Movie and TV Series Recommendation using Web Ontologies

Sparsh Kotriwal
Software Engineering)
Arizona State University
Tempe, Arizona
skotriwa@asu.edu

Shilpi Hiteshkumar Parikh
Software Engineering
Arizona State University
Tempe, Arizona
sparik11@asu.edu

Kunj Viral Kumar Mehta
Software Engineering
Arizona State Univeristy
Tempe, Arizona
kmehta25@asu.edu

Nimil Sanjeev Shah
Software Engineering
Arizona State University
Tempe, Arizona
nshah55@asu.edu

Piyush Jayawant Rewatkar
Software Engineering
Arizona State University
Tempe, Arizona
prewatka@asu.edu

***Abstract*—The amount of data present on the internet today is so much that the user is often confused or overwhelmed over the availability of choices. The magnitude of data present also makes it time consuming and does not aid quick decision making. Recommendations come into picture when a system could read through a large amount of data and provide user with a relevant information. This paper aims to develop a recommendation system for movies and TV series based on the input from the user's liking. With the vast quantity of movies and TV series released every day, it has become increasingly difficult for one to decide what to watch. The software we would develop will do a semantic analysis on the dataset and the users input and provide recommendations. For the develop recommendation system, we will be using different attributes from Netflix Movies and TV Shows dataset from Kaggle and MovieLens dataset from GroupLens.**

***Index Terms*—Semantic Web, Ontologies, Dataset, RDF, Recommendation Systems**

I. Introduction

Our software serves as a recommendation system for movies and TV/Web series on the netflix platforms, we take advantage of linked data in semantic using OWL/RDF using the movies of the user's choice. The tags of the inputs from the user is matched with the movies in the dataset and a similar movie is returned as recommendation. Currently, we plan to use the Netflix Movies and TV Shows available on Kaggle which has columns like type which defines whether its a movie or TV show, the title of the show, director information, cast, country, release year, rating, duration, listed in which defines the genre of the movie or series and a short description. We can make use of listed in , description or combination of several columns to decide which movie/series a user would like based on the input they provide.The dataset has 8800 data points which would serve as a large corpus of dataset for recommendation.

People often prefer to watch movies of a specific director and that can be found easily using ontology developed and the RDF. For movies/series of a particular genre we can use the listed in attribute. With linked data in picture we can use a combination of several columns like listed in and director together which could be used to find all Drama movies by Stephen Spielberg as an example. The implementation of this recommendation system makes it really easy for the user to make a movie decision.

II. Problem Definition

The importance of suggestions in today's age of media, where consumers have virtually endless options because to services like Netflix and Amazon, cannot be overstated. Recommendations can be made for a variety of services, including video on demand and online shopping. Recommendations are often used to display comparable products that the consumer uses or that are frequently used in the consumer's region. In order to make recommendations for movies that our users haven't yet viewed, we will estimate their ratings. Then, depending on the anticipated ratings, movies are indexed and recommended to users. In order to do this, we will anticipate future ratings by using user feedback and movie history.

III. Literature Survey

This paper [1] demonstrates how semantically enhanced methods can improve recommendation systems' accuracy. The objective of this paper is to present a personalized recommendation system based on ontologies that give semantic applications personalized recommendation services based on movie representations, profiles of users, and cast profiles. As part of personalization, domain ontologies are used: on the one hand, by applying a domain-based inference method, the user's interests are modeled more logically and accurately; on the other hand, our recommendation approach uses SPARQL queries, which give more suitable filtering capabilities [2]. Experiments conducted on the MobieLens, IMDB movie dataset show that semantics-based recommenders can contribute to improving the quality and accuracy of recommendations by adding additional knowledge.

Szomszor in the paper [3] has proposed a hybrid movie recommendation system that utilizes features of folksonomy. This paper explains how tagging sites with large-scale community data can enrich a reccomendation system by creating a model using Semantic Web with valuable knowledge.User profiles can be created using user's level of interest based on folksonomy generated tag clouds avaliable on the Internet Movie Database [4]. Experimentations is done whether we can increase the effectiveness of recommender systems by giving them access to larger amountsof data, which necessitates the integration of relevant data from heterogeneous sources [5] in this paper.It has been contended that tackling the information implanted in folksonomies can prompt structure shallow ontologies that are more responsive to information change after some time and thereby reducing the overhead in building the systems using Semantic Web only.

