Data preparation, information retrieval and ontology design applied to computer video games

Bruno Piedade
Faculdade de Engenharia da Universidade do Porto
Portugal, Porto
up201505668@fe.up.pt

Daniel Marques
Faculdade de Engenharia da Universidade do Porto
Portugal, Porto
up201503822@fe.up.pt

Abstract

This report initially describes the process of obtaining, preparing and analyzing a dataset regarding video games and their user reviews. Subsequently, the report documents the process of information retrieval on the dataset, including tool selection, document definition, indexing, retrieval, and evaluation. Finally, it summarizes the process of modelling, populating and querying an ontology. The data was obtained using crawling and scraping techniques on the Steam store website, utilizing the Scrapy framework. After collecting the data, it was refined using Python scripts to remove duplicate entries and finally merged to form a dataset exported as JSON lines. The resulting dataset contains 1,013 video games and a total of 45,619 user reviews. A conceptual data model has been drawn to better visualize the obtained attributes and entities. Furthermore, relevant statistical information has been plotted using Seaborn and Matplotlib to characterize the dataset. This analysis revealed substantial growth in game releases and user review submissions around the year of 2013. Afterwards, *Solr* was selected as the search tool and the game documents were defined, along with its reviews. After difficulties in indexing with a parent-child hierarchy for games and reviews, respectively, a flat approach was used, flattening the user reviews into each game document. The indexing process explored some of *Solr*'s analyzer capabilities, such as modifying the schema to accommodate English language indexing improvements, like stemming and synonyms definition. After indexing the dataset, three information needs were defined, along with its queries and information retrieval approaches which varied through field boosting, phrase searching, and synonym usage. These approaches were then compared and evaluated based on metrics like precision-at-k, average precision, and recall. Some information need results were also compared to Steam's search system. The results revealed that the variations from the default retrieval approaches displayed improvements in the considered metrics. In the final part of the project, an ontology was built using *Protégé* with the goal of representing knowledge regarding a game and its user reviews. The ontology was based on the conceptual data model and integrates other ontologies (e.g. the *Video Game Ontology*) since interoperability is encouraged in Semantic Web. Afterwards, the ontology was populated with a small portion of the dataset using the *Cellfie* plugin for *Protégé*. Finally, to ensure the usefulness of our ontology, a set of queries based on common retrieval tasks in the domain were defined, implemented in SPARQL and executed with the query engine *ARQ*.

CCS Concepts

• **Information systems** → **Information storage systems**; **Information retrieval**; **Semantic web description languages**; *Data management systems.*

Keywords

video game, user review, dataset, scraping, crawling, conceptual model, domain, refinement, statistics, information retrieval, semantic web

1 Introduction

The video game industry has been growing steadily throughout the years generating a total of 135 billion dollars in 2018 [5]. What was initially a niche hobby, gaming has exploded into popularity reaching a huge audience of all age-groups from the more casual to the most hardcore. PC gaming, in particular, has been gaining traction, following along with the trend. One of the major contributions was the appearance of quality digital distribution services, such as Steam and GOG which gave customers access to an immense catalogue of games readily available at any time. Steam has become one of the biggest hubs of video game content on the web, boasting a hefty catalogue of over 35,000 games, more than 1 billion registered accounts and an average of 8.5 million concurrent active users [6, 30] making it one of the best sources for computer video game information including the latest releases and user reviews for each game. In this context, video games are currently a hot topic and interest in them is higher than ever. Considering the demand for information and the data available it makes sense to combine both, giving users the ability to search for the games that interest them.

As new online video game stores arise, so does the need to stand out. The Epic Games store [13], a digital video game storefront, chose to encourage game developers to exclusively sell their products to them [26], instead of other stores like Steam. However, this type of approaches scatters the data across the Web. With Semantic Web the information is given well-defined meaning (semantics), thus enabling machines to understand the data, despite their source [8].

In this report, we describe the stages of data preparation, information retrieval and ontology design on the topic of computer video games. The process of data preparation required choosing a data source, picking a retrieval method, assessing the dataset's quality and applying refinements, analysing and characterizing the dataset, building a conceptual data model and defining possible retrieval tasks in this domain. The process of information retrieval involved selecting a search tool, defining the documents

and collection, indexing the documents, describing information needs, answering them using different approaches and comparing results. With the goal of contributing to Semantic Web, we've modelled an ontology to represent knowledge regarding video games, populated it with information from the dataset, defined a set of user needs, implemented queries that satisfy them and executed them on the ontology.

2 Data preparation

2.1 Data sources

All of the data originates from the Steam store website [6]. Steam is a video game digital distribution service platform developed by Valve Corporation. While Steam contains a variety of video game-related information, such as hardware and game development tools, the focus of the project is on video games and their user reviews.

Since Steam is a trusted source with numerous details regarding video games, the team decided to use it as its only source of information for the dataset.

2.2 Retrieval methods

The data was extracted from the Steam store website [6] using web crawling and scraping methods.

The scraper used was based on a project developed by Andre Perunicic (*prncc*) and is available on Github [23] at the time of writing. The project was archived and last updated on February 11, 2018. As a result, the scraper was not fully up to date with Steam's most recent changes and required extensive fine-tuning and fixing in order to retrieve the desired information.

