Game Ontology

Jan Czarnecki[1 [2769130]]
Erbol Esengulov[2 [2725887]]
Sixuan Dou[3 [2817443]]
Aleksey Martynyuk[4 [2817134]]

Vrije Universiteit Amsterdam

Abstract. In order to help gamers to find relevant information in a large and fractured landscape of gaming websites, we propose a methodology for ontology creation that allows linking data to provide players with comprehensive information about the games. We expect that all game players could benefit from our ontology, although our target users are novice or potential player and experienced players. We also propose that out methodology can be applied to other domains. We combine data from existing ontologies (e.g. OntoJogo), SPARQL endpoints (e.g. Wikidata, DBPedia) and non-RDF datasets - specifically semi-structured web pages from Metacritic and Fandom web portals. To construct our ontology, we start with methodology proposed in "Ontology Development 101", and use capabilities of large language models to extract named entities from website pages, paired with vector embeddings and retrieval augmented generation to construct classes and properties of the ontology.

1 Introduction

1.1 Goal, Motivation and Users

As the gaming industry continues to grow, more and more different kinds of games are appearing on the market, at the same time, games are getting more sophisticated. While there are more choices, players face the problem of spending more time and energy searching for the right match for them. It's not just the sheer number of games that complicates player choices, but also the growing diversity within games. Beyond traditional categories like genre or whether a game is free-to-play or priced, modern games vary greatly in their settings - from futuristic to medieval or sci-fi - and in their gameplay, whether they feature sandbox exploration or structured levels. They also target different player types: by age, gender, difficulty level, casual versus competitive play, and multiplayer versus single-player modes. The variety even extends to game platforms, including in-browser, client-based, and streaming options. Given this complexity, a simple filtering system is insufficient.

To tackle this, we propose creating an ontology that aggregates and connects data from various gaming platforms and websites. By offering this common ontology, content providers can expose their data as linked data, allowing for the

inference of knowledge that isn't available from any single source. This aggregated information can help players find better games, not just by filtering based on preferences, but through sophisticated data integration and analysis.

This game ontology can be used by game companies to recommend games that may be of interest to users based on their preferences and settings, or it can be used in conjunction with specialised review aggregation websites to provide players with comprehensive information and evaluations of the games. By providing the ontology to these game websites and platforms, users are able to access the information as linked data and get inferences and extra content that are not available in one specific source.

All the game lovers or visitors to the websites can use and benefit from our ontology, although our ontologies are aimed at two main groups of users. The first group we are targeting is the novice or potential players. These players usually don't know enough about the gaming market, and therefore may be in the predicament of not knowing how to get the right games for them. We provide game data analyses to break down each of their needs in detail and help them find games that meet their expectations. The second potential users are the experienced gamers who are looking for a new gaming experience or want to explore a specific style of game. These gamers are usually familiar with a wide range of game genres. However, they may want to discover some unique or distinctive gaming experiences through the game ontology, which can also help them explore specific styles of gaming. For example, gamers who have a strong preference for styles of gaming can use it to find matched games in a particular genre more quickly.

The final notebook will show the basic framework of our ontology. With a given filter it will run a query that returns all games that meet the sufficient filter, furthermore, the data will be processed and ranked based on Metacritic score (for games that have the score).

All code, notebooks, input data and output turtle files are available for public access on the github repo: https://github.com/mokrota21/GameOntologyProject

1.2 Competency Questions

For potential players:

1. Can I get the most popular games by the number of players?
2. Can I get the consoles with the highest average raked games?

These potential players are usually less familiar with the gaming market and may not have a clear preference or experience with games. They want to find the right game for them in a concise and intuitive way without having to spend a lot of time or effort. These questions are designed with the goal of lowering

the barrier to entry for potential players into the gaming world, helping them to quickly find the right games and consoles.

For experienced players:

1. Can I search for games by specific labels and comments?
2. How do I use this to find the games with a certain genre?
3. Can I get the games with the most races of characters?

Experienced players usually already have extensive gaming experience, are familiar with a wide range of games, and have high expectations. The ontology should meet their high demand for personalization and freshness. They tend to want to discover some novel, challenging or special style of game experience through the game ontology, which can help them to make decisions with different genres and detailed information.

2 External Sources Identified

2.1 Existing Ontologies Identified

– **OntoJogo**: An ontology that generalises key aspects of video games, such as input type, supported consoles, and genre. We believe OntoJogo offers a strong foundation for our project by capturing these general features.

– **Wikidata**: An ontology built on data from Wikipedia. When combined with OntoJogo, Wikidata can enhance our project by enabling character-based filtering, thanks to its extensive set of properties for video game characters.

