Star Wars

Final report Knowledge and Data 2023 - Group 43

Amund Strøm - 2822237
Nobbie Trouw - 2649116
Rico van Leeuwen - 2685893
Selma Kocabiyik - 2759524

1 Introduction

1.1 Goal and Motivation

Star Wars is a vast fictional universe with numerous films. Our main goal is to perform a detailed analysis of Star Wars characters' popularity. Specifically, we would like to find out if there is a correlation between characters' popularity, their relationships in the movie, and screen time. The problem that we are addressing here is about understanding whether different factors from the movie have any impact on characters' popularity. Since Star Wars is so popular and has a lot of different characters, the visualization of this analysis will provide interesting results for fans.

1.2 Stakeholders

The stakeholders are the Star Wars fans who want to dig deeper into the Star Wars universe. With the data analysis, we would like to extend fans' Star Wars knowledge. During our analysis of the data, we will provide a visualization of character relationships, their screen time, and their worldwide popularity. Additionally, we will provide some extra general information about the characters. The purpose of this analysis is to explain some things explicitly in the Star Wars universe and inform fans by visualizing different data. With the visualization of the data, fans can see the relationship between screen time and the relationship of the characters versus their popularity. Since they can view multiple characters on the same graph, they can also see differences between the characters.

2 Design and Walkthrough

Our purpose is to investigate the potential correlation among Star Wars characters' worldwide popularity, their screen time in movies, and their relationships with other characters. The analysis consists of three main steps:

1. **Screen time analysis:** Since it could be the reason that having more screen time also leads to high popularity, we found data about characters screen times.
2. **Popularity trend analysis:** After retrieving data about screen time, we collected the worldwide popularity of these characters through the years. So we could analyze the relationship between popularity and screen time.
3. **Relationship analysis:** The relationship between the characters usually gets a lot of attention from fans. Therefore, we analyze the relationship between characters. In our ontology, characters who have screen times also have FOAF (Friend Of A Friend) properties, which show their relationships with other characters. The total number of these relationships will lead us to the conclusion that a character's popularity increases because of their relationships with specific characters.

Visualization strategy

We integrate screen time and popularity for each character into the ontology to use later on in the graph visualization. We used graphical representations to visualize our findings. The first graph highlights the correlation between the popularity and screen time of the chosen movie and the chosen characters. With the second graph, the user can see the relationship between popularity and the total relationships of a character. Beside the graphs, general information about the selected character is provided. The reason that we used graphs to visualize our data is that they offer a more intuitive representation for users. Moreover, with the provided general information (abstract) about the characters, users can get extra information about the characters. In addition to the graphical representation and abstract of characters, we have provided a selection list, which enables users to choose their desired characters. By having a selection option both for movies and characters, users can access data on their favorite characters and movies. This approach ensures a personalized and user-focused experience.

3 Existing Ontologies Identified

The primary ontology for our project is centered around the Star Wars universe. This ontology covers a broad list of characters from the Star Wars movies, along with general details about these characters, such as their age, name, and race. Crucially, it also includes the relationships that each character has with other characters and their connection with specific groups. This was essential information for our project since it directly correlates to our research question, and therefore was the driving factor for reusing this specific ontology.

Our second choice of ontology is FOAF. Given the dense web of relationships among the characters, we require an established framework that describes a person's relation to other people. FOAF provides us with all the essential tools to effectively represent the relationships we are working with, making it the ideal choice for our project. Although FOAF doesn't offer extensive formal semantics, meaning it's not very strong for inferencing new facts, it does help people establish a common vocabulary for discussing things. It is highly recommended to use such similar vocabularies, which was the reason why we choose FOAF as the second ontology.

4 External Datasets

4.1 External source 1

The first external datasource we use is the screen times of each character in the first nine Star Wars movies. The data comes from imbd.com and shows eleven lists of characters with their respective screen time in minutes rounded to quarter-minutes. Each Star Wars movie has its own list. We merged the lists about the nine main movies into one table, leaving us with a table that shows for each character in which of the nine main movies they appeared

and for how many minutes. With this data, we can compare the characters screen time in each movie and their popularity.

4.2 External source 2

The second external datasource we use is a csv file with a number of Star Wars characters and their search interest worldwide on Google since 2004. The data was collected in 2018 by using Google trends which stores the Google searches from 2004 onwards. This means that the data was collected before the last movie was released and that the Google searches in the first few years after the first and second movies aren't present. Using this datasource to give each character a popularity rating is not ideal, but since we don't have the resources or time to conduct a survey of Star Wars fans about their favorite characters, this data will at least give quite a good impression of how popular each character is. Our application and ontology could always be improved by ourselves or others by inserting better data into the ontology.