The scraper is written in Python [12] and is built using the Scrapy framework [17]. The dataset features two major classes: games and reviews. These classes are extracted independently from Steam and are later merged using a Python script. To retrieve games, the scraper uses crawling techniques that receive a starting URL as an input and proceeds to crawl into each game page it finds, which is then parsed, and advances into the *next* page, if available. The reviews are retrieved differently: the scraper is also given a starting URL, but instead of crawling, it parses the available reviews and, because the page features infinite scrolling, it forces Steam to send more reviews by simulating scroll input.

2.3 Dataset quality and refinement

As the data was collected by the group using the scraper most of the quality assurance and refinement was done *a priori* during the fine-tuning of the scraper. In order to achieve this, some care went into ensuring that only relevant data was collected and that they were in the expected structure. Some of these measures include: adding default values for fields when no information was available or removing them entirely when they weren't applicable; removing duplicate user reviews (resulting from a Steam display bug, as each user is limited to one review per game); supporting different date formats and choosing a standard (dd-mm-yyyy); removing unwanted characters from text attributes (for example, HTML tags and some Unicode characters).

It was possible to ensure data consistency because the data in the Steam store website [6] was itself coherent and structured, meaning that there were no instances of inconsistencies commonly found in public datasets such as multiple names for a single entity. Steam's game submission system requires the publisher to fill detailed forms which are then analyzed by Steam's support services. This system guarantees Steam's authenticity as a data source.

2.4 Dataset characteristics

Since the dataset was obtained using crawling and scraping, there was complete control over how to export the collected information. The team decided to export as a text file, under a JSON lines format [29]. This format allowed us to have multiples reviews per video game entry, forming only one dataset, which is something that would otherwise be too complex or inefficient in other formats such as CSV or XML. An example of a game and a user review entries in the dataset can be seen in the Appendices A.1 and A.2, respectively.

The dataset represents a sample of Steam's products, featuring 1,013 video game entries with a variable number of reviews each, resulting in a file size of 52.3 MB. It contains a total of 45,619 reviews, submitted by 40,021 distinct users.

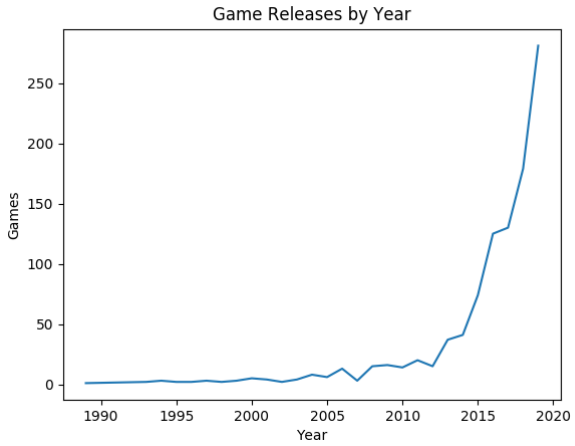

Figure 1: Game releases by year.

The line plot in Figure 1 illustrates the game releases over the years, in the collected sample. At first, the chart might indicate a sudden growth around 2015. However, that is not the case since the gaming industry has been steadily growing for decades [27]. The unexpected nature of the plot can be explained by the retrieval methods used for obtaining the sample of games. When crawling and scraping, it's common to use a starting URL from which the program starts to find game links and anchors to the next page. The starting URL used was Steam's game search page with relevance sorting [7]. It's thus expected that more recent games are "more relevant" according to Steam.

The line plot in Figure 2 describes a more linear temporal distribution of user review submissions compared to game releases, although still biased towards more recent games. This is the expected behaviour for two reasons. Firstly, while Steam was launched in 2003, it was only 7 years later that the platform introduced a so

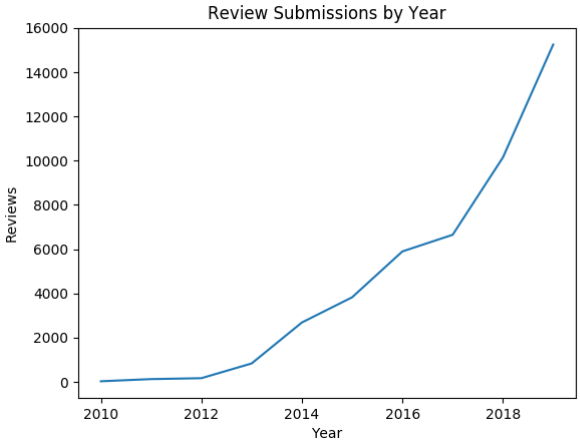

Figure 2: User review submissions by year.

called "Recommendation system", which was eventually overhauled and renamed to "Review system" in 2013 [14]. Secondly, users are able to review any game, independently of its release date, as long as they've bought it from the Steam store. The bias towards more recent games is also justified since the dataset contains more recent products and Steam's user base is ever increasing [30].

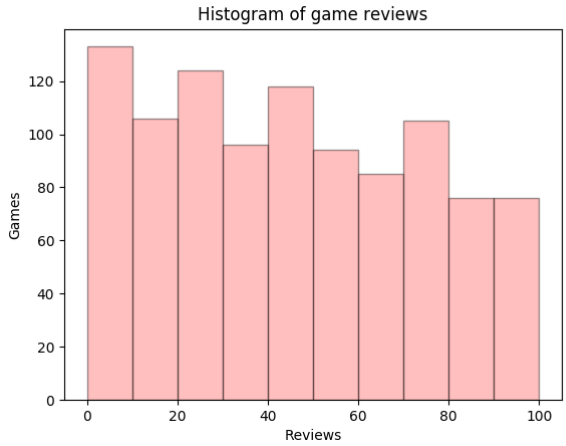

Figure 3: Reviews per game.