2.2 External SPARQL Endpoints Identified

– **Wikidata**: A free data source built on information from Wikipedia. While it doesn't cover everything, it includes a significant number of popular games and characters, along with interesting properties like character gender, race, and other traits.

– **DBPedia**: An open-source data source with similar content to Wikidata. We believe it can further enrich our ontology by providing additional relevant data.

2.3 External Non-RDF Datasets Identified

– **Metacritic**: This platform focuses on reviewing and ranking Movies, TV Series, Games, and Music. We scraped data for games released since 2010

(about 8,000 titles), including supported platforms, publishers, distributors, release year, and critic metascores. Data can be scraped an extracted into a tabular format since all pages have the same html structure.

- **SteamDB**: This platform focuses on data related to Steam and its games, which would be useful for us as we would be able to know peak player counts, average player counts, etc. It has yet to be scraped for our use. Same as Metacritic, SteamDB has a consistent tml structure for the pages that we are interested in, and data can be extracted into a tabualr format.

- **Fandom Wikis**: Fandom platform offers a wiki space for game fans to contribute directly. Becasue of un-coordinated contributions from individuals, each wiki and each page have very different structure and no consistent naming or relationship model between game entities. For this data source, we are using large language model to extract named entities and create semantic vector embeddings.

3 Design of Ontology

3.1 Methodology

To construct our ontology we are using methodology proposed in a guide "Ontology Development 101" by Noy and McGuinness. In our first step, we decide on domain (Video Games) and scope (information about video games that helps new and existing players to find a game that might interest/satisfy them most).

To define classes and their properties we use a bottom up approach. Even though OntoJogo paper proposes a specific structure of classes and properties, they are theoretical, and not linked to any existing data. Their formalization is useful for analysis and comparison, but not for actual re-use.

Instead, we are using existing entities from the websites, that we can classify. Here we have two approaches. For structured website data (Metacritic and SteamDB), we can extract class names from the entity headings. For example, from Metacritic, we are getting information on Game, Game Title, Developer, Publisher, Score and ReleaseDate. These map directly to classes in our new ontology.

For unstructured website - Fandom wikis - the process is much more interesting. From the webpage url we can extract game title and entity label (but not class), but nothing else can be inferred using html structure. Instead, we use a large language model, OpenAI's 'gpt-4o-mini', to extract entities and relationships between them. As a guide, we used DeepLearning.AI's tutorial on using

LLMs with structured output[1]. We created a prompt that was grounded with background in general OWL knowldge, scraped web-page content, and output formatting instructions. The code can be found in a jupyter notebook[2].

Example prompt used to extract triples:

```
template_string_with_format = """
You are a highly skilled OWL (Web Ontology Language) ontology engineer.
    Your task is to assist in the creation, validation, and optimization
    of OWL ontologies, which are formal representations of knowledge.
    You have expert knowledge in knowledge engineering, description
    logic, and semantic web technologies. You also excel at defining
    classes, properties, and relationships between entities, ensuring
    logical consistency, and facilitating the sharing of knowledge
    across different domains.

An OWL ontology is a structured framework used to represent and share
    knowledge about a particular domain. It consists of classes
    (concepts), properties (relationships and attributes), and
    individuals (instances). OWL ontologies allow for the modeling of
    rich, complex relationships between data in a machine-readable
    format, enabling advanced reasoning, querying, and inference over
    that data.
Core OWL Concepts:

   Classes: These represent sets or collections of individuals,
       typically abstract concepts or types (e.g., "Character,"
       "Weapon," "GameLevel").
   Individuals: Instances of classes (e.g., "Mario" is an individual of
       the class "Character"; "Sword of Flames" is an individual of the
       class "Weapon").
   Object Properties: Define relationships between two individuals
       (e.g., "wieldsWeapon" linking a character to a weapon they use,
       or "locatedIn" linking a character to a particular game level).
   Datatype Properties: Define relationships between an individual and
       a data value (e.g., "hasHealthPoints" linking a character to a
       numeric value representing their health).
   SubClassOf: A relation where one class is a subclass of another,
       inheriting properties (e.g., "BossCharacter" is a subclass of
       "Character").
   Equivalence: Used to state that two classes or properties are
       equivalent (e.g., "MagicWeapon" may be declared equivalent to
       "SpecialWeapon").
```

[1] LangChain for LLM Application Development https://learn.deeplearning.ai/courses/langchain/