4.3 External SPARQL endpoint

As our external SPARQL endpoint, we use DBpedia which is a collaborative project that extracts structured information from Wikipedia and makes it available as linked data on the web. We use DBpedia to get general information about each character to make the ontology more dense.

5 Design of Ontology

The ontology's domain and scope revolve around the characters within the Star Wars movies and their connections to one another and various associations. The ontology is designed to align with the application's goal, which is to visualize whether there's a link between a character's relationships, their time on screen, and their popularity. To achieve this a character is connected with object properties which describe the relations it has to others, and data properties that contain the different screen times. The characters are then filtered in classes by which movies they appear in, to make it accessible to compare characters from a specific movie.

5.1 Methodology

The ontology was created by using a top-down approach. We began at the top layers of our ontology, “Actor” which encompasses everything that plays a role in the movies and “Feature” which encompasses everything an Actor can be. We worked our way down from “Actor” to the individual species such as “Human” and organizations like “Academy” and from “Feature” to the specific roles of each character such as “Jedi” or “Adult”. Many of these classes were already in the Star Wars ontology that we reused, this ontology also contained many object-properties that showed relationships like “raisedBy” that proved useful for analyzing the relationships between the characters. Other classes from the reused ontology that were not useful for us such as all classes for planets were removed. By in the end

connecting object properties that showed relationships between characters from the Star Wars ontology with object properties from FOAF we made the ontology more reusable for other people as now it is much clearer that these properties show relationships between characters as this is what FOAF is widely known for.

5.2 Conceptualisation

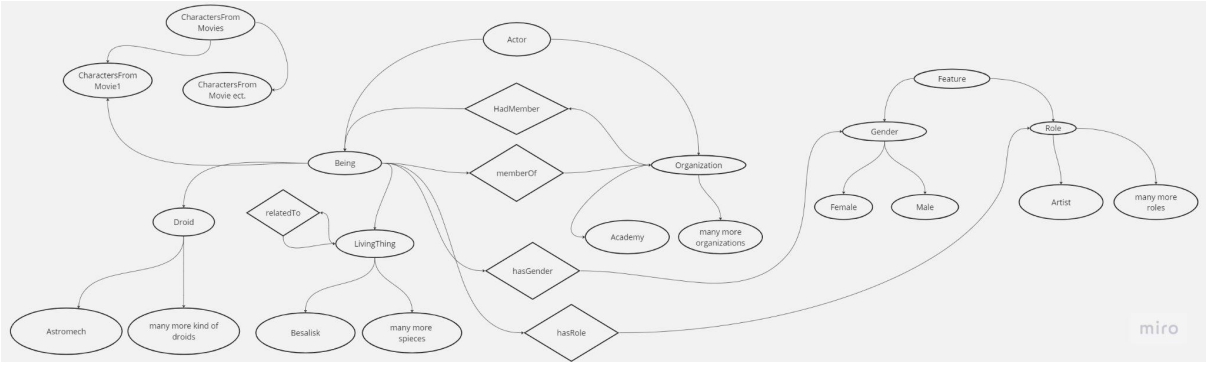


Fig. 1 - Ontology conceptualisation.

The two main classes of the ontology are “Actor” and “Feature”.
Feature is divided into “Gender” and “Role”. “Role” has subclasses that represent any role a character can have such as “Jedi” or “Pet”. These classes are mainly used for showing relations between characters using properties, for example two characters of type “Jedi” have the “alliesWith” relation.
Actor is divided into “Being” and “Organization”. “Organization” has a subclass for each kind of organization like a “Guild”. This is used in combination with the “memberOf” property which is SameAs FOAF:Member. For each “memberOf” relation a character has they have one more relationship. “Being” has 2 subclasses “Droid” and “LivingThing”. These classes have subclasses for the different types of droids and different races respectively. The characters can now fit in the classes. TX-20 a “TacticalDroid” or Jabba a “Hut”.
“CharactersFromMovies’’ is a class that has as instances all the characters from our imported data. We made this class because the ontology that we reused has a lot more characters than are present in our data. These characters are either only on screen for less than ten seconds or are from other places of the Star Wars universe like the books or series. The “CharactersFromMovies’’ is useful because we only need the characters from the movies in our application. “CharactersFromMovies” has a subclass for each of the movies so we can make distinctions between the movies too.

5.2.1 Classes

This table does not include every class since there are way too many, but covers the most relevant and with the highest level in the hierarchy.