The histogram in Figure 3 demonstrates a moderate consistency in reviews per game. This behaviour is a result of crawling since the number of scraped reviews had to be limited as some games have over 3 million user reviews. To ensure some degree of deviation, the limit of scraped reviews per game was set to a random number with a maximum of 100.

The bar chart in Figure 4 shows all the distinct genres present in the dataset and their frequency in games. This is an important metric to consider for future work regarding information retrieval on the dataset. For example, searching for action related keywords

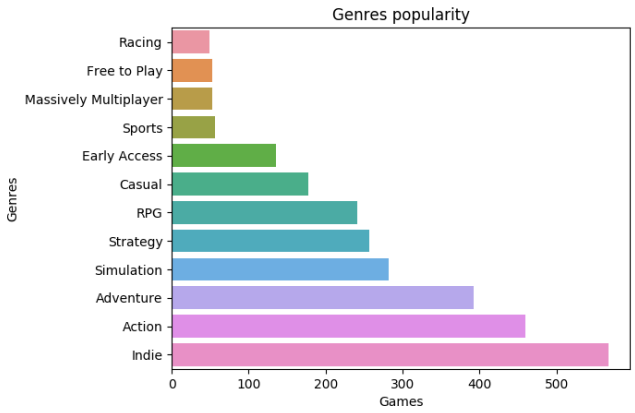

Figure 4: Game genres popularity.

like "guns" and "destruction" will lead to more games being returned by the search engine with a high relevance rating. On the other hand, less, but still relevant, video games are expected to be returned for racing related searches.

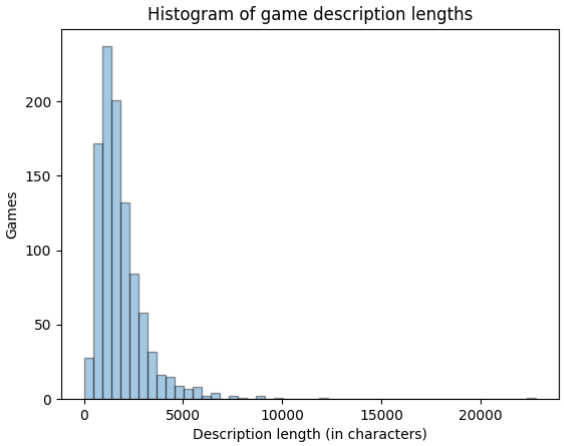

Figure 5: Textual game description length.

The histogram in Figure 5 identifies and characterizes another important metric for information retrieval. It illustrates the number of characters used in the "About" section of a Steam game, which describes a game in great detail. As a game description, it has enormous potential for matching search keywords for that game. Its length can give an estimate of how difficult it would be to find them on this attribute.

2.5 Conceptual data model

The conceptual model for the data (shown in Figure 6) was designed after deciding what data should be collected and how they relate to each other. Note that this model applies to the data and also to the domain since all entities of interest and their relations are depicted. Other concepts exist in the video game theme (for example, financial

information). However, these are irrelevant for the scope of this project as its main focus is the games themselves and how the users feel about them.

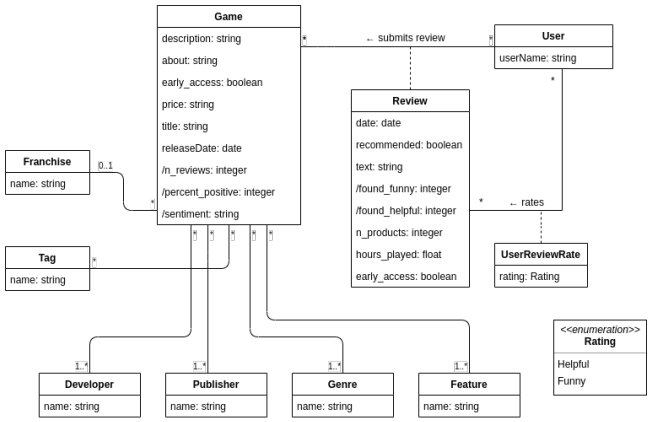

Figure 6: Conceptual data model.

Around the centre are the two main classes which represent the core entities of the dataset: the Game class and the Review class.

- **Game** - Class which stores all the relevant information about a game.
  - **Title** - Name of the game as it appears on Steam.
  - **Description** - Concise textual overview of the game.
  - **About** - Detailed textual description of the game.
  - **Early access** - Indicates whether or not the game is currently under early access meaning that it's being financed by the customers while under development.
  - **Price** - The price asked to purchase the game.
  - **ReleaseDate** - Date when the game was or is going to be officially launched.
  - **N reviews** - Total number of user reviews.
  - **Percent positive** - Percentage of users that recommended the game in their reviews.
  - **Sentiment** - Textual representation of the Percent positive attribute, for example, *Mostly Positive*.
- **Franchise** - Series of game products around the intellectual property to which the game belongs.
- **Tag** - User-defined classifications that identify characteristics of the games.
- **Developer** - Entity responsible for the development of the game.
- **Publisher** - Entity responsible for distributing, marketing and possibly financing the game.
- **Genre** - Classification of the game based on gameplay interaction, for example, *Action*.
- **Feature** - General characteristics ranging from Multiplayer to Virtual Reality support.
- **Review** - Association class which stores all the relevant information about a user review regarding a certain game.
  - **Date** - When the review was posted.
  - **Recommended** - Indicates whether the user recommends the game or not.
  - **Text** - Textual content of the review.
  - **Found funny/Found helpful** - Number of users that rated the review as funny/helpful.
  - **N products** - Number of products the user owned at the time of review submission.
  - **Hours played** - Number of hours the user played of the game at the time of review submission.
  - **Early access** - Indicates whether the game was in early access or not at the time of review submission.
- **User** - Class which stores all relevant information about a user.
  - **Username** - Name of the user on the Steam platform.
- **UserReviewRate** - Association class which stores information regarding the rating of a review by a user.
  - **Rating** - Whether a user rated review that is not his own as funny/helpful.