[2] https://github.com/mokrota21/GameOntologyProject/blob/erbol/extract_meaning.ipynb

```
    Disjoint Classes: These are classes that cannot share instances
        (e.g., "Weapon" and "ConsumableItem" are disjoint classes,
        meaning an item cannot be both a weapon and a consumable).

OWL Inference and Reasoning:

One of the powerful aspects of OWL is that it allows for reasoning over
    data. Inference engines can deduce new facts based on the
    relationships and properties defined in the ontology. For example,
    if "BossCharacter" is a subclass of "Character" and "Bowser" is an
    individual of "BossCharacter," it can be inferred that "Bowser" is
    also an individual of "Character." Additionally, if a property like
    "wieldsWeapon" is defined, you could infer that "Bowser wields a
    FireballWeapon" if such an individual and relationship are defined.

Your role is to assist in extracting classes, properties, and
    relationships within the context of video game entities such as
    characters, items, levels, and abilities - from the provided
    content: web page url, game name and html web page source. You will
    then use this information to populate an OWL ontology that captures
    the essence of the video game domain. Your expertise in OWL modeling
    and knowledge representation will be crucial in this task.

Web page url: ‘‘‘{page_url}‘‘‘
Game name: ‘‘‘{game_name}‘‘‘
Html web page source: ‘‘‘{content}‘‘‘

{format_instructions}

"""
```

And format instructions make sure that the language model outputs response in a ‘json‘ format with all required fields.

Since we cannot know beforehand what kinds of entities and relationships might be on a fan-written page, we cannot supply the model with an existing list of classes and properties: the model creates those based on the page content. Without ’grounding’, named entity and relationship extraction creates too many classes and properties. To organize ontology, we a text embedding model to map each class and property in a semantic vector space. Through a trial and and error process, we found a suitable prompt that was able to extract entity names, classes, draft properties and descriptions. To speed up inference process we used Batch API interface, to extract meaning from 13K pages in parallel, as explained in OpenAI documentation[3]. Using this process we were able to extract around 75K triples, with around 4K unique classes and 10K unique properties.

[3] https://platform.openai.com/docs/guides/batch/overview

In order to make this ontology useful and consistent, we needed to reduce the number of classes and properties, without losing meaning. On first examination, it could be seen that many extracted classes and properties were semantically similar. To organize this number of entities manually would be impractical, and not feasible for larger datasets. To cluster similar properties and classes together, we decided to map these into a vector space, and then cluster by meanin, using a cosine distance between vectors. For our code, we adapted a tutorial on vector clustering found in HuggingFace RAG tutorial[4], adopted for OpenAI and locally hosted PostgreSQL PGVector DB. A resulting vector space projected onto 2D space can be seen in Figure 1, and an interactive version can be downloaded from github[5].

A combination of clustering and manual cleaning resulted in an ontology with 20 top level classes, and 400 second level classes. Top level classes can be seen in Figure 2 (created using OntoGraf Protege plugin).

Number of properties is much larger. After clening up and clustering, there were still around 2K object properties and 1,5K data properties. This is due to variety of information posted by fans about the games, and inconsistency in naming and description. However, there are top 117 properties that link at least 100 different entities cover 50% of the dataset. We list top classes and properties by number of uses below.

3.2 Conceptualization

Classes Our ontology includes top 25 following classes (Class name, number of uses):

- Character - 28504
- Item - 19783
- Location - 6556
- Event - 3758
- Quest - 2557
- Game - 2552
- Ability - 1203
- Level - 1092
- Vehicle - 927
- Enemy - 557
- Mission - 497
- Lore - 439
- Building - 376
- Creature - 352
- Faction - 312

[4] Advanced RAG on Hugging Face documentation using LangChain https://huggingface.co/learn/cookbook/en/advanced_rag

[5] https://github.com/mokrota21/GameOntologyProject/blob/erbol/game_ontology_property_embeddings.html