According to this paper [6] in every domain the user's satisfaction and acknowledgement is of prime importance. The movie recommendation systems are built commonly based on the web data that we have. These recommendations succeed in generating personalised recommendations based on the data gathered through the web but they lag in providing precise recommendations. The primarily cause of this problem is lack of domain knowledge. The recommendation performance can be enhanced greatly by integration of web data with the domain information. Along with this if we increase the similarity measure among the recommendations, the performance accuracy improves as well. In this paper [7], the authors have proposed a new ontology which is based on semantic similarity measure to improve the performance of movie recommendation system. For experimental evaluation of the proposed recommendation system in this paper, MovieLens dataset (1M) [8] has been used. This dataset contains approximately 6040 users and 3883 movie names. For all of these movies, the users have provided ratings. 1000209 ratings have been noticed in the dataset. The proposed approach comprises of 4 phases in the system. The first phase is Construction of domain ontology in which there are multiple levels first being the Movie. This have several other attributes and data-types of it. The second phase is data processing which involves cleaning of the raw data into acceptable format for further steps. The third phase is pattern discovery which computes the web patterns using all the rating data that have been collected. After this the system is test for pattern analysis which is the final step in the system. For evaluation of the system, precision, recall and F-measure metrics are considered. Compared to the existing available movie recommendations, this system gave 76% accuracy.

According to the authors of [9], the semantic gap issue in multimedia retrieval systems has received a lot of attention. In the published paper, the authors have proposed an approach where in test a new method for suggesting movies that pulls low-level Mise-en-Scène information from multimedia content and combines it with high-level information from the wisdom of the community. The type of recommendation system developed was a content-based recommendation system. The authors used 3 variations of the algorithm. These 3 variations varied by the types of features that the recommendation system was tested on. The features such as conventional movie attributes, Mise-en-Scène features and a hybrid model which combined both of these features was taken into consideration. The paper took two approaches one of which was system-centric evaluation and the other being user-centric online experiment involving 100 users. The author paper did four kinds of result analysis relevance, diversity, novelty and satisfaction and achieved grater than 90% metrics for coverage and novelty which is a concludes how successful the experiment was.

The paper [10] talks about a generalised recommendation system that can help users in an arbitrary system get more out of the tools and technologies they are leveraging in order to obtain more relevant results. The paper argues that Linked Data can be utilized over conventional recommendation system already in place to gain more accurate and varied information that would be contextually useful to the user. The paper highlights "feature selection", i.e., making use of only the most relevant subset of the supposedly large and convoluted web of linked data that would be available to the recommendation system, and displaying useful results, while using linked data summarization to achieve this standard. The authors compare an ontology-based feature selection mechanism to a legacy system and showcase the results.

In the aforementioned legacy recommendation systems, the most relevant dimensions were usually selected by actual human beings who were "domain experts." In this project, we aim to formulate a recommendation system for movies and TV series. The paper discusses two approaches to feature selection, one based on summarizing the linked data and then selecting the features, and the other iterates on each instance in the data set. The former uses the ABSTAT framework [11] for feature selection by filtering out the properties from the Linked Data summary. The latter utilizes the previously discussed legacy statistical approaches to extract the top-N features. The authors compare several approaches and conclude that Information Gain yielded the best results in terms of its similarity in the domain of statistical implementation of feature selection and recommendation.

The authors run the two above mentioned feature selection approaches to three different data sets: movies, books, and music. The conclusion that is drawn by the authors is that the first approach, i.e., the ABSTAT framework-based Feature Selection yields more relevant results for the movies and books data set; however, it appears that the Information Gain approach yields better results for the music data set. At the same time, keeping in mind that the results of Information Gain were not found to be statistically significant in comparison to ABSTAT, the authors conclude that ABSTAT leads more contextually accurate results overall.

This paper is particularly helpful for our project topic since it makes an argument for the usefulness of Linked Data in Recommendation Systems in general, and more importantly, it tests the recommendation system that the authors developed in three contextually different systems: movies, books and music, each containing a different arrangement of feature points, thus solidifying its claim.