2.6 Retrieval tasks

The expected return value of the retrieval tasks are documents which represent games and reviews as these are the focus of the project. However, there is the possibility of returning other types of variables including other entities, specific fields of some entities and processed numeric values.

Possible retrieval tasks include:

- What are the most recommended games by the users?
- What games are about a specific theme?
- What are the most recent games?
- What games belong to a specific genre?
- What games support a specific feature?
- What games are free-to-play?
- Which game reviews are considered the most helpful?
- Which game reviews are from users with the most hours played before writing the review?

Some of the aforementioned tasks can be combined into more complex tasks. For example:

- What are the most recent games of a specific genre?
- What are the most recommended free-to-play games?

3 Information retrieval

3.1 Tool selection

The group started by experimenting with information retrieval tools, namely *Solr* [11] and *Elasticsearch* [20], in order to understand and make an informed decision regarding which one to choose. On one hand, although the installation process was straight-forward, the group felt that *Solr* required a large initial effort in order to understand the basics. Furthermore, (as discovered later) the documentation regarding more advanced topics was somewhat lacking which was offset thanks to the large amount of community discussion available, mainly due to the age of this tool. However, after the initial hurdle, the graphical interface proved itself as an advantaged, being simple and easy to use. On the other hand, *Elasticsearch* was overwhelming as this software has been going beyond a text search tool, adding features like log analysis and visualization, thus increasing its learning curve [19]. In the end, after considering the

pros and cons and having practical experience with both, and despite the initial challenges, the group ultimately chose *Solr* as it seemed appropriate for supporting the previously defined retrieval tasks and, having been in the market for longer, a more mature tool with higher community support and great performance.

3.2 Collection and documents

The initial intent for the system was the retrieval of both games and reviews. This required the storage of each individual entity as its own document. However, due to challenges found during the indexing process (discussed in the next section), the focus shifted towards only returning games. Subsequently, the end result consists of a single collection, containing a set of documents, each regarding a game which, in turn, corresponds to the data in a single line of the original dataset.

3.3 Indexing

The first stage for indexing the information of the dataset was deciding the most optimal document granularity and how to split the data for each in order to meet the retrieval requirements. As previously established, the dataset contained one game per line, each with the multiple fields for its details and an array for the associated reviews. This meant that indexing both games and reviews as individual documents required splitting the original data accordingly which could be achieved through *Solr*'s JSON transformation and indexing utilities. With this in the mind, the group considered three main options:

(1) Index the dataset as is, according to the default behaviour, meaning the outcome would be one document per game, with its nested reviews flattened for each.
(2) Index using the split parameter only for the reviews, resulting in one document per review with the associated game fields repeated in each one.
(3) Index using the split parameter for both the games and review, resulting in one document per game and review.

The option of splitting the reviews (2) was discarded early on. Even though this approach met the group's requirements, the repetition of data in each document was significant. The group then considered splitting the games and reviews (3) and explored its potential. This approach retained the original structure of the dataset without any redundancy and with the option to reassemble the nested fields as required at query time via *Solr*'s *block join query parsers* and *child doc transformer*. However, major difficulties were found when trying to boost fields while retaining the parent/child relationship. Specifically, boosting a game's score based on its reviews and vice-versa. Due to this, in the end, the group opted to index the dataset as is, flattening the nested fields (1), which resolved the previously described difficulties. One of the main disadvantages of this approach was losing the ability to directly return individual reviews or games, however, one can still post-process the data in the front-end and adapt the output according to the needs at hand.

The next step was refining the auto-generated schema in *Solr*'s schemaless mode, which includes all the fields present in the dataset. This involved tuning some of the fields, in particular, the game's *description*, *about*, *genres*, *features*, and *tags*, and the review's *text*. The most relevant change for these fields was modifying their field type from *text_general* to *text_en*. This modification allowed for English language-aware optimizations and improvements. The indexing and search processes were optimized by the usage of English stop words, which ignores words that are considered irrelevant for information retrieval [18]. Additional improvements were possible through English specific stemming, which reduced inflectional and derivationally related forms of words [18].

3.4 Retrieval

Building upon the previously defined retrieval tasks, it was necessary to expand and adapt the retrieval objectives for the system which should, at this stage, be mostly focused in exploring the system's full-text search capabilities and strictly returning games. In this sense, a new set of retrieval tasks were defined as follows:

(1) What games are about a specific theme?
(2) What games are aimed at adult audiences/contain mature content?
(3) What games can be played in Virtual Reality (VR)?

Satisfying these tasks required extensive exploration of the textual fields of each entity, namely, the title, description and about of each game and the text (i.e content) of each review. In order to perform the queries, *Solr*'s eDismax parser was used, as it supported full-text search according to a set of parameters. Of particular importance, the *qf* (query fields) parameter was extensively used for choosing and boosting the considered fields in the search.

