Technical Report

Group: H

Approach to Ontology Modelling

Description of Competency Questions

The competency questions were created with the goal of using all four datasets in mind. Each question references data from at least two of the original CSV datasets. The questions also correspond to a variety of different attributes in the data, such as IMDB ratings, budget, directors, and streaming platforms. Many of the questions return a title or multiple titles of movies as results. However, we also made sure to vary the results by having some focus on streaming platform information, specific dates, etc.

The competency questions were the first piece of work done for this project and hence had a large effect on the ontology and uplift. In creating these questions, we had to keep multiple things in mind. The main two things were the structure we wanted for the data, and how achievable it was to implement the uplift in order to answer these questions. For example, we decided to have an overall ‘media’ class with ‘movie’ and ‘tv show’ subclasses, despite the movie/show titles not originally being separated this way. This makes sense in relation to our competency questions and queries, while also making the linked data different to the original datasets. Our final list of questions is shown below:

1. Are there any shows/movies on both Netflix and Disney+ that were added to Netflix after 2019?
2. Compare the percentages of movies/shows on offer from the United States by Disney+ and Netflix.
3. Are there any Netflix movies with the same director as a different Disney+ movie?
4. Is the most common age rating of US Netflix shows/movies different to that of US Disney+ ones?
5. On average, are Disney+ movies or Netflix movies longer in duration?
6. What are the top 10 Disney movies with a rating of G on Disney+, based on box office revenue?
7. What Disney+ movies with a single director had a budget of less than 4 million?
8. What Netflix movies added to the website in 2021 have an IMDB rating of 8+?
9. What are the top 5 Disney movies with a duration of 100+ minutes on Rotten Tomatoes?
10. Of all the Disney+ movies with a budget of less than 2 million, which has the highest rating on IMDB?

Description of Datasets

We used four datasets for this project. Dataset 1 [1] contains information on Disney+ movies and TV shows. Dataset 2 [2] contains information on Netflix movies and TV shows. These two datasets have the same format/columns and made a lot of sense to combine together.

They provide attributes such as title, type (movie or TV show), release year, age rating, etc. The combination of these two datasets served as the basis for our project.

Dataset 3 [3] provides extra information about Disney movies. As such, it enhances the information on Disney+ titles that we have from Dataset 1. Along with the usual attributes such as title, it contains many extra attributes such as language, box office revenue, budget, distributor, etc. Dataset 4 [4] provides us with extra information on both Netflix and Disney+ movies. It also contains information on Hulu and Prime Video titles but this aspect was irrelevant / not in the scope of our project. This dataset mentioned which streaming platforms each title is on. This is useful as we may have information on a Disney+ movie from another dataset that is actually also on Netflix. It also provides extra information such as IMDB and Rotten Tomatoes ratings for the movies.

Assumptions Made

The first assumption made relates to our inverse functional property. We are assuming that each piece of media has one main origin country, rather than multiple. Furthermore, we assume that movies and TV shows can be streamed from multiple platforms, it does not have to be just Netflix or just Disney+. As such, these pieces of media could have more than one 'addedTo' date, for when they were added to both Netflix and Disney+. As such, we created addedToNetflix and addedToDisneyPlus properties instead. Our datasets also could have up to 50 actors listed in their 'cast' or 'actor' columns. As such, we made an assumption that the first five actors' names were the most prominent and counted those as the main actors of that piece of media. Our final assumption is that there are shared titles across our datasets, which is how we connect them.

References to Sources Used/Reused e.g. SIOC, FOAF for people

We re-used 3 different ontologies in this project. The ontologies we apply are as follows:

- We used http://xmlns.com/foaf/0.1/Person on the FOAF for the Person class.
- We used https://dbpedia.org/resource/Disney+ on DBpedia for the Disney+ class
- We used https://dbpedia.org/resource/Netflix on DBpedia for the Netflix class.

Data Mapping Process

Since there are 4 separate datasets, we created 4 separate mapping files for uplifting. This way the process will not get more complicated. Before directly uplifting the data from the datasets, we had to do some pre-processing. This included:

- Converting date to an appropriate format for type dateTime.
- Converting ratings from the format '1/10' to '1'.
- There was a single empty line in the Dataset 2 csv that meant queries could not execute.

The 'netflix_titles.csv' and 'disney_titles.csv' had both Movies and TV Shows. As such, we had to write an SQL query to dynamically assign the predicateObject values based on whether it was a movie or TV show. In the 'dataset_4.csv' all the movies were present from Netflix, Disney+. So according to the flag a sqlQuery was written that dynamically assigns the value to the predicateObject starting this media is present on the platform.