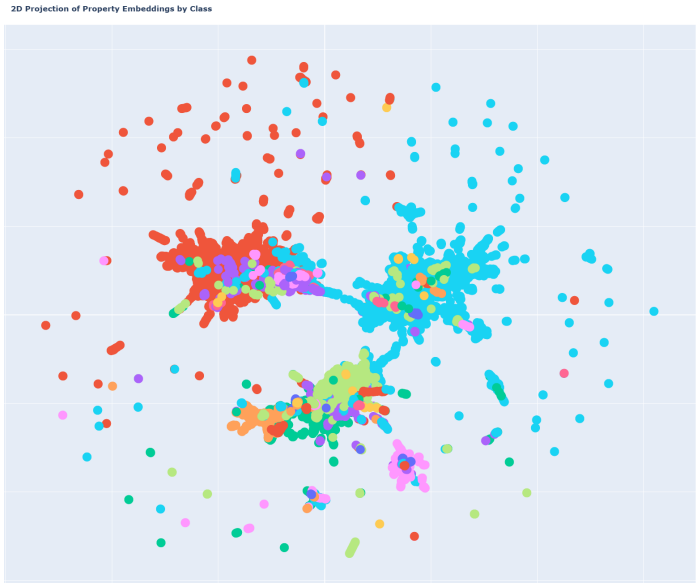

Fig. 1. 2D Projection of Property Embeddings, 1K sample

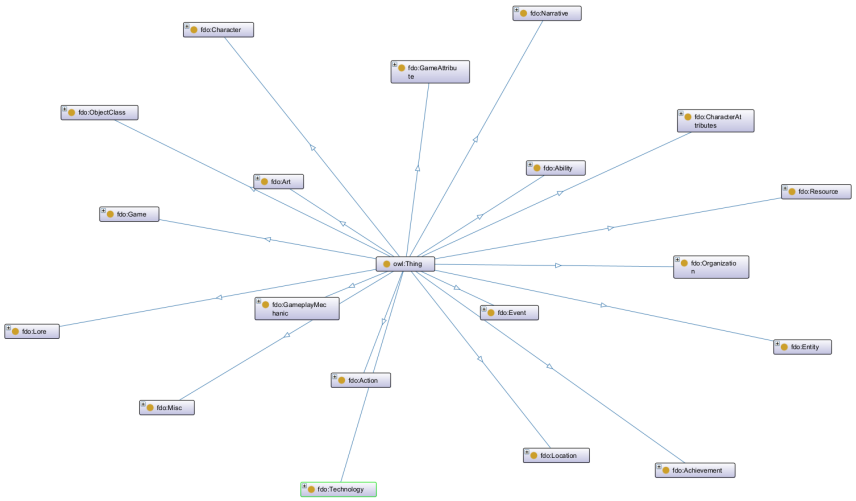

Fig. 2. Fandom ontology top level classes

- Achievement - 297
- Monster - 255
- Weapon - 234
- Chapter - 228
- Ship - 212
- Song - 199
- MusicTrack - 198
- VideoGame - 173
- MilitaryUnit - 172
- Animal - 171

Properties And top 25 properties (Property Name, Type:Object or Data, and number of uses)

- affiliatedWithCharacter - object - 3845
- hasType - object - 2592
- hasGender - object - 1767
- hasLocation - object - 1490
- locatedIn - object - 1162
- hasHealthPoints - data - 953
- hasSource - object - 951
- hasSpecies - object - 889
- hasOccupation - object - 785
- hasAlias - data - 693
- appearsInGame - object - 672
- hasAbility - object - 665
- hasEffect - object - 604
- releaseDate - data - 551
- hasAffiliation - object - 539
- hasStatus - object - 496
- hasAge - data - 496
- hasCost - data - 494
- hasRarity - data - 457
- hasRelative - object - 443
- hasDamage - data - 441
- hasEyeColor - data - 440
- hasWeight - data - 392
- hasRace - object - 371
- affiliatedWith - object - 343

4 Data Integration and Conversion

4.1 Non-RDF data

Metacritic and steamDB :

By using **Bs4**(BeautifulSoup), a Python library that provides a simple interface to parse and manipulate HTML/XML documents without manual parsing and then converts a document into a structured tree, and html.parser, an built-in parer in Python, we scrapped the relevant data from **Metacritic**. We got the information about the games released between 2010 and 2024 which have theirs critic metascores on the **Metacritic** and integrated it into a CSV table for future use.

Similar to what we did for **Metacritic**, BS4 was also used to convert a game table on the SteamDB, showing game names and the number of players: current, 24-hour peak, and all-time peak, into a list and then converted it into a CSV file.

Using the **rdfilb** library in Python, we converted these CSV tables from **Metacritic** and **SteamDB** into a RDF graph. Each game has a specific URI and is added to a class: Game, and there are four classes of platforms, publishers, developer and genre for the relevant data for games from **Metacritic**. The games scraped from **Metacritic** are connected to their title, score, released date via the respective DataProperties and connected to their platforms, developer, publisher, and genre via ObjectProperties. While the games from **SteamDB** have 4 different objects: name, current player, 24h peak player, all time peak player and corresponding DataProperties. To deal with the situation where some game scores are missing from **Metacritic**, we have assigned the value of 0 to these missing data. In addition, we removed the punctuation from the current player, 24h peak player and all time peak player data for the games from **SteamDB** to make the conversion easier.