The first information need is parameterized by its *theme*. An instance of this information need was made with the theme being 'building a city/village'. The resulting query was simplified as to not restrict the type of area to build, time period, or for what purpose. Therefore, the query became *q = 'build'*. Two retrieval options were explored. Both searching the fields *title*, *description*, *about*, and *reviews.text*. The difference between both approaches lies on boosting the reviews' text by a factor of 5.

The underlying information need for the retrieval task (2) is finding games that are aimed at adult audiences (over 18 years old) or, in other words, those that contain mature content such as explicit violence or sexual themes. For this purpose, the search query chosen was *q = 'mature'* as it represents the most well-established term when referring to this type of content in the gaming ecosystem mainly due to its adoption by the gaming rating boards such as the Entertainment Software Rating System (ESRB) [4]. Similarly to the process behind the retrieval task (1), two search approaches were used. Both were applied to the fields *description*, *about* and *reviews.text*. The difference between the two was applying a boost factor of 2 to the field *reviews.text*.

The retrieve task (3) defines the information need of finding games that can be played with virtual reality devices. The expression 'virtual reality' is commonly abbreviated into 'VR' amongst the gaming community. Exploring the retrieval system to satisfy this information need, four approaches were examined. Similarly to other retrieval tasks, the fields to search were the *title*, *description*, *about*, and *reviews.text*. Firstly, the phrase 'virtual reality' was used as query, without quotation marks. Secondly, the same expression was used as query, but this time quotation marks were used to ensure the order is respected and an exact match is found. Thirdly, since the acronym 'VR' is widely used as a substitute for

	Game	R	P@k
1	Armor Clash 3	Y	1.00
2	Constructor Plus	Y	1.00
3	Wooden Battles	N	0.67
4	Supremacy 1914	N	0.50
5	MicroTown	Y	0.60
6	Overdungeon	N	0.50
7	Buoyancy	Y	0.57
8	Journey Of Life	N	0.50
9	Transport Fever	Y	0.56
10	Low Magic Age	N	0.50

Table 1: Results from Retrieval Task (1) - Default

	Game	R	P@k
1	Imperium Romanum Gold Edition	Y	1.00
2	Rise to Ruins	Y	1.00
3	Workers & Resources: Soviet Republic	Y	1.00
4	Project Highrise	Y	1.00
5	From the Depths	N	0.80
6	Children of the Nile: Enhanced Edition	Y	0.83
7	Anno 1404	Y	0.86
8	My Lands: Black Gem Hunting	Y	0.88
9	8-Bit Armies	Y	0.89
10	Planet Coaster	Y	0.90

Table 2: Results from Retrieval Task (1) - Boosted

	Game	R	P@k
1	Lady and Blade	Y	1.00
2	Demoniaca: Everlasting Night	Y	1.00
3	Dreamfall: The Longest Journey	N	0.67
4	Catherine Classic	Y	0.75
5	Dex	N	0.60
6	The Witcher 2: Assassins of Kings Enhanced Edition	Y	0.67
7	Grandia® II Anniversary Edition	N	0.57
8	Blackguards	N	0.50
9	CUSTOM ORDER MAID 3D2 It's a Night Magic	Y	0.56
10	The Letter - Horror Visual Novel	N	0.50

Table 3: Results from Retrieval Task (2) - Default

	Game	R	P@k
1	The Witch's Love Diary	Y	1.00
2	War of Rights	Y	1.00
3	Demonheart	Y	1.00
4	Love Ribbon	Y	1.00
5	WILL: A Wonderful World	Y	1.00
6	LISA	Y	1.00
7	Lady and Blade	Y	1.00
8	Blackguards	N	0.88
9	Pummel Party	Y	0.89
10	Negligee	Y	0.90

Table 4: Results from Retrieval Task (2) - Boosted

the complete expression, *Solr*'s synonym capabilities were used to establish this correspondence and afterwards the same quoted expression was searched. Finally, because 'VR' is an established tag on Steam, the *tags* field was searched for this acronym, instead of the aforementioned fields. The last retrieval method was carried out on structured data, thus becoming *less interesting* for the purpose of this project. However, its results are important as they can be seen as an accurate approximation to ground truth, without having to manually categorize dozens of documents.

Besides the 3 presented retrieval tasks, many others were considered and explored but ultimately were deemed not relevant as either: a) were very similar to database querying and therefore had no value for text-based searching; b) the initial results without boosting were already satisfying and there wasn't any or a negligent margin for improvement; c) the relevancy criteria was too vague, subjective and difficult to define and as a result the evaluation process became unfeasible. Such cases include searching for games which are fun, creative, or lengthy.

3.5 Evaluation

As explained in the last section, each retrieval task was performed in various ways. Always starting with a more naive approach and then moving towards a more considerate method. To prove our hypothesis of the benefits of the more considerate approach over the naive, some metrics were gathered or calculated.

The results for the retrieval task (1) can be seen in Table 1 and Table 2. Both tables present the top 10 documents retrieved by *Solr*, a manual relevance judgement, and the value for the precision-at-k metric [3]. The former refers to the naive approach, while the last regards to the more considerate. The precision-at-k was based on manual relevance judgements, meaning each game was analyzed to confirm or deny that the game satisfied the information need. These precision values validate the initial hypothesis that boosting up the text from the reviews in the retrieval process would, in certain cases, improve the precision. In this case, with boosting, a perfect precision was achieved until the 4^{th} game (inclusively), while the default settings didn't surpass the 2^{nd}. Most notably, the average precision [3] with boosts is 0.9283 against 0.7454 without them.