Of all the research works discussed above, none have touched upon the fact that in this current world, every user has a time requirement and constraint as well. A movie would be something that a person would watch in full but with TV series in the picture, not everyone would have the time to watch 5 seasons of a particular series. Considering this fact, our approach of movie/series recommendation would also take into account, that a user is recommended movies which match their time constraints. A user can request for a highly rated web series with with less than 3 seasons which is thriller and action and has the actor as John Krasinki, then our algorithm should easily suggest that the best recommendation to watch is Jack Ryan.

IV. Data Collection and Pre-Processing

Massive amount of data are analyzed in the process of ”data mining” meaningful intelligence from them in order to assist enterprises in problem-solving, trend prediction, risk reduction, and opportunity discovery. Data mining is similar to actual mining in that both involve sifting through vast amounts of content in search of valuable resources and elements.In order to address problems, data mining also entails building connections and identifying patterns, anomalies, and correlations, which results in the production of useful information. Data mining is a broad and diverse process with a variety of parts, some of which are even mistaken for data mining itself.

The first phase of Data Mining is Data Acquisition. We will be using the Netflix Movies and TV Shows[4] dataset in our project which contains CSV file with different features about Netflix Movies and TV Shows.Several features such as title, director, cast, country, duration etc are mentioned in the dataset. Further, it consists metadata for ratings and description of around 7787 movies and TV Shows. The features that our recommendation system will be using are type, title, director, cast, duration and description. Our first approach will revolve around recommending movies to users based on Title, Cast, Director and Duration. Further, we’ll be implementing an algorithm to suggest movies to users using the words in the Movies/TV Shows description in the system.

The second phase of Data Mining is Data Preprocessing Fix duplicate, missing, or corrupted data issues to format the data in before providing as an input to the recommendation system. The entire dataset was in raw format along with some missing values as well. We preprocessed the entire dataset in Python using the a built in module Pandas. Pandas is a famous Python Library basically used for data preprocessing. The entire dataset was transformed into series and dataframes so it can be easily manipulated. There were few null values in our dataset which were corrected by filling them with .fillna() function of Pandas.

After that all the duplicate values were dropped from the dataset using dropduplicates(). The entire data was now free from missing, irreleavnt and duplicate values. The final output

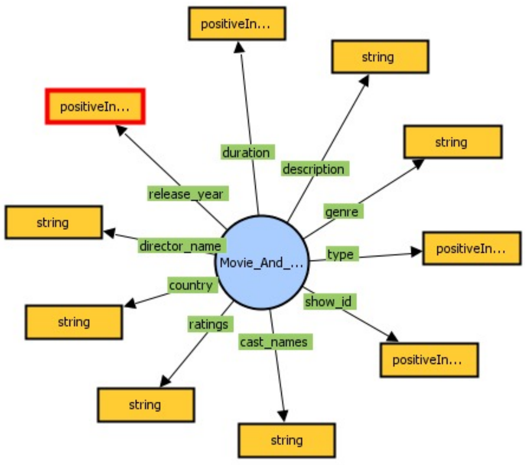

Fig. 1. Movies_And_Series_Ontology

we got was the required Movies and TV Shows dataset which can be used for further analysis and to validate it’s accuracy. Data inconsistency such as date_added columns was named to release_date in the dataframe.

V. Ontology Design and Visualization

We are using the following ontologies for our project:

- Movies and Series Ontology: This ontology consists of one single class called Movie_And_Series. This class has data properties like director_name, country, ratings, cast_names, genre, description which are all of string types. It further has release_year, show_id, type and duration which are kept as positive integers. The data values for this ontology has been fetched from the Netflix Movie and Shows dataset of kaggle. A visualization of this ontology prepared via using VOWL Plugin in Protege can be found in ”Fig. 1”.
- User_Ratings Ontology: This ontology also consists of one single class called User_Ratings. This class has data properties like movie_title and genres of type string, USER_ID and Movie_ID belong to type positive integers and Avg_user_rating of type float. The data value for this ontology has been fetched from the MovieLens dataset [8]. A visualization of this ontology can be found at ”Fig. 2”.

VI. Approach and High Level System Design

We are implementing movie/series recommendations based on the following three algorithms.