Class name	Meta-Property
Actor	
Feature	
Gender	Subclass of Feature

Female	Subclass of Gender Equivalent to gender some Actor Disjoint with Male
Role	Subclass of Feature
Adult	Subclass of Role Disjoint with Baby, Child
Baby	Subclass of Role Disjoint with Adult, Child
Jedi	Subclass of Religious Role Equivalent to memberOf value ‘Jedi Order’
Being	Subclass of Actor
Droid	Subclass of Being
Living thing	Subclass of Being
Organisation	Subclass of Actor
CharactersFromMovies	
CharactersFromMovie1…9	Subclass of CharactersFromMovies Equivalent to hasScreenTimeMovie1…9 some xsd:decimal

Table 1.

5.2.2 Properties

This list does not include every property, but covers the most relevant.

Properties	Meta-properties
connectedTo	Symmetric
gender	Domain: Being Range: Gender
hadMember	inverse of: memberOf
hadRole	Domain: Being Range: Role
inConflictWith	Symmetric Domain: Actor

	Range: Actor
knew	Symmetric same as foaf:knows Domain: Actor Range: Actor
locatedIn	Transitive Inverse of: locationOf
memberOf	Transitive foaf:member inverse of: hadMember
relatedTo	Transitive Symmetric Domain: Living thing Range: Living thing

Table 2.

5.3 Formalization / Implementation

Of the RFDS vocabulary usage in our ontology, the most notable usage is in defining subClass and subProperty relations between classes and objects. Most of the subClass relationships are visualized in the conceptualization figure above (Fig. 1) and the tables below it. There are basically three main classes(Actor, CharactersFromMovies and Feature) of which all other classes are either a direct or indirect subClass.
Usages of owl vocabulary properties are almost fully listed in the table above(Table 2). Apart from owl properties, owl restrictions are primarily used in the ‘CharactersFromMovies' classes in order to infer the instances that should belong to each of those classes.

6 Integrating Data

The first dataset used in our project came from a webpage and thus we first had to retrieve this data and prepare it in a way such that it could be imported into our ontology. Since it was not that much data, we decided to just enter all the data into an excel sheet mostly by hand. After this, we had two datasets in the form of an excel spreadsheet. Both of these datasets concerned linking characters to certain objects/properties. However, the character names in the spreadsheet were not always written exactly the same way as the characters already in our ontology. To fix this, we retrieved all the names using a SPARQL query and did a look up for each name in the datasets. Using this, we had a quick overview of all the names in the dataset which were not directly corresponding to a name in our ontology and were able to quickly resolve these by hand. With the data all sorted out, each dataset was loaded in Onotext refine and mappings were made. After the mappings were created, the mapping was then converted to SPARQL and opened in GraphDB. After running the generated Construct query, the results were checked and then modified to Insert the data into our

ontology. With the screen time data imported the characters should be able to be inferred in the ‘CharactersFromMovieX’ classes.

7 Meaningful Inferences

Inferencing plays a vital role in our ontology as it's responsible for determining new class memberships and character relationships. Initially, each character comes with a limited set of explicit object/data properties and class connections. However, by allowing inferencing it populates each character's profile by introducing additional properties and expanding their class memberships. This allows us to see if there is a connection between a character’s relations, screen time and popularity.