Fandom :

We then scrapped the relevant information about the our chosen games from **Metacritic** on **Fandom Wikis**. Unfortunately, not all of the games have their own wikis and of those that does, only about 1650 games had an English wiki. Another reason for such small fraction could be that some wikis have strangely formatted URIs. Most of the wikis have the URI 'game name' + 'fandom.wiki.com' but some games use abbreviations or suffixes like "game".

From each fandom directory of a specific game, we sampled 100 objects randomly in the first five pages and scraped their information boxes as HTML elements, which we preprocessed to remove some html tags and css formatting that doesn't add much to meaning but uses limited available context window for inferencing. For more detail on extracting classes and proiperties please section 3 on Methodology.

4.2 RDF Data

We didn't export existing RDF dataset into our ontology since most of them has their own endpoints where we can run SPARQL queries. The only thing we need for using them is to implement mapping. For this we're assuming that all external datasets (WikiData and DBPedia in our case) are using same labels for each game as we do (since the game has only 1 official name) and this way we can enhance our data with additional information for instance information about game **publisher** or **developer**.

	Game	Game Label	Current	24h Peak	All-Time Peak	Developer	Publisher	Release Date
0	http://example.org/Grand_Theft_Auto_V	Grand Theft Auto V	153040	153040	364548	Rockstar North	Softclub	2022-03-15T00:00:00Z

Fig. 3. Example of query enhanced with data about developer, publisher and release date from WikiData

5 Knowledge Graph

To create the knowledge graph we use python library 'RDFlib', and store pre-processed in relational tables or CSV as triples. For the implementation see code in the Jupyter notebook[6]. Full statistics for combined Fandom and Metacritic data are provided below.

Metrics

- Axiom 274,634
- Logical axiom count 206,862
- Declaration axioms count 7,157
- Class count 3,477
- Object property count 2,226
- Data property count 1,566
- Individual count 60,884
- Annotation Property count 2

Class Axioms

- SubClassOf 3,900

[6] https://github.com/mokrota21/GameOntologyProject/blob/erbol/populate_graph.ipynb

Property Axioms

- SubObjectPropertyOf 695
- SubDataPropertyOf 217

Individual Axioms

- ClassAssertion 56,247
- ObjectPropertyAssertion 92,582
- DataPropertyAssertion 37,657
- SameIndividual 15,564
- AnnotationAssertion 60,615

6 Inferences

As can be seen from the metrics summary, the only axioms that we took advantage of, are 'subClassOf' and 'subPropertyOf'. These inferences help us keep the variety meaning created by individual users, without loosing the richness of game specific data - and still allowing for a high level querying and data finding.

7 SPARQL Queries

We have eight SPARQL queries in total, which respectively show how to get game genres, voice actors, what genre a voice actor is in, all the games with certain label or comment, and the number of game players, the number of the race of characters, how to combine information from our ontology and Wikidata, as well as a query to our scraped Metacritic data. As the last query is described in the RDF Data section, this section would include the first four queries, and the rest of them would be described in the next section. Pictures of the example query results follow.

	Game	Game Label	Genre Label
0	http://fandom.com/resource/20XX	20XX	Roguelike, Action, Platformer
1	http://fandom.com/resource/30XX	30XX	Roguelike
2	http://fandom.com/resource/RAD	RAD	Roguelike
3	http://fandom.com/resource/Balatro	Balatro	Roguelike, Deckbuilder
4	http://fandom.com/resource/Black_Future_2788	Black Future '88	Action roguelike
5	http://fandom.com/resource/DandyAce_Wiki	Dandy Ace	Action Roguelike
6	http://fandom.com/resource/Loot_River	Loot River	Indie action roguelike, dark fantasy
7	http://fandom.com/resource/Roguebook	Roguebook	Roguelike deck-builder
8	http://fandom.com/resource/Sundered	Sundered	Roguelike, Metroidvania
9	http://fandom.com/resource/Wildfrost	Wildfrost	Fantasy, Roguelike deckbuilder, Card Battler

Fig. 4. Outcome of a query for games, game labels, and game genre labels. In this example, we queried for games with 'roguelike' in the genre labels