1. Recommendation using cast, director, country, rating, and genres as features.

2. Recommendation using the words in the movie/TV series descriptions as features.

3. Recommendation by the system on the top trending Movies/TV series.

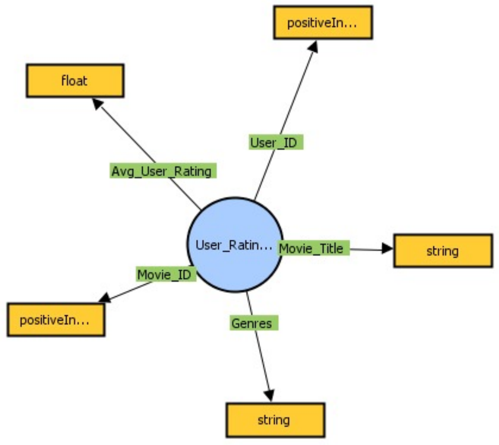

Fig. 2. User_Rating_Ontology

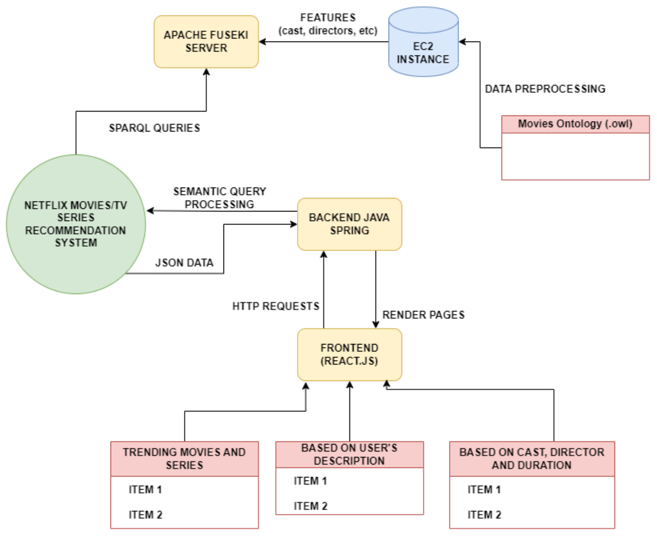

Fig. 3. High Level System Design

This type of recommendation comes under Content Based filtering. In the content-based approach, items with similar properties are recommended using a series of distinguishing features. The prediction in this approach is calculated based on ratings for similar items made by the active user. [6] Content-based methods aim to describe observed interactions between users and items by building models based on available ”features”. Our both ways to recommend movies relies on the Movies-Series Ontology. Using this ontology, we find movies or series in our database based on features like the cast, directors, countries, ratings, and genres, and suggest them to users who have never seen them. We will implement the front-end UI components in React.js and back-end part in Java Spring Boot, which will be used to query databases using SQARQL queries for certain HTTP requests that users request through the front-end. We are planning to use AWS EC2 to auto-scale our instances and S3 to store the movie dataset.

VII. Implementation Plan

Following are the steps that our team will be executing in the process of implementing this project:

1. Creation of ontologies
The movie recommender system ontologies will be created using Protégé

2. Creation of instances for the ontology Keeping the volume of instances needing to be created for this project in mind, we will be creating these instances using Cellfie, a Protégé plugin that can be used to quickly create large volumes of instances for a Protégé ontology file.

3. Data Cleaning and pre-processing
The data is used as a CSV file, which will be imported into python, and several pre-processing operations such as removing null values, dropping unnecessary columns, assigning data frames to relevant features, etc. is done using pandas and other related python libraries.

4. Building algorithm based on parameterized data set
Once the data is preprocessed and ready to use, we will start creating the algorithm for the recommender system keeping the high-level system design outlined in section VI in mind, viz. the parameters cast, director, country, rating, and genre.

Another algorithm will be simultaneously implemented for a recommender that uses key words as its features to generate recommendations.

5. Querying over the added data and algorithm
The data set is stored in an AWS EC2 server instance, from which the data can be queried using SPARQL queries based on the developed algorithm on an Apache Jena Fuseki server.

6. Developing the system and UI
Finally, the back-end of the system will be developed in Java Spring API, which communicates to the Recommender to fetch results, which are then given to the UI system, developed using React.js.