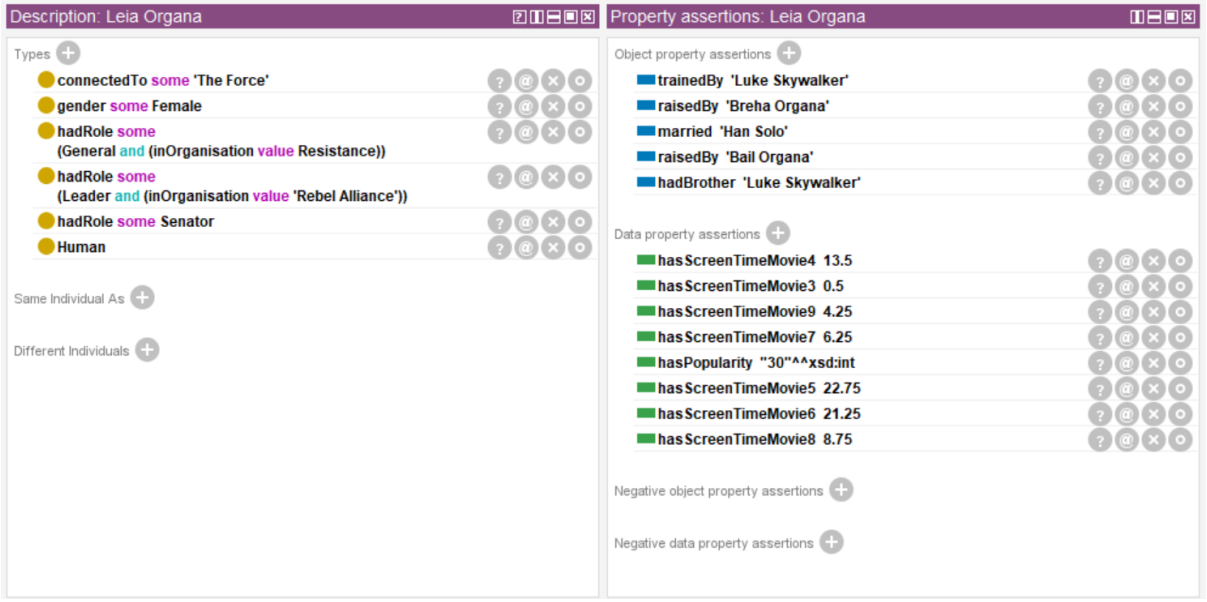


Fig. 2 - Individual from Protégé with inferencing turned off.

In Figure 2, we can observe the individual "Leia Organa" along with her associated properties and memberships. The current number of relations to other characters and groups are minimal and does not represent the true amount. This would produce poor outcomes within our application. However, it is crucial to note that we do possess the vital external data, which includes her screen time in each movie in which she made an appearance and her popularity score.

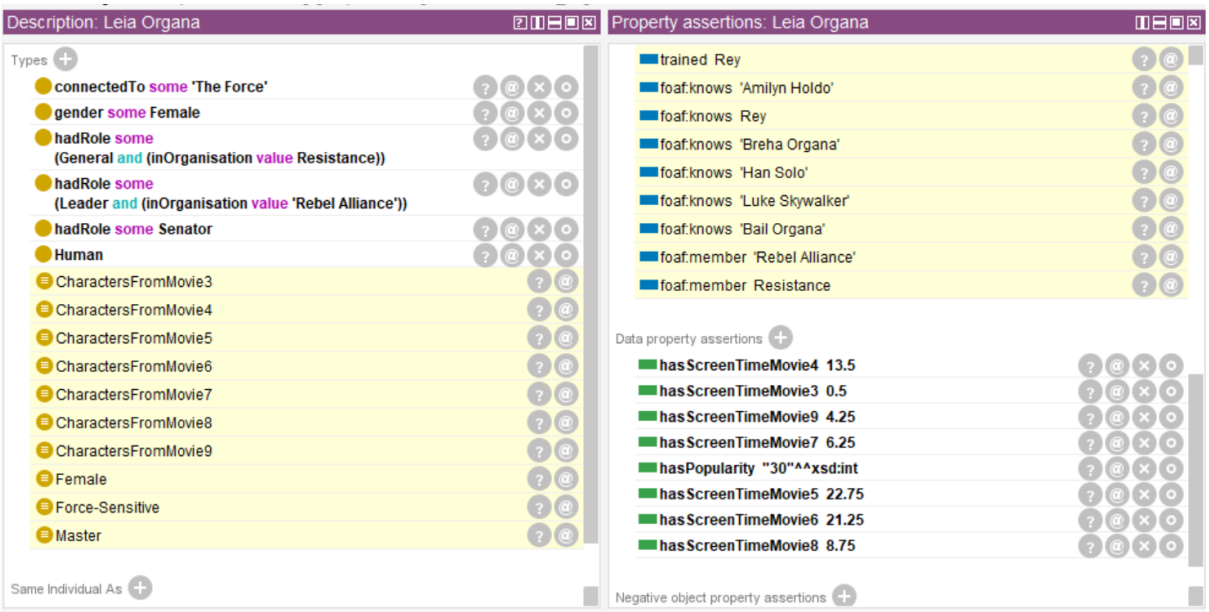


Fig. 3 - Individual from Protégé with inferencing turned on.

In Figure 3, it becomes evident that enabling inferencing results in a substantial growth in the number of relationships, which more accurately represents the true number of relations and provides us with the opportunity to explore potential correlations between these relationships, screen time, and popularity. Additionally, it's worth noting the appearance of new class memberships, that tell the movies in which the character appears in. This allows a more structured and systematic approach to extracting characters associated with specific movies.