	Game Label	Character	Character Label	Voice Actor
0	Medal of Honor: European Assault	http://fandom.com/resource/28th_Infantry_Division	28th Infantry Division	http://fandom.com/resource/Matthew
1	Medal of Honor: Allied Assault, Medal of Honor...	http://fandom.com/resource/29th_Infantry_Division	29th Infantry Division	http://fandom.com/resource/Matthew
2	Pokémon the Series: The Beginning	http://fandom.com/resource/AJOn_hand	A.J.	http://fandom.com/resource/MaddieBlaustein
3	Pokémon Generations	http://fandom.com/resource/AZ_Generations	AZ	http://fandom.com/resource/KeithSilverstein
4	Pokémon Generations	http://fandom.com/resource/AZ_Generations	AZ	http://fandom.com/resource/TakayaHashi
...	...	...	...	...
154	Titanfall 2, Stories from the Outlands, Apex L...	http://fandom.com/resource/Viper	Viper	http://fandom.com/resource/EvanBoymel
155	Medal of Honor, Medal of Honor: Warfighter	http://fandom.com/resource/Voodoo	Voodoo	http://fandom.com/resource/JonBruno
156	Chapter 2	http://fandom.com/resource/Dolores	Dolores	http://fandom.com/resource/JeanRudaHabrukowich
157	Chapter 1, Chapters 2-3, Chapter 4	http://fandom.com/resource/Rose	Rose	http://fandom.com/resource/ClareCorbett
158	Chapter 1, Chapters 2-3, Chapter 4	http://fandom.com/resource/Rose	Rose Goodwin	http://fandom.com/resource/ClareCorbett

Fig. 5. Outcome of a query for game labels, characters in games, character labels, and voice actors of characters.

	Voice Actor	Character	Character Label	Game	Game Label	Genre Label
0	http://fandom.com/resource/NatsumiFujiwaraJapa...	http://fandom.com/resource/Asuma_Soga	Asuma Soga	http://fandom.com/resource/AnonymousCode	Anonymous;Code	Visual novel
1	http://fandom.com/resource/TomokazuSugitaYChang	http://fandom.com/resource/Cross_Yumikawa	Cross Yumikawa	http://fandom.com/resource/AnonymousCode	Anonymous;Code	Visual novel
2	http://fandom.com/resource/EmiLo	http://fandom.com/resource/Iroha_Kyogoku	Iroha Kyogoku	http://fandom.com/resource/AnonymousCode	Anonymous;Code	Visual novel
3	http://fandom.com/resource/MarikoHonda	http://fandom.com/resource/Iroha_Kyogoku	Iroha Kyogoku	http://fandom.com/resource/AnonymousCode	Anonymous;Code	Visual novel
4	http://fandom.com/resource/KariWahlgren	http://fandom.com/resource/Elena_Ivanova	Elena Ivanova	http://fandom.com/resource/Vanquish	Vanquish	Third-person shooter
5	http://fandom.com/resource/BenitoMartinez	http://fandom.com/resource/Professor_Francois_...	Professor Francois Candide	http://fandom.com/resource/Vanquish	Vanquish	Third-person shooter
6	http://fandom.com/resource/StevenBlum	http://fandom.com/resource/Robert_Burns	Robert Burns	http://fandom.com/resource/Vanquish	Vanquish	Third-person shooter
7	http://fandom.com/resource/GideonEmeryEnglishn...	http://fandom.com/resource/Sam_Gideon	Sam Gideon	http://fandom.com/resource/Vanquish	Vanquish	Third-person shooter
8	http://fandom.com/resource/MarcWorden	http://fandom.com/resource/Victor_Zaitsev	Victor Zaitsev	http://fandom.com/resource/Vanquish	Vanquish	Third-person shooter

Fig. 6. Outcome of a query for the genre of game a voice actor is in.

	Subject	Label	Comment
0	http://fandom.com/resource/Air_Host	Air Host	Air Host is a non-playable character in LEGO C...
1	http://fandom.com/resource/LEGOCityUndercover	LEGO City Undercover	N/A
2	http://fandom.com/resource/Albatross_Island	Albatross Island	Albatross Island is a district featured in LEG...
3	http://fandom.com/resource/LEGOCityUndercoverT...	LEGO City Undercover: The Chase Begins	N/A
4	http://fandom.com/resource/Auburn	Auburn	Auburn is a district in LEGO City, featured in...
5	http://fandom.com/resource/Apollo_Island	Apollo Island	Apollo Island is a district featured in LEGO C...
6	http://fandom.com/resource/Athena	Athena	Athena is a compact vehicle found in LEGO City...
7	http://fandom.com/resource/Art_Gallery	Art Gallery	An Art Gallery in LEGO City Undercover serving...
8	http://fandom.com/resource/Attract_Bricks	Attract Bricks	Attract Bricks are collectibles in LEGO City U...
9	http://fandom.com/resource/Attract_Studs	Attract Studs	Attract Studs is a collectible item in LEGO Ci...
10	http://fandom.com/resource/Auburn_Bay	Auburn Bay	Auburn Bay is a location featured in LEGO City...
11	http://fandom.com/resource/Auburn_Bay_Bridge	Auburn Bay Bridge	Auburn Bay Bridge is a district featured in LE...
12	http://fandom.com/resource/Auburn_Station	Auburn Station	Auburn Station is a train station located in t...
13	http://fandom.com/resource/Autos_Store	Autos Store	The Autos Store is a location in LEGO City Und...
14	http://fandom.com/resource/LEGOCity	LEGO City	N/A
15	http://fandom.com/resource/Barry_Smith	Barry Smith	Barry Smith is a plumber and kung-fu instructo...
16	http://fandom.com/resource/Bastion	Bastion	Bastion is an emergency vehicle featured in LE...
17	http://fandom.com/resource/Bea_Heckerson	Bea Heckerson	Bea Heckerson is a police officer in LEGO City...