A. *Data Generation and Mapping*

We have partially used cellfie plugin for Protege to add data to our ontologies in xls format. For the User_Rating_Ontology we have added data manually from the dataset without the use of cellfie plugin [12]. The following rules have been used to map data into Movie_And_Series Onltology

”sheetName”: ”Sheet1”,
”startColumn”: ”B”,
”endColumn”: ”M”,
”startRow”: ”2”,
”endRow”: ”+”,
”comment”: ””,
”rule”: ”Individual: @B* Types: Show nFacts: show_id @C* (xsd:integer)Facts : type @D* (xsd :string)Facts: title @E* (xsd:string)Facts: director @F* (xsd:string)Facts: cast @G* (xsd:string)Facts: country @H* (xsd:string)Facts: release_year

@I* (xsd:integer)Facts: duration @K* (xsd:integer)Facts: genre @L* (xsd:string)Facts: descreption @M* (xsd:string)"

B. Testing

- Get movie recommendations based on Cast
- Get movie recommendations based on Genre
- Get series recommendations based on Genre
- Get recommendations based on User's Description
- Get recent trending movies and series

The outputs were verified with the existing results to validate the results.

C. Fuseki Serve Setup

We configured all the setup of Fuseki server to run our SPARQL queries. We installed the fuseki server by the Apache and configured it to our system. We run the fuseki server locally initially. The data instances could be easily uploaded using the Apache Fuseki Server UI. Once the data is uploaded, we will set the endpoint and source point for running the SPARQL queries. To run the queries, we define the set of prefix as well.

VIII. Front-end and UI

The Fuseki server was mapped to the AWS EC2 instance via the Fuseki interface, so we verified that SPARQL queries were working. Next, we tested the EC2 endpoint URL with the Postman app and identified the URL for the query. In order to display the results to the user we developed a front end using React.js. Whenever a button in the front end is clicked, we make an HTTP request to the endpoints hosted on AWS to retrieve the results. Additionally, users can search for specific movies in the loaded results.

IX. Evaluation and Results

The Netflix Recommendation App suggests movies and series to users based on genre, cast, description etc. Further this application also recommends movies and series based on the current trend to the users. This implementation was made using SPARQL queries and setting up the Fuseki server. The results have been tested and are fairly accurate with the required outputs desired. In the search option, the user can enter fields such as cast, genre, director etc and the we have been able to suggest movies or series based on these inputs. In this section we have added the UI designs of the app, along with different use cases to validate the results. Lastly, the entire application has been beautified with additional components such as pagination, hooks and single content modal.

X. Challenges/Limitations

We faced several challenges during the course of the project:

- We created the ontology based on the basis of the dataset and general knowledge of those column fields, we had to make changes to the dataset to add it to the ontology with the help of plugin.
- Dealing with missing data in datasets