Uplifting the data types was also taken care of, eg. duration was uplifted as type ‘integer’, date was uplifted as type ‘dateTime’.

The dataset had only one column ‘cast’ which had a range of actors from 0 to 50. In order to handle this we created 5 extra columns in the dataset. Since 50 actors is a big number, we only uplifted 5 actors separately, but we are also uplifting the entire ‘cast’ column so that data is not lost.

Explanation of Use of Inverse, Symmetric and Transitive properties

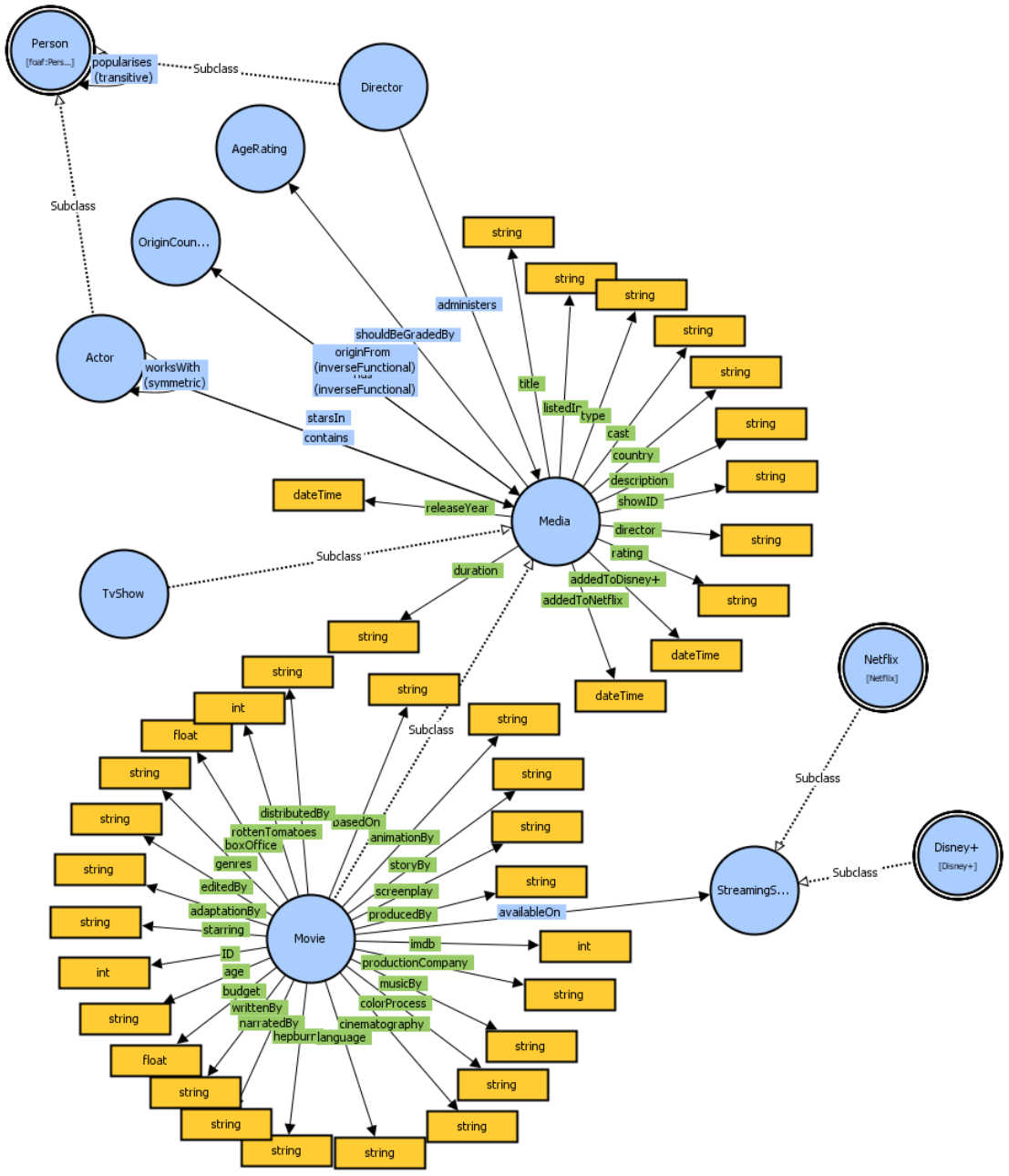
We demonstrated the Inverse Functional approach with the "has" property and the "originFrom" property that we created in the Object Properties section. The "has" property takes the class "OriginCountry" as the domain and "Media" as the range. The "originFrom" property takes the "Media" class as the domain and the "originCountry" property as the range. Another inverse property is that of the “Media” class’ “title” as each media, whether it be a TV show or movie, will only have one title.