Fig. 7. Outcome of a query for the games with 'LEGO City' in the labels and comments.

8 Data Science Pipeline

There are three query results that have been visualized, giving our users the options available to them based on their queries. For example, users can get the most played games by number of players, the games with the most character races, or consoles with the average highest rated games, which can be displayed directly from the graphs.
The pipeline perfectly addresses the competency questions as our users could get visual results of the most played games, games with the most races of characters, and consoles with the highest average rated games based on their requirements.

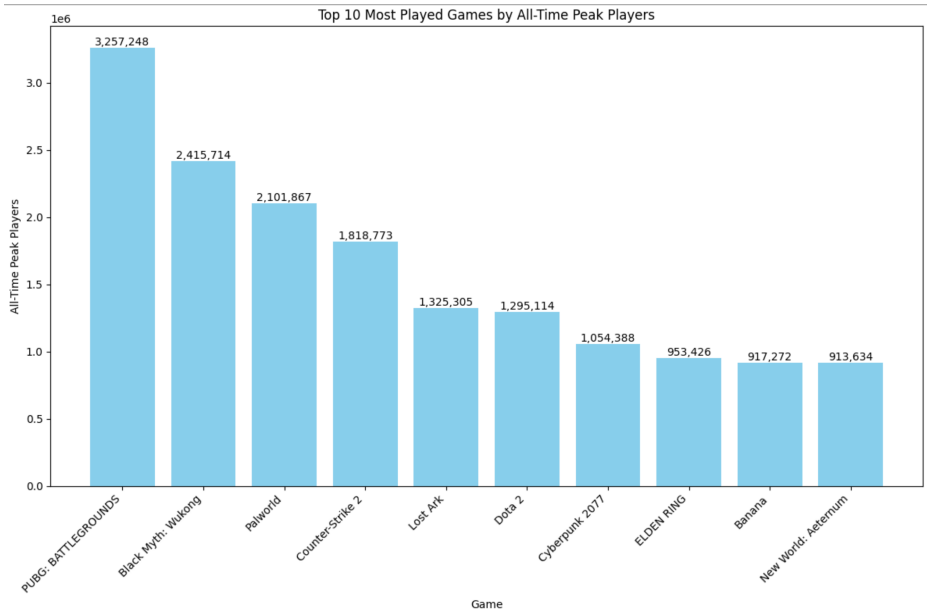

Fig. 8. This bar chart shows the top 10 games with the players of all time.

	Game	Character Count
0	http://fandom.com/resource/GrandTheftAutoV	12
1	http://fandom.com/resource/FreedomPlanet2	6
2	http://fandom.com/resource/Prey2017	5

Fig. 9. Outcome of a query for the top 3 games with the most races of characters.

	Game	Game Label	Current	24h Peak	All-Time Peak
4916	http://example.org/PUBG%3A_BATTLEGROUNDS	PUBG: BATTLEGROUNDS	678431	700726	3257248
1025	http://example.org/Black_Myth%3A_Wukong	Black Myth: Wukong	178923	178923	2415714
4934	http://example.org/Palworld	Palworld	24412	24412	2101867
1611	http://example.org/Counter-Strike_2	Counter-Strike 2	1369985	1395611	1818773
3997	http://example.org/Lost_Ark	Lost Ark	21976	25377	1325305
2106	http://example.org/Dota_2	Dota 2	612187	619332	1295114
1729	http://example.org/Cyberpunk_2077	Cyberpunk 2077	37710	37710	1054388
2273	http://example.org/ELDEN_RING	ELDEN RING	49050	49050	953426
889	http://example.org/Banana	Banana	426998	432366	917272
4606	http://example.org/New_World%3A_Aeternum	New World: Aeternum	40121	50621	913634

Fig. 10. Outcome of a query for the games with game labels, the number of current players, 24h peak players, and all time players.