Table 3 and Table 4 show the top 10 documents returned from the retrieval task (2). A similar evaluation approach to the retrieval task (1) was applied. Once again, one can clearly conclude boosting provides better results. In the boosted instance only 1 false positive was found on the 8^{th} position in relation to 5 false positives in the default setting. Furthermore, the average precision is 0.9765 while boosted and 0.7944 otherwise.

The information need derived from the retrieval task (3) allowed for further exploration of the retrieval tools. Table 5 presents the results for the unquoted approach of searching for *virtual reality*. For a starting point, a relatively high average precision of 0.8957 was achieved, with only two false positives. The following approaches improved the results significantly with both reaching perfect precision at 10 results. However, as seen in Table 6 and Table 7, the retrieved games only have one in common. A pattern can be seen in Table 7 which might shed some light on this difference. In its

	Game	R	P@k
1	Premium Bowling	Y	1.00
2	Contractors	Y	1.00
3	Tales of Escape	Y	1.00
4	Broken Reality	N	0.75
5	SUPERHOT VR	Y	0.80
6	ARAYA	Y	0.83
7	Escape First	Y	0.86
8	Deisim	Y	0.88
9	Axiom Verge	N	0.78
10	Vanishing Realms	Y	0.80

Table 5: Results from Retrieval Task (3) - Unquoted Expression

	Game
1	Premium Bowling
2	Tales of Escape
3	SUPERHOT VR
4	Escape First
5	ARAYA
6	Contractors
7	Deisim
8	Vanishing Realms
9	Knockout League - Arcade VR Boxing
10	Final Assault

Table 6: Results from Retrieval Task (3) - Quoted Expression

	Game
1	SUPERHOT VR
2	Sairento VR
3	Titanic VR
4	Kingspray Graffiti VR
5	Gun Club VR
6	Cloudlands : VR Minigolf
7	Bartender VR Simulator
8	Guided Meditation VR
9	Apollo 11 VR
10	Touring Karts

Table 7: Results from Retrieval Task (3) - Using synonyms

list, 9 out of 10 games have the term 'VR' in their game, implying the frequent use of the acronym instead of the complete expression, especially in the game title. It's important to note that the *title* was not boosted, the results come naturally as users mention the name of the game (including the 'VR' term) in their reviews. Lastly, we performed an analysis meant to compare the previous approaches to (close to) ground truth, not only for the first 10 results (which were manually judged on their relevance), but for all the results. For this, the *tags* field was used as it's a structured field with known possible values, one of which is 'VR' which means the game is playable in Virtual Reality. The results from this experiment showed that while the unquoted approach retrieved a total of 431 documents, only 56 had 'VR' as a tag. An improvement was made with the introduction of quotation marks as it returned 35 'VR' tagged games from a total of 44 retrieved. However, the same can't be said for the synonyms method, since the precision was lower (from 0.7955 to 0.6583), getting 79 documents correct from a total of 120 retrieved. Despite the setback in precision, the recall was increased as the collection features 82 VR games, based on their tags. With the previous data, the recall values can be calculated [3] for each approach: 0.6829, 0.4268, and 0.9634, respectively.

	Game	R	P@k
1	Football Manager 2020	Y	1.00
2	Counter-Strike: Global Offensive	N	0.50
3	Dota 2	N	0.33
4	Rocket League®	Y	0.50
5	eSports Legend	N	0.40
6	SMITE®	N	0,33
7	VR SUPER SPORTS	Y	0.43
8	Sports Bar VR	Y	0.50
9	Wild Animal Sports Day	Y	0.56
10	SC2VN - The eSports Visual Novel	N	0.50

Table 8: Results from Retrieval Task - Steam

	Game	R	P@k
1	Come on Baby!	Y	1.00
2	VRC PRO	Y	1.00
3	Steel Circus	Y	1.00
4	NBA 2K20	Y	1.00
5	Skater XL	Y	1.00
6	HoloBall	Y	1.00
7	Hot Shot Burn	Y	1.00
8	Steep	Y	1.00
9	Mutant Football League	Y	1.00
10	Axis Football 2019	Y	1.00

Table 9: Results from Retrieval Task - Developed system

3.5.1 *Comparing against Steam*

In this section, we present a brief comparison and evaluation between the developed search engine and the one supplied by Steam itself.

For this purpose, we'll ascertain 2 clear information needs expressed in single term queries:

(1) What games are sports-based?
(2) What games are focused on cars?

For evaluation, we'll consider the top 10 results provided by both systems. Because Steam contains not only games but other types of content as well, a type filter was set in order to select only games. Besides this, no other filters or modifications were applied in any case. Regarding the developed engine, and for all information needs, the fields considered for searching were *title*, *description*, *about* and *reviews.text* without any boosts applied.

The first information need (1) is concerned with searching for sports-based games, that is, games that simulate at least some aspects of a sporting activity such as football or basketball. Therefore, the query term chosen was *q = 'sports'*.