We showed a Transitive property with the "popularises" object property that we created between the Person classes. The Person class includes the Actor and Director classes as subclasses. The logic is that people can make other people more popular, which in turn can make other people they are associated with more popular. This may be via actors working together, directors working on the same movie as an actor, etc.

We showed a Symmetric property with the "worksWith" object property that we created between the Actor classes. If Actor A works with Actor B, then Actor B naturally has worked with Actor A.

Overview of Ontology

Figure 1 - Diagram of ontology showing the links between different entities



We have added the above ontology figure to our report to express all our relationships. We created the basis for our design process by capturing the common features of the data sets we have. We needed to successfully combine these parts into common classes. One of the approaches we have implemented for this is that we created the StreamingService class for our Netflix and Disney+ datasets.

We have completed the ontology design characteristics specified in the project description with the right combinations of features found in a wide range of datasets. In order to make

our ontology design more meaningful, we also paid attention to data types and the use of labels.

We can represent some of the approaches we have taken as follows:

- We implemented the transitive function on the "Person" class with the "popularises" object property.
- We implemented the symmetric approach with the "worksWith" object property on the "Actor" class.
- We applied the inverse function to our model with the "originFrom" and "has" object properties. We used the "Media" and "OriginCountry" classes for this.
- We reused our "Person", "Disney+", "Netflix" classes with appropriate resources.

Overview of Design

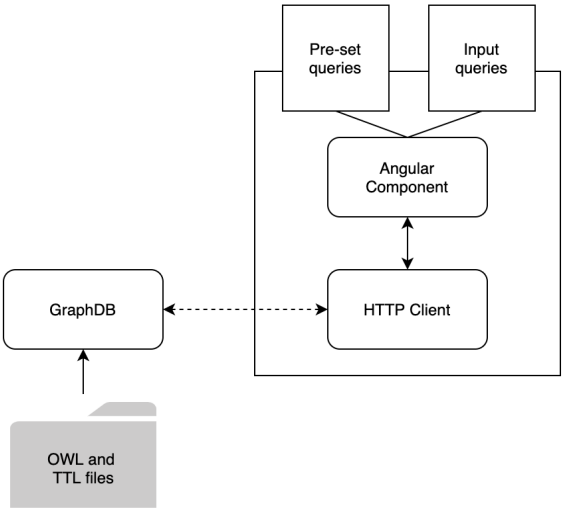
Description of Application Query Interface

The query interface for this project is set up using the Angular TypeScript framework [5] on the front end and the RDF store GraphDB [6] on the back end. The application is hosted on localhost, and it connects to the RDF store using HTTP, which is also hosted locally.

The interface is a one-page application that displays a list of pre-determined queries that the user can pick from a drop-down list to run against our triplestore. It also includes an input box that allows the user to write their own queries to send to the triplestore. Prefixes used in the queries are displayed above the input box, although users can add additional prefixes if they wish when inputting their own queries.

Once the user selects or inputs a query, the query text is sent to GraphDB, where it is parsed and used to retrieve the resulting triples. These triples are then sent back to the interface in JSON format, where they are parsed and then displayed in a table. If no triples are returned, then a message is displayed. The architecture of this system is shown in Figure 2

Figure 2 - Architecture of the different components used in the interface, inspired by [7]



This interface is a slightly modified version of the interface originally mentioned during our progress report presentation. Previously, rdfstore-js [8] was used on the backend to upload and parse our turtle files directly in our application. However, we found that the package did not perform well for the more complex queries, and it threw many parsing errors for data type compatibility. We were able to progress much more quickly with GraphDB so we decided to switch. An example of running the first query in the interface is shown in Figure 3.