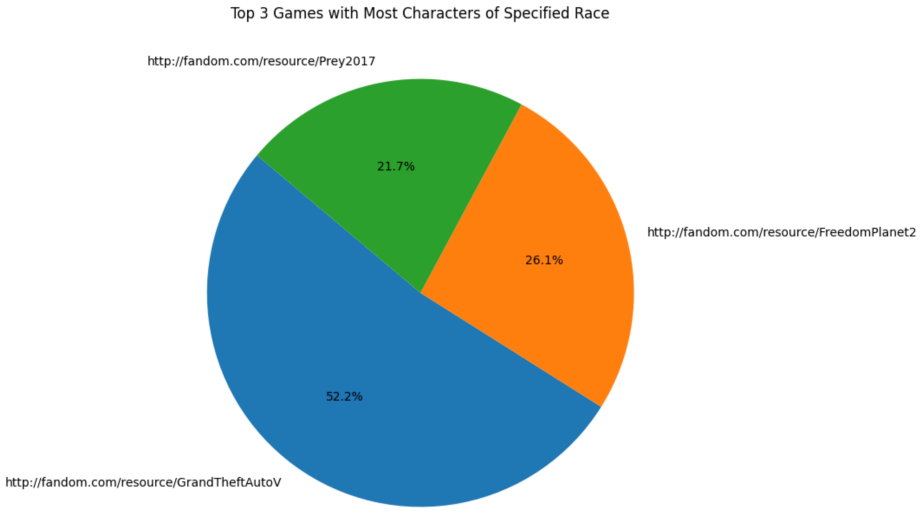

Fig. 11. This pie chart shows the top 3 games with the most races of characters.

	Platform	Average Rating
0	PlayStation 2	77.921053
1	Meta Quest	75.040816
2	Xbox Series X	74.258065
3	PlayStation 5	74.110769
4	PlayStation	73.833333
5	iOS (iPhone/iPad)	73.262737
6	Nintendo Switch	72.850880
7	Xbox One	72.205940
8	PlayStation 4	71.900390
9	PC	71.690940

Fig. 12. Outcome of a query for the top 10 consoles with highest rated games dedicated to them.

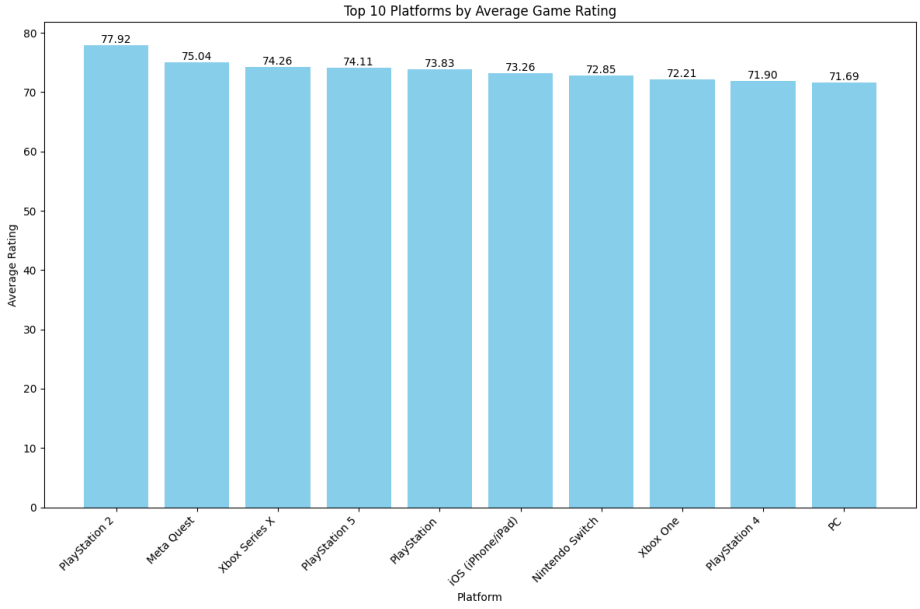

Fig. 13. Bar graph depicting the top 10 consoles with highest average rated games for it.

9 Contributions and Justification

9.1 Jan Czarnecki

- Queries
- Query Visualization
- Formalization Graph

9.2 Erbol Esengulov

- Scraped data from **Metacritic** and **Fandom**.

9.3 Sixuan Dou

- Scraped data from **SteamDB**.
- Converted Non-RDF data from **Metacritic** and **SteamDB** to RDF triple.

9.4 Aleksey Martynyuk

- LLM prompt engineering
- LLM batch processing
- Creating vector embeddings and clustering
- Ontology engineering methodology for Fandom data