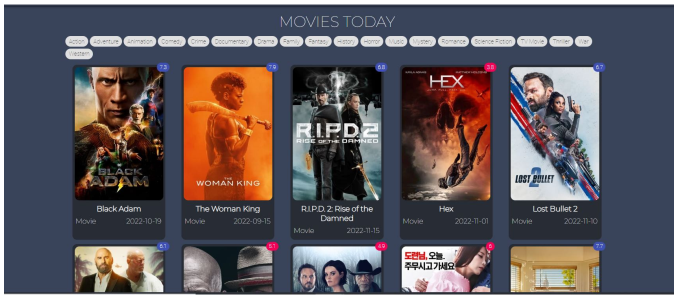

Fig. 4. UI Design for Movies Page

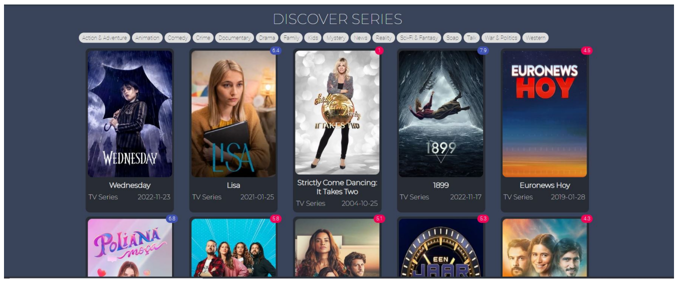

Fig. 5. UI Design for Series Page

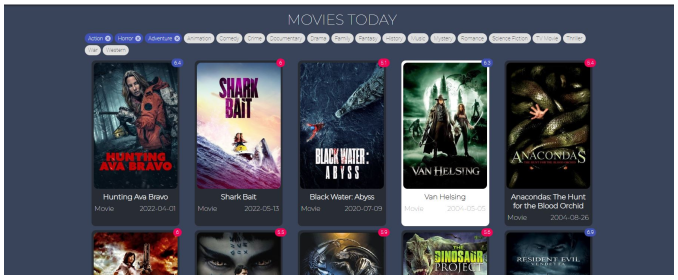

Fig. 6. Movie Recommendations based on Genre

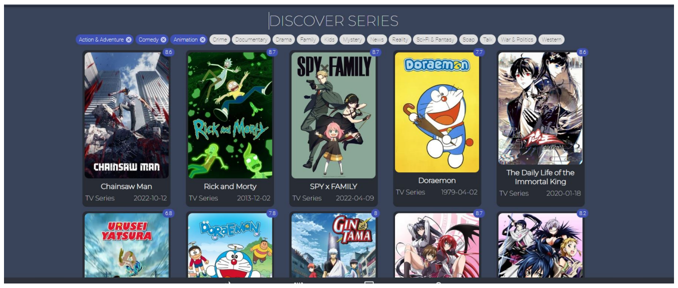

Fig. 7. Series Recommendations based on Genre

- Writing SPARQL queries with regards to multiple dataset was a difficult task and difficult to figure out.
- Hosting data on Fuseki Server was a really challenging task and took a lot of time to figure out errors with regard to that.
- We had to add data to few ontologies manually because of lack of time to implement cellfie plugin.
- The front end which is built on react, also gave us challeges in single pagination

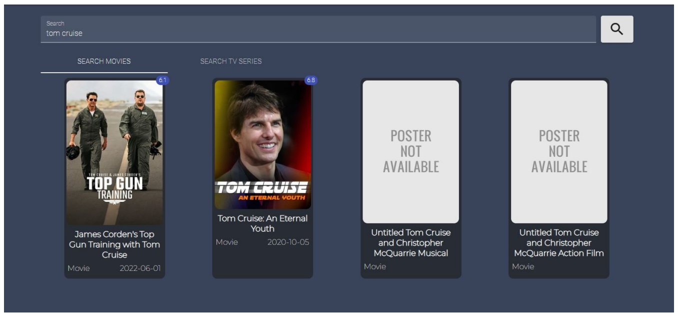
Search
tom cruise
SEARCH MOVIES
SEARCH TV SERIES
James Corden's Top Gun Training with Tom Cruise
Movie 2022-06-01
Tom Cruise: An Eternal Youth
Movie 2020-10-05
POSTER NOT AVAILABLE
Untitled Tom Cruise and Christopher McQuarrie Musical
Movie
POSTER NOT AVAILABLE
Untitled Tom Cruise and Christopher McQuarrie Action Film
Movie