Figure 3 - Example of the interface running query 1

Query Selector

1) Are there any shows/movies on both Netflix and Disney+ that were added to Netflix after 2019?

```
SELECT DISTINCT ?title WHERE {
  ?media rdf:type ?type;
    entertainment:title ?title;
    entertainment:addedToNetflix ?date;
    entertainment:provider dbpr:Netflix;
    entertainment:provider dbpr:DisneyPlus;
    FILTER(?date > "2019-01-01T00:00:00Z"^^xsd:dateTime)
    FILTER((?type = entertainment:Movie) || (?type = entertainment:TVShow))
}
```

Run

2) Compare the percentages of movies/shows on offer from the United States by Disney+ and Netflix.

3) Are there any Netflix movies with the same director as a different Disney+ movie?

4) Is the most common rating of US Netflix shows/movies different to that of US Disney+ ones? (e.g. MA vs PG)

5) On average, are Disney+ movies or Netflix movies longer in duration?

6) What are the top 10 Disney movies with a rating of G on Disney+, based on box office revenue?

7) What Disney+ movies with a single director had a budget of less than 4 million?

8) What Netflix movies added to the website in 2021 have an IMDB rating of 8+?

9) What are the top 5 Disney movies with a duration of 100+ minutes on Rotten Tomatoes?

10) Of all the Disney+ movies with a budget of less than 2 million, which has the highest rating on IMDB?