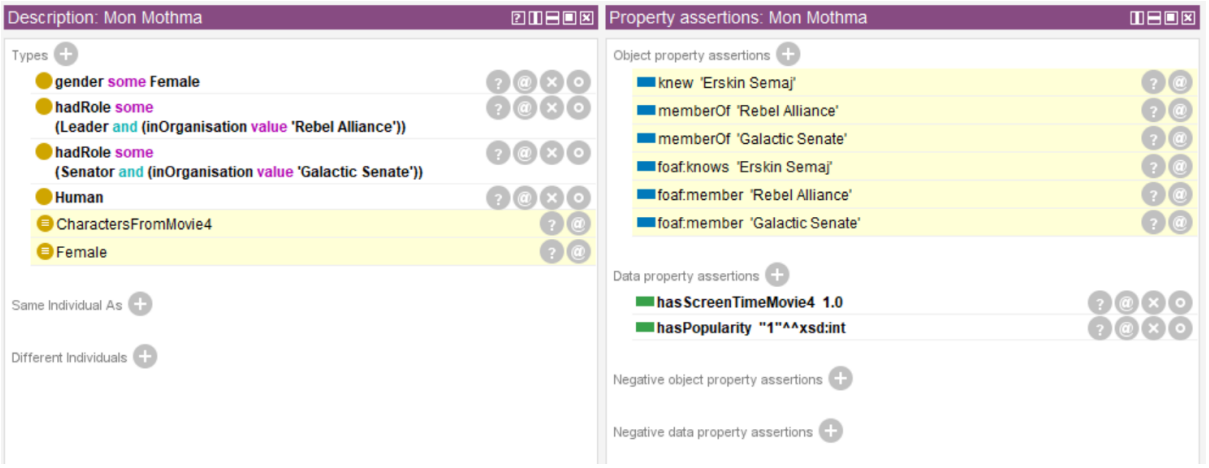


Fig. 4 -Individual from Protégé with less relations.

Figure 4 illustrates a direct relationship between a character's screen time and popularity, revealing that characters with lower screen time and popularity scores tend to have fewer connections and class memberships.

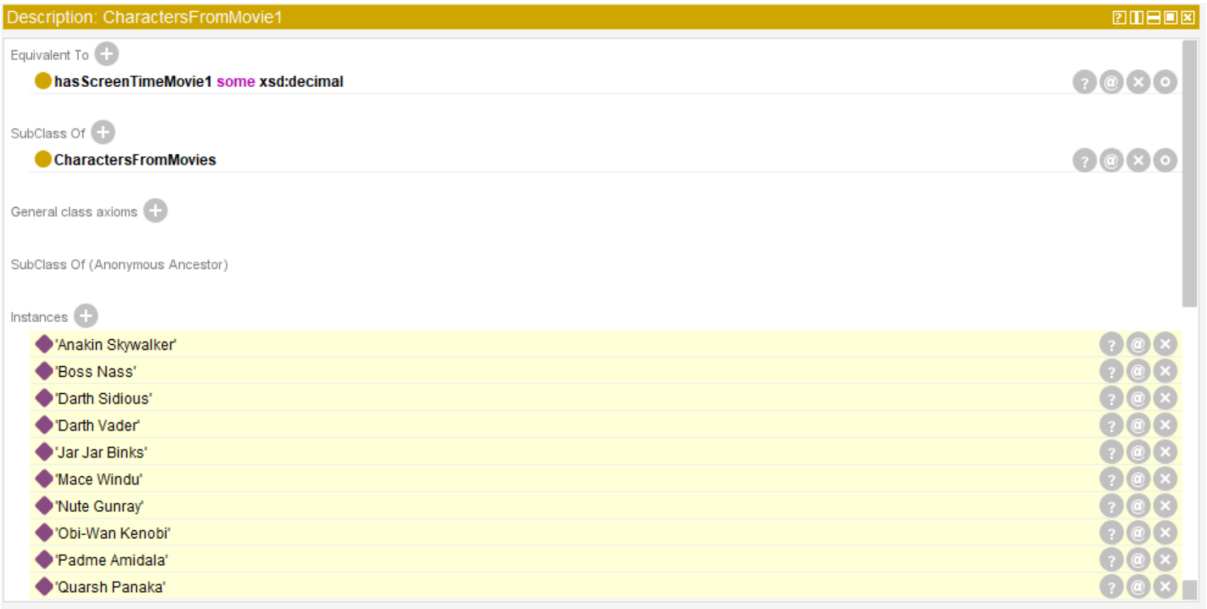


Fig. 5 - Class in Protégé.

In Figure 5, we have a list of all characters featured in the initial movie. There is an equivalent class for each movie, housing all the characters that made an appearance in that specific film.

8 Relevant SPARQL queries

Within our project, we employ SPARQL queries to present the relationships, screen time, and popularity metrics for each character across all movies. We visualize this data in a Pandas dataframe for a more user-friendly and readable format. The initial step in this process is to gather a list of characters from each movie.

```
1  PREFIX sw: <http://www.kandd.org/group43/ontologies/star-wars#>
2  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3
4  SELECT DISTINCT ?char
5  WHERE {
6      ?char rdf:type sw:CharactersFromMovie1
7      FILTER(REGEX(STR(?char), "^http://www.kandd.org/group43/ontologies/star-wars#"))
8  }
```

Fig. 6 - Unique characters SPARQL query.

