use case 3

The user might want to search based on ratings. In this use case the use will give the minimum rating as an input to the application. For instance, if the user gives minimum rating as 5, then all the movies having rating 5 or more will be displayed. The data displaying will be similar to the use case 2 as shown in the figure 12.

Use case 4:

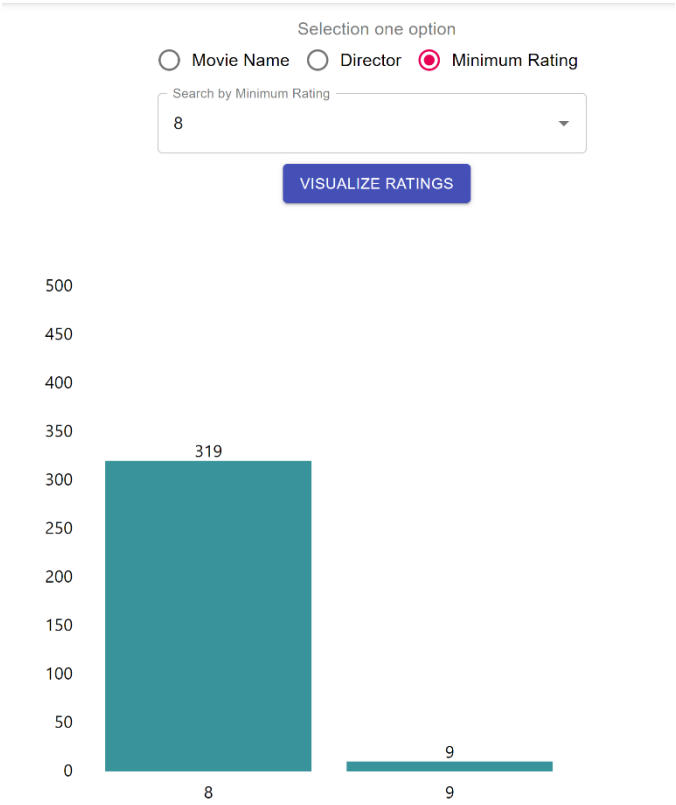


Figure 13: Use case 4

This is the last case that user can fetch the data. After displaying the data in use case 3, a button will be popped up saying Visualize Ratings. When the user clicks on that button a Bar Chart will be displayed as shown in the figure 13. The chart consists of ratings and number of movies for that respective ratings.

XII. Evaluating Application

This project has data over 5000 rows spreading over 3 different cloud locations. Since the federated query is based on the data from all the 3 cloud servers it took around 300 seconds to complete. The data obtained from the servers are verified manually from the raw data (Excel sheets) and RDFs as well.

XIII. Challenges Faced

One of the major challenges faced for this project is to host the data in 3 different cloud hosting providers. Unfortunately, during the implementation phase Microsoft Azure had some issues with the fueski server and was not able to give permissions to access publicly. The that part was hosted in another AWS cloud instance making it physically available in different location. Once it was hosted in AWS everything went fine without any breaks or issues.

XIV. Future Scope

This project can be further extend to include more datasets from different open source websites to increase the accuracy. One other feature that can be extended will be to include more fields for the search criteria such as searching based on actors or actresses. Last feature that can be implemented is to include more visualizations regarding the cast and movie information. For instance director Christopher Nolan has 15 movies rated 8 and 30 movies rated 9.5. Having these kinds of projects will always helps viewers and producers to produce more effective content.

XV. References

[1] https://dspace.mit.edu/handle/1721.1/113502
[2] https://www.imdb.com/interfaces/
[3] https://in.bookmyshow.com/
[4] https://www.kaggle.com/stefanoleone992/imdb-extensive-dataset
[5] https://www.w3.org/wiki/JSON_Triple_Sets
[6] https://www.w3.org/OWL/
[7] https://www.w3.org/TR/rdf-sparql-query/
[8] https://medium.com/analytics-vidhya/exploratory-data-analysis-of-the-imdbs-movie-database-from-a-data-scientist-perspective-b1dfd7455d1
[9] https://medium.com/@onpillow/01-investigate-tmdb-movie-dataset-python-data-analysis-project-part-1-data-wrangling-3d2b55ea7714
[10] http://rstudio-pubs-static.s3.amazonaws.com/336722_2193716117584b63a2a6ebb837217d85.html
[11] http://www.snee.com/bobdc.blog/2015/08/querying-machine-learning-movi.html
[12] https://doi.org/10.1007/978-3-642-10488-6_22

XVI. Preliminary Data Sets

1. https://data.world/himan/imdb-movie-dataset/workspace/file?filename=movie_data.csv
2. https://datasets.imdbws.com/
3. https://www.kaggle.com/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews
4. https://www.kaggle.com/tmdb/tmdb-movie-metadata