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX entertainment: <http://www.example.org/entertainment/>
PREFIX dbpr: <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
```

Submit

Responses:

Clear table

title
Genius
Christopher Robin
Muppets Most Wanted
The Princess and the Frog
Solo: A Star Wars Story
John Carter
For the Birds
The Nutcracker and the Four Realms
The Muppets

Description of Queries

The example queries we developed to test our system are derived from our competency questions. These queries require the combination of multiple datasources, therefore testing the uplifting process to ensure that all the data is correctly connected. We created draft queries and updated them once the uplifted data and ontology were at a more final state.

The ten queries used to test our datasets are constructed from the questions outlined in the competency questions section above. competency questions outlined earlier in this document. We used four different prefixes for constructing our queries. Three are external: *xsd, dbpr, and rdf*, and one is our dataset's base IRI: *entertainment*. These prefixes are added at the beginning of every constructed query.

A range of SPARQL functionality is used for creating queries to answer our competency questions. To filter out responses, we use the FILTER() clause with different variable types to exclude movies or tv shows by date, rating, entertainment type, budget, and duration. We also use the GROUP BY statement to combine responses by a certain variable. The ORDER BY statement is used to sort the responses in either ascending or descending order. Aggregate functions, such as COUNT(), SUM(), and AVG() are used in our queries to calculate data based on our datasets, such as average duration or percentage of media from the United States. The UNION statement is used to combine the results of two sub-queries,

such as finding top results for both Netflix and Disney Plus. Finally, the LIMIT statement is used to reduce the number of responses returned. This is helpful for questions that only want a specific number of results, such as finding the top five movies under a given context.

Discussion of Challenges Faced While Ontology Modelling or Creating Queries and Mappings

One of the challenges we had in constructing our queries was figuring out how to query different data types. For example, movie duration was both given as a decimal value (120.0) and a string value ("120 min") for certain entries in our datasets, so using mathematical clauses such as AVG() would give us an error. This led us to go back and do more preprocessing of our data to make sure all data types were compatible.

Another challenge we faced had to do with uplifting the netflix movie dataset. This turtle file threw an error when being uploaded to GraphDB, and we had no way of knowing which line was causing the error. Therefore we had to do a manual search through the uplifted data to see which movie, hidden in 211,560 lines of code, was causing the problem. We found out that the issue was with certain symbols included in a specific movie's title IRI.

We also had to make sure that null values were properly represented in the CSV files as, if they were not, the columns assigned for the uplifted values would not be correct. Preprocessing in general was a challenge as we did not know from the beginning what exactly we would have to do. Instead, it was a process of figuring out what was going wrong and why. For example, queries relating to dates would not work which ended up being due to the dates in the CSV files not being formatted correctly.

Uplifting the data and assigning it a class of 'Movie' or 'TV Show' was also a challenge as originally we were not sure how to do a form of an 'if' statement using R2RML. Other than the main documentation page for R2RML, there is not a lot of information online about this type of use case. We ended up being able to use an SQL query to handle the two cases separately. Similarly, we wanted to split the 'cast' column up into separate actors. However, we could not find a way to split up a single column like that using R2RML. As such, we had to add an extra preprocessing step for the CSV files. This same problem was coming while we were using Dataset 4 [4] in which we had to determine whether the Movie was streaming on Netflix or Disney+. This was also achieved using SQL query.

In terms of ontology modelling, it was a challenge to decide on which properties would be transitive, inverse, and symmetric. Our ontology is generally quite simple and there are not many complex connections between classes. As such, it was particularly difficult to find a transitive property. We ended up having to create a property that did not originally exist in the CSV to fulfill this requirement.

How We Organised Our Project

We organised this project in a couple different ways. A shared Google Doc was used to document meeting notes, progress reports, and any other information that was useful to the team members. In addition, we had both online and offline meetings every few days to discuss progress on our individual tasks, as well as offer advice and help to those working on other tasks. We primarily split the the different project tasks among us in the following way:

Sophie: Data Selection / Application scope and Competency questions / Query design
Tolga: OWL Ontology design
Bhushan: Uplift R2RML mapping
Madeleine: UI and query design

However, since all of the tasks require the work from other tasks, we had to work together to make sure everything was working correctly. We also organised the writing of the technical report based on the tasks that each person focused on during the project development.

Conclusions

This project iterates through the processes of data processing, ontology modelling, RDF uplifting, and querying. We faced many challenges during this project, but in the end we have a working ontology, along with uplifted data and an interface to test our queries.

We strongly represented our data and approaches in the Ontology Model. We would have liked to focus more on the annotations part during our design, but we had to present a design that was as accurate as possible, which underpins and provides the basis for all our approaches in a given time frame.

For the query interface, our users are able to test out each of our queries and directly see the results. The interface also includes an input box, so users can test even their own queries against our model. Although this interface works with the ontology and queries to date, its performance can be weakened if the data and models were to evolve. Overall, our interface provides the ability to test all other parts of our project and see how they work together.

Overall, we have created an RDF dataset, with an accompanying ontology, that is able to be queried via an interface. We have successfully represented the data in the way that we planned. From competency questions, to queries and ontology, to the uplift, all of the parts work together to achieve this.

References

[1] Bansal, S. (2021, October 2). *Disney+ Movies and TV Shows* (Version 1) [Movies and TV Shows listings on Disney+]. Kaggle. https://www.kaggle.com/shivamb/disney-movies-and-tv-shows

[2] Bansal, S. (2021, September 27). *Netflix Movies and TV Shows* (Version 5) [Listings of movies and tv shows on Netflix]. Kaggle. https://www.kaggle.com/shivamb/netflix-shows

[3] Kumari, S. (2021, September 18). *Real World Movie Dataset* (Version 1) [Films produced by and released under Walt Disney Pictures]. Kaggle. https://www.kaggle.com/sonukumari47/disney-movie-dataset

[4] Bhatia, R. (2021, August 2). Movies on Netflix, Prime Video, Hulu and Disney+ (Version 3) [A collection of movies found on these streaming platforms]. Kaggle. https://www.kaggle.com/ruchi798/movies-on-netflix-prime-video-hulu-and-disney

[5] Jain, N., Bhansali, A., & Mehta, D. (2014). *AngularJS: A modern MVC framework in JavaScript*. Journal of Global Research in Computer Science, 5(12), 17–23. https://angular.io/

[6] Ontotext. (2021). *GraphDB* (9.10) [RDF triplestore]. Ontotext AD. https://graphdb.ontotext.com/documentation/standard/index.html

[7] Dechesne, E. (2021, May 19). *Using OntoText and GraphDB to power an Angular app.* Medium. https://medium.com/@elvin.dechesne/using-ontotext-and-graphdb-to-power-an-angular-app-part-2-6fa9d39629d6

[8] Garrote, A. (2015). *rdfstore-js* (0.9.17) [JS RDF store with SPARQL support]. MIT License. https://github.com/antoniogarrote/rdfstore-js

[9] W3Schools. (2021). *W3Schools Free Online Web Tutorials*. https://www.w3schools.com/

[10] Musen, M.A. *The Protégé project: A look back and a look forward*. AI Matters. Association of Computing Machinery Specific Interest Group in Artificial Intelligence, 1(4), June 2015. DOI: 10.1145/2557001.25757003. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4883684/

[11] Garijo, D. (2017). WIDOCO: A Wizard for Documenting Ontologies. *Lecture Notes in Computer Science*, 94–102. https://doi.org/10.1007/978-3-319-68204-4_9

[12] Arslan, T., Borole, B., Comtois, M., & Crowley, S. (2021). *OntologyProject* [Code for project]. GitHub. https://github.com/maddiecomtois/OntologyProject