In the figure above, we extract the unique characters from the first Star Wars movie. The "filter" keyword is used to exclude any subjects that do not fulfill the specified prefix. This is necessary because, during the integration of external data concerning screen time for each character, certain characters with a dash in their names were assigned a different prefix. While addressing this in the ontology was straightforward, involving the use of the "same as”

property, the SPARQL query requires to filter out these individuals.

```
PREFIX sw: <http://www.kandd.org/group43/ontologies/star-wars#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT DISTINCT ?char
WHERE {
    ?char sw:hasScreenTimeMovie1 ?x
    FILTER(REGEX(STR(?char), "^http://www.kandd.org/group43/ontologies/star-wars#"))
}
```

Fig. 7 - Unique characters SPARQL query.

When retrieving the characters from a particular movie, our preferred approach would be to utilize the dedicated classes that house this information, such as "CharactersFromMovie1." However, an issue arises during ontology inference within GraphDB, as it fails to infer instances for these classes. Therefore, we resort to a less elegant workaround, which involves relying on the predicate that stores the screen time information for each character in every movie, as illustrated in Figure 7.

```
SELECT ?char (COUNT(?movie) AS ?movieCount)
WHERE {
    ?char ?movie ?x .
    FILTER(REGEX(STR(?char), "^http://www.kandd.org/group43/ontologies/star-wars#"))
}
GROUP BY ?char
ORDER BY ASC(?movieCount)

# Define movie
VALUES ?movie {
    sw:hasScreenTimeMovie1
    sw:hasScreenTimeMovie2
    sw:hasScreenTimeMovie3
    sw:hasScreenTimeMovie4
    sw:hasScreenTimeMovie5
    sw:hasScreenTimeMovie6
    sw:hasScreenTimeMovie7
    sw:hasScreenTimeMovie8
    sw:hasScreenTimeMovie9
}
```

Fig. 8 - Total screen time SPARQL query.

The SPARQL query shown in figure 8 calculates the total screen time for all characters and arranges them in ascending order. By employing the "value" keyword, we assign more values to the "movie" variable that it must query. In this manner, we obtain the characters associated with each movie.

```
PREFIX sw: <http://www.kandd.org/group43/ontologies/star-wars#>

SELECT DISTINCT
    ?char
    (COUNT(DISTINCT ?memberOfEntity) as ?memberOfCount)
    (COUNT(DISTINCT ?marriedEntity) as ?marriedToCount)
    (COUNT(DISTINCT ?knewEntity) as ?knowCount)
    (SAMPLE(?popularity) as ?sampledPopularity)
WHERE {
    ?char sw:hasPopularity ?popularity .
    FILTER(REGEX(STR(?char), "^http://www.kandd.org/group43/ontologies/star-wars#"))
    OPTIONAL { ?char sw:memberOf ?memberOfEntity . }
    OPTIONAL { ?char sw:married ?marriedEntity . }
    OPTIONAL { ?char sw:knew ?knewEntity . }
}
GROUP BY ?char
```

Fig. 9 - Total relations SPARQL query.

The SPARQL query seen in figure 9 provides us with the count of unique relations and the popularity scores for each character. The query will only fetch characters possessing a popularity score, resulting in the exclusion of a large portion of the character dataset. The reason for this is the fact that our integrated data contains that information for only around 40 characters. Furthermore, by using the optional keyword on the relations, the query will return even if it does not have any relations.

9 Data Science pipeline

In our application, we constructed an interactive visualization tool that provides insights into character data from the Star Wars universe. Our process integrates data from both our ontology and external sources, such as DBpedia with SPARQL queries.

SPARQL queries

At the beginning of the notebook, the turtle form of our ontology file can be viewed. Here, we used *rdflib* to convert the ttl file to turtle format. In our implementation, four different SPARQL queries were used. To retrieve relevant data, three of these queries communicate directly with our ontology; *fetch_characters_from_movie*, *fetch_data_for_character_with_popularity* and *fetch_relationship_counts*. The fourth, *fetch_general_info_from_dbpedia*, communicated with the DBpedia SPARQL endpoint in order to obtain general information about selected characters. So once we have the user selected characters from *fetch_characters_from_movie*, they then used to fetch data from the DBpedia endpoint via *fetch_general_info_from_dbpedia*, to obtain general information. With *fetch_data_for_character_with_popularity* we retrieve screen times and popularity of the given characters from the ontology through the properties (*hasPopularity* and *hasScreenTimeMovie*). On the other hand, *fetch_relationship_counts* finds the relationship that one character has through the properties *memberOf*, *married*, and *knew*. These are also FOAF properties that we count to get the total relations of one character. Additionally, it gets popularity value for characters too. Retrieved data is structured in JSON format and we store the data in a Pandas DataFrame using the following code (the same logic used for each retrieved data):

```
# from fetch_data_for_character_with_popularity function
df = pd.json_normalize(results['results']['bindings'])
df['property'] = df['property.value']
df['screenTime'] = df['screenTime.value']
df['popularity'] = df['popularity.value'].astype(int)
df = df[['property', 'screenTime', 'popularity']]
```

Graph plotting

In our application, we visualize our data with two graphs, one for screen time vs popularity and the other for total foaf relationship vs popularity. To plot the graphs, we used Pandas.