Fig. 8. Recommendations based on Cast

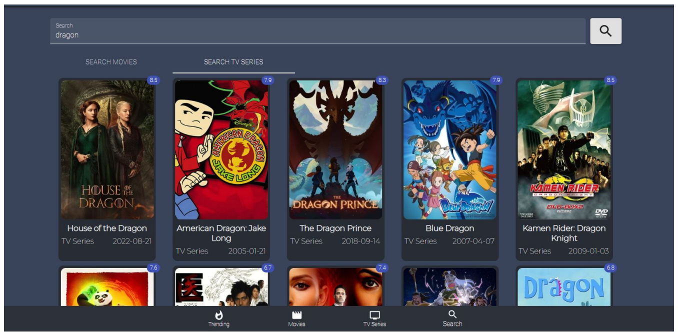

Fig. 9. Recommendations based on User's Description

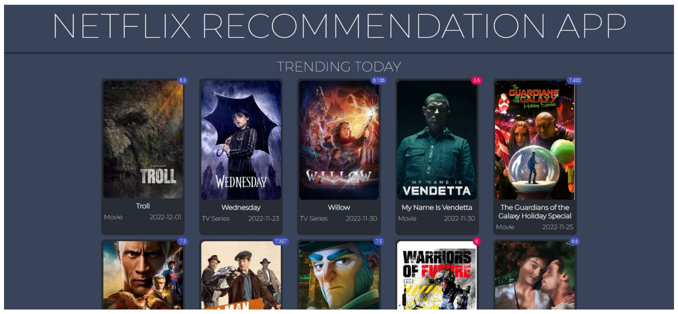

Fig. 10. Recommendations based on Trending

XI. Future Work

There can be multiple technologies that can be clubbed with the presently used tech stack. The Movies and TV Shows ontologies can be made separately in future to allow more and enhanced functionality. The recommendation and querying can be made more effective and precise with the separate ontology. Different methodologies can be used to come up with more advanced and accurate results which could provide accurate recommendations. There can be different types of recommendation algorithms apart from the proposed recommendation algorithms. We can recommend the users movies and TV shows based on a personalized interest form as well as based on their history records. Another technologies can be merged with the semantic concept to increase the user experience and interaction.

XII. Roles and Responsibilities

All of the team members have been equally contributing towards the Project to implement it successfully. Two of the team members Sparsh and Shilpi have been working on towards creating the Ontology design and carrying out literature survey for data mining models that can be used for feature prediction and how to set it up using jena fueski. Nimil and Piyush have completed the data preprocessing part using data mining steps and formatted the dataset, so it can stored in the database. They have suggested a high level system design approach that can be implemented. Kunj worked on the implementation part and carried out literature survey regarding the backend API calls using Springboot and integrate with front end. For the implementation part Sparsh and Shilpi, worked on adding instances to the build ontologies and setting up the SPARQL queries with backend. Kunj worked on setting up the apache jena fuseki and integrated them with our SPARQL queires. Nimil and Piyush worked on building the react app and integrate it with the existing backed. Lastly, they deployed the app including the mobile devices.

XIII. Conclusion

The purpose of this research paper was to determine how to choose from the vast amount of data available online the information that is necessary. A recommendation is, in short, information about things you might be interested in. Our focus was on providing recommendations to users for movies and TV shows since they are one of the major sources of entertainment. Our recommendations were based on two approaches. One is based on the features i.e recommendation using cast, director, country, rating, and genres as features. Second, is based on recommendations using the words in the movie/TV series descriptions as features. Movielens dataset was used to test these recommendations, and they were found to be effective.

References

[1] Victor Codina and Luigi Ceccaroni. A recommendation system for the semantic web. In *Distributed Computing and Artificial Intelligence*, pages 45–52. Springer, 2010.

[2] Chuan Shi, Chong Zhou, Xiangnan Kong, Philip S Yu, Gang Liu, and Bai Wang. Heterecom: a semantic-based recommendation system in heterogeneous networks. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1552–1555, 2012.

[3] Martin Szomszor, Ciro Cattuto, Harith Alani, Kieron O'Hara, Andrea Baldassarri, Vittorio Loreto, and Vito DP Servedio. Folksonomies, the semantic web, and movie recommendation. 2007.

[4] Kaggle. Netflix movies and tv shows.

[5] Rui Yang, Wei Hu, and Yuzhong Qu. Using semantic technology to improve recommender systems based on slope one. In *Semantic web and web science*, pages 11–23. Springer, 2013.

[6] Vandana Mohan Patil and JB Patil. A new semantic similarity measure based on ontology for movie rate prediction.

[7] Shinhyun Ahn and Chung-Kon Shi. Exploring movie recommendation system using cultural metadata. In *Transactions on Edutainment II*, pages 119–134. Springer, 2009.

[8] Grouplens. Movielens 1m dataset.

[9] Mehdi Elahi, Yashar Deldjoo, Farshad Bakhshandegan Moghaddam, Leonardo Cella, Stefano Cereda, and Paolo Cremonesi. Exploring the semantic gap for movie recommendations. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 326–330, 2017.

[10] Tommaso Di Noia, Corrado Magarelli, Andrea Maurino, Matteo Palmonari, and Anisa Rula. Using ontology-based data summarization to develop semantics-aware recommender systems. In *European Semantic Web Conference*, pages 128–144. Springer, 2018.

[11] Matteo Palmonari, Anisa Rula, Riccardo Porrini, Andrea Maurino, Blerina Spahiu, and Vincenzo Ferme. Abstat: linked data summaries with abstraction and statistics. In *European Semantic Web Conference*, pages 128–132. Springer, 2015.

[12] https://github.com/protegeproject/cellfie-plugin.