Table 8 and Table 9 present the results collected. The average precision measured was 0.5968 for the Steam's search engine and 1 for the developed engine. Even though it may appear jarring initially, upon further analysis, we can understand the discrepancy between the outputs lies in Steam's highly rated games related to *eSports*, which do not meet the information need criteria. Although these games are related to a sporting activity, the games themselves

	Game	R	P@k
1	Project CARS 2	Y	1.00
2	Grand Theft Auto V	N	0.50
3	Project CARS	Y	0.67
4	Rocket League®	Y	0.75
5	Disney•Pixar Cars	Y	0.80
6	Disney•Pixar Cars Mater-National Championship	Y	0.83
7	Project CARS - Pagani Edition	Y	0.86
8	RC Cars	Y	0.88
9	Disney•Pixar Cars 2: The Video Game	Y	0.89
10	Virtual SlotCars	Y	0.90

Table 10: Results from Retrieval Task - Steam

	Game	R	P@k
1	My Summer Car	Y	1.00
2	Car Mechanic Simulator 2015	Y	1.00
3	Automation - The Car Company Tycoon Game	Y	1.00
4	UNDER the SAND - a road trip game	Y	1.00
5	Drift86	Y	1.00
6	VRC PRO	Y	1.00
7	Production Line : Car factory simulation	Y	1.00
8	Need for Speed: Shift	Y	1.00
9	DiRT Rally	Y	1.00
10	rFactor 2	Y	1.00

Table 11: Results from Retrieval Task - Developed system

are not sports-based. From this analysis, we may conclude that Steam's relevance scoring system heavily factors in user popularity, seeing as *eSports* is currently a very popular trend in the gaming world which attracts millions of players [1].

The intent of the second information need (2) is the retrieval of games focused on cars. The search query chosen was *q = 'cars'*. We considered the games as relevant as long as cars play a major part of its content regardless of the type of interaction or goal (e.g. driving, racing or building).

The average precision measured was 0.8412 for Steam's search engine and 1 for the developed system. By analysing the results in Table 10 and Table 11 as well as the average precisions we can conclude that, although, the average precision is significantly distinct between both engines the number of relevant results was very similar. The observed difference was due to a false positive in the 2^{nd} retrieved game. Once again, the results support the previously established hypothesis of the popularity playing a relevant part in scoring as the title in question, *Grand Theft Auto V*, is extremely popular [16].

In the end, the mean average precision (MAP) measured [3] was 0.7190 for Steam's engine and 1 for the developed system.

To conclude, a truly comprehensive evaluation would require extensive further research including the analysis of a much larger number of information needs along with measurement of multiple metrics such as the aforementioned MAP and 11-point interpolated average precision for both systems. However, for the purposes of this evaluation, the results seem promising in demonstrating the developed search system can, at least for some information needs, perform as well if not better than Steam's.

4 Semantic Web

4.1 Tool selection

We've used *Protégé* [21] as an ontology editor since it's one of the most complete available [2]. Notwithstanding, this decision negatively impacted the ontology populating process as *Protégé*'s import plugin *Cellfie* [15] exhibits memory management issues when loading files or generating axioms, and unreasonable loading times, severely limiting the amount of the dataset that could be imported. For querying, we first started using *Protégé*'s *SPARQL Query* plugin [24]. However, it didn't display any information regarding the execution time, thus leading to the adoption of the query engine *ARQ* [10]. Excluding these constraints, *Protégé* proved to be a very versatile and intuitive tool for modelling the ontology. The ontology was then exported as an OWL (Web Ontology Language) [28] file — a Semantic Web language designed to represent knowledge supporting classes, data and object properties, restrictions, and individuals.

4.2 Ontology

4.2.1 Existing ontologies The first step was researching existing ontologies regarding video games, aiming for their adoption as interoperability of knowledge from several sources is a common goal of Semantic Web. An ontology for this domain was found: the *Video Game Ontology* [22]. From this ontology, we've integrated the classes *Game*, *Genre* and *Player*. Some properties were also reused, for example, a game's *releaseDate* from the aforementioned ontology and a player's *nick* from the *Friend Of A Friend* ontology. The most substantial disadvantage of integrating the *Video Game Ontology* is its lack of adoption by other developers. However, doing so expands the individual scopes of the ontologies, possibly bringing meaning to more information.

The main goal of the *Video Game Ontology*, according to the authors, is to "capture knowledge about events that happen in video games and information about players" [22]. In contrast, the goal of our ontology is to represent knowledge regarding a game, its general characteristics, and its user reviews. As such, even though there were overlapping concepts, leading to the integration of the *Video Game Ontology*, most of the conceptual data model had to be modelled from the ground up in our ontology.

4.2.2 Building the ontology As stated, we've built an ontology with the goal of representing knowledge regarding video games, based on the *Video Game Ontology*. Furthermore, we were able to expand the domain to include the game's user reviews and some general characteristics. Our ontology, shown in Appendix B.1, is composed of the following classes, object properties, and data properties:

4.2.2.1 Classes We've defined 10 classes in our ontology, which perfectly match the UML classes in the conceptual data model (Figure 6), apart from the *User* class which, as a result of integrating with the *Video Game Ontology*, had to be renamed to *Player*. Therefore, the classes present in our ontology are: *Game*, *Player*, *Review*, *Developer*, *Publisher*, *Genre*, *Feature*, *Tag*, *Franchise*, and *UserReviewRating*.

These classes are described in detail in Section 2.5. All classes are disjoint between each other, except for *Developer* and *Publisher*.

4.2.2.2 Object Properties We've defined 10 object properties, based on the relationships between the UML classes of the conceptual data model (Figure 6). Mapping simple associations required only one object property to be created. However, each association class (Review and UserReviewRating) required two object properties per relationship. For instance, in the case of a review submission by a user — association between User and Game, resulting in the association class Review —, two object properties were created: (1) *gameReview* – Linking games to its reviews; (2) *userReview* — Linking users to their submitted reviews. The same thought process was applied to the relationship that handles a user voting (with a rating) on a review, with the object properties being *userVote* and *reviewVote*.