Our purpose was to make points on the graph that represent a character,as can be seen within the following code.

```
for i, row in df.iterrows():
      plot_axes.annotate(f"{row['characterName']}",
            (row['screenTime'], row['popularity']),
            textcoords="offset points", xytext=(0,-15), ha='center', fontsize=8)
```

We set the max value for the limit of the y-axis based on the highest popularity value of the selected character to ensure optimal readability.

```
max_popularity = df['popularity'].max()
if max_popularity in range(1, 50):
      plot_axes.set_yticks(range(0, 51 , 5)) # Ticks for every 5 units up to 50
      plot_axes.set_ylim(0, 51)

elif max_popularity >= 100:
      ylim_top = max_popularity + 10
      plot_axes.set_yticks(range(0, ylim_top, 5))
      plot_axes.set_ylim(0, ylim_top)

else:
      plot_axes.set_ylim(0, 100)
```

We used exactly the same logic for the *plot_relationship_count_vs_popularity* function too. In addition to these parts, other parts of the plotting functions were to be used for x-ax and y-ax arrangements and for the design of the graph. To color the points in the graph, *get_color_screentime* and *get_color_relationship* are used.

Widgets and Main function

We used two buttons to confirm the selected movie and characters. With the *widgets,* we provide a high quality user interface, and with *Checkbox*, *Button,* and *VBox,* we arrange the buttons.

The main function connected each function for visualization. We used *HTML* to convert general information from the DPpedia to normal text to make it more readable. Additionally, we used *concatenation (concat* from *Pandas)* to concentrate the data.

Used Python libraries and tools

- ipywidgets: Used for users to see movie options and have a selection list for the characters. The highly functional user interface that this library provides makes the application easy to use.
- IPython.display
- rdflib: Allowed us to work with RDF data from the ontology.
- SPARQLWrapper: We used this for sending SPARQL queries to endpoints and getting the data for our analyses.
- pandas: With Pandas, we analyzed our data and formed graphs for visualization.
- matplotlib.pyplot: With this, we created our visualizations and plots.
- re: Used to clear the input from the user.
- HTML (from IPython.display): It is used to display HTML content in our notebook and see the data clearly.

10 Justification

We were all part of choosing what the application should be about and brainstorming solutions to different problems encountered; each member also wrote approximately 1/4 of the report. The main contributions for each member were as follows:

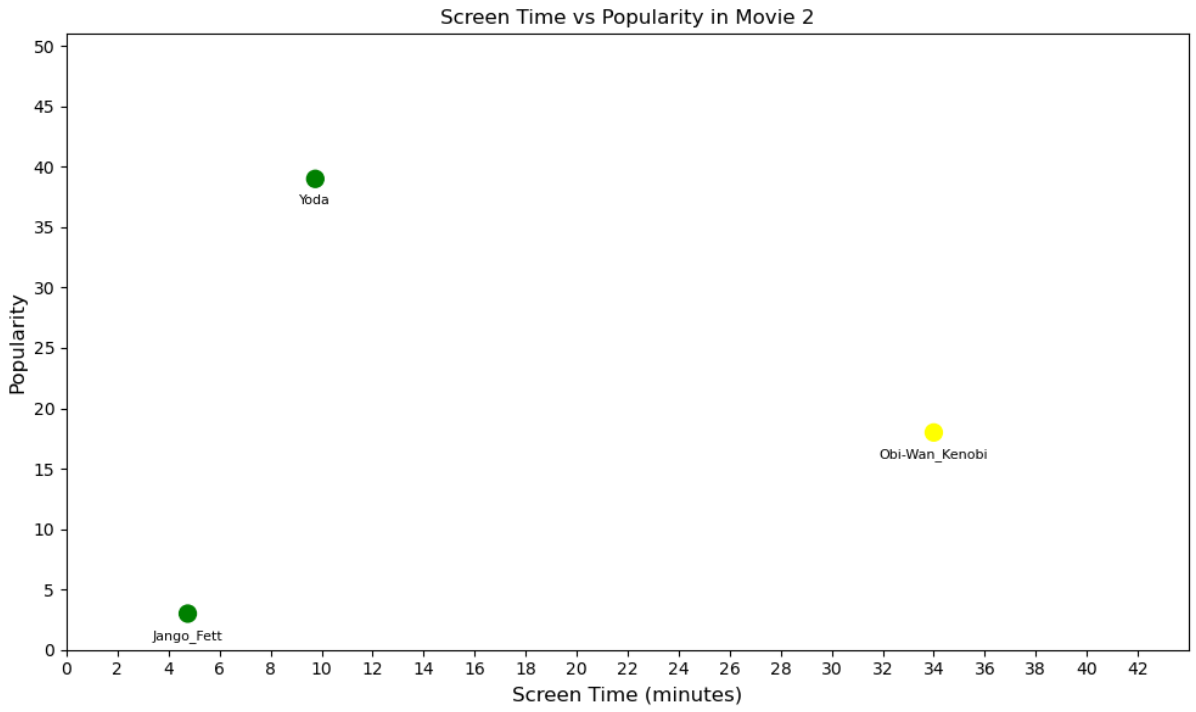
Amund Strøm - SPARQL queries.
Nobbie Trouw - Ontology conceptualization.
Rico van Leeuwen - Cleaning and integrating external data.
Selma Kocabiyik - Visualize data in Panda.

References

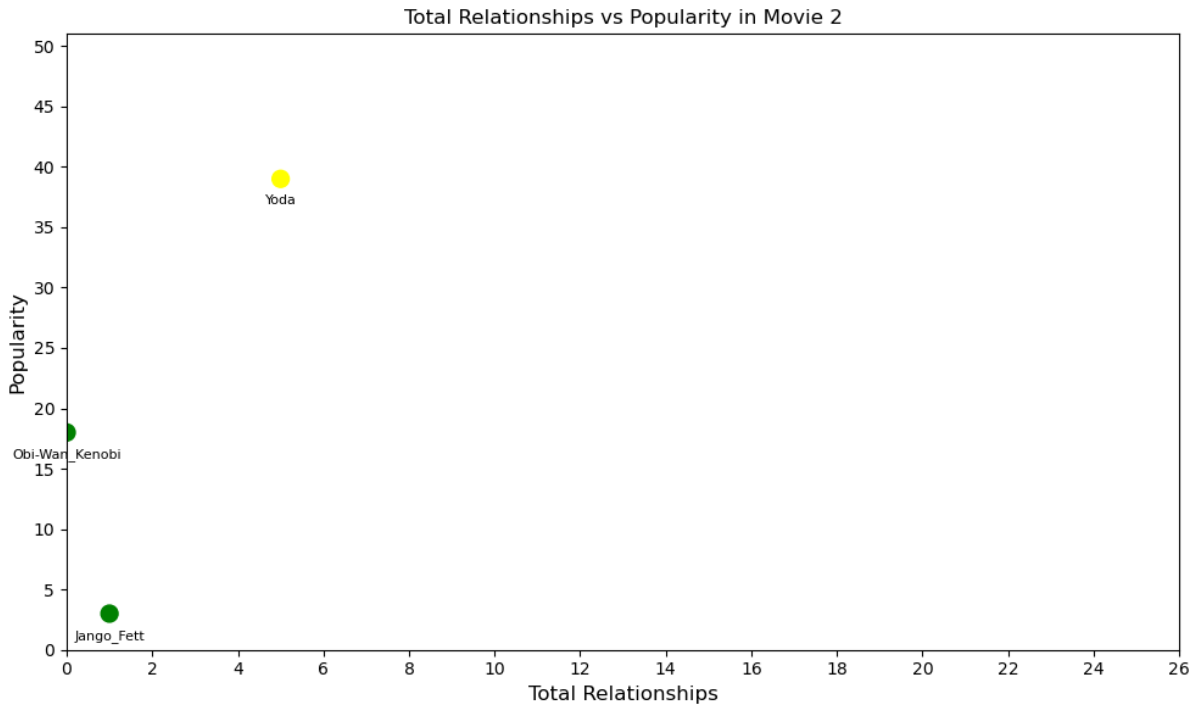
Ontologies:
(nickdrummond/star-wars-ontology: https://github.com/nickdrummond/star-wars-ontology)
(FOAF: http://xmlns.com/foaf/0.1/)

Popularity for each character:
(GoogleTrends/data:
https://github.com/GoogleTrends/data/blob/master/2018-04-05_Star_Wars_characters.csv)

Screen Time for each character:
(imdb: https://www.imdb.com/list/ls027631145/)



Screen time vs popularity graph from the application



Total relationships vs popularity graph from the application