In general terms, due to the nature of the object properties (their domains and ranges), they're inherently asymmetric — the domains and ranges are not interchangeable — and irreflexive — they don't relate with themselves. In addition, no disjoint object properties exist in our ontology.

In specific terms, some object properties are functional or inverse functional. For example, *gameReview* is inverse functional, since its domain is *Game* and its range *Review* and a review is always linked to one and only one game. A full list of the object properties and their restrictions can be found in Appendix B.2.

4.2.2.3 Data Properties We've defined 15 data properties, based on the attributes of the UML classes of the conceptual data model (Figure 6). For consistency reasons, none of the derived attributes were modelled into the ontology. A list of all the data properties can be consulted in Appendix B.3, since the names are readily comprehensible, apart from the *userName* attribute that was mapped into *foaf:nick* to accommodate the *Friend of a Friend* ontology. From this ontology, we've also used *foaf:name* for each class with a name.

In terms of data properties characteristics, none of them are functional — meaning each class individual only has at most one value per data property —, and no disjoint relations are applicable either.

4.3 Populating

After designing an ontology, an important aspect is to populate it to ensure its usefulness and correctness, which are further proved once queried. To populate the ontology, we've used *Cellfie* [15] — a *Protégé* plugin for creating or modifying ontologies from spreadsheets. As such, our data had to be converted from the JSON-Lines format to a tabular format, for which we've chosen CSV (Comma-Separated Values) [25], in order to form a spreadsheet. The conversion required multiple CSV files since our data involved multiple object types (games and reviews), and various one-to-many and many-to-many relations (leading to arrays). *Cellfie* also requires a set of rules that manipulate and map the data from the spreadsheet to the ontology, allowing for the creation of individuals and object/data property assertions.

The most substantial obstacle in populating the ontology was the tools themselves as the importing process could last several minutes for only a small fraction of the dataset. Due to these constraints, we populated the ontology with 11 video games, each with 5 reviews, for a total of 55 entries. Every class and property was populated with data from the dataset obtained in Section 2. Because of this, entities related to users voting on game reviews were the only part of the ontology that could not be populated with real data as it was not available in the dataset.

4.4 Querying

The next step towards demonstrating the usefulness of our populated ontology was querying it using common user needs of this domain. For the definition of the queries, we've looked back at the previously proposed retrieval tasks as a basis and elaborating them more towards the goal of ontology querying. In total, we've defined 9 user needs, translated them into SPARQL queries and retrieved its results from the populated ontology. The queries were executed on the *ARQ* query engine [10]. The queries must be prepended with the prefixes listed in Listing 1. Unlike information retrieval, these results are exact, thus not requiring the same evaluation metrics. However, a common data retrieval metric — execution time — was registered for each query and is available in Table 21. All queries were executed 5 times and their execution times were averaged. A small peculiarity to note is that the first execution always took substantially longer (10 to 100 times) than the rest, indicating a strong cache influence.

Some of the developed queries return all individuals of a certain class such as games (e.g. query for user need **(1)**) or reviews (e.g. query for user need **(6)**) and although this is manageable when working with a small amount of data, in a real-life system with thousands of individuals some pagination mechanism would be essential. This could be achieved through the combined use of **LIMIT** and **OFFSET** clauses which SPARQL provides in order to specify the number of retrieved individuals. Its usage is straightforward by appending the clauses accordingly with no other changes to the query required.

Listing 1: Query Prefixes

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-
    syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX cvgo: <http://www.semanticweb.org/dapi/
    ontologies/2019/11/ComputerVideoGameOntology#>
PREFIX vgo: <http://purl.org/net/
    VideoGameOntology#>
```

(1) What are the most recommended games by the users?

The objective of this user need is retrieving the games sorted by the percentage of users which recommended the game in their reviews. To achieve this, the resulting SPARQL query includes an outer and an inner query. The inner query selects and calculates the percentage itself by counting, for each game, the number of total reviews and the positive reviews. Then, in the outer query, the title of the game and the previously calculated percentage are selected, ready to be displayed.

This query makes use of **UNION** clause to mix all and positive-only reviews in a single table which can be combined by game via the **GROUP BY** clause, allowing the count of the total amount of reviews through the **COUNT** function.

Listing 2: Query for user need (1)

```
SELECT ?title ?percent_pos
WHERE {
    ?game cvgo:title ?title
    {
        SELECT ?game (COUNT(?pos_reviews) *
            100 / (COUNT(?reviews)) as ?
            percent_pos)
        WHERE{
            {
                ?reviews  cvgo:gameReview ?
                    game .
            }
            UNION
            {
                ?pos_reviews cvgo:gameReview
                    ?game ;
                        cvgo:recommended
                            true .
            }
        }
        GROUP BY ?game
    }
}
ORDER BY DESC(?percent_pos)
```

title	percent_pos
EuroTruckSimulator2	100.0
WarThunder	100.0
Rust	100.0
Cities:Skylines	100.0
TeamFortress2	100.0
MONSTERHUNTER:WORLD	100.0
TotalWar:WARHAMMERII	100.0
Remnant:FromtheAshes	100.0
Counter-Strike:GlobalOffensive	100.0
Indivisible	80.0
BlackDesertOnline	40.0

Table 12: Query results of user need (1)

Table 12 demonstrates the obtained results containing two columns, the title of the game and the corresponding percentage of positive reviews (referred to as *percent_pos*). All games are retrieved.

(2) What are the games with some term in the title?

In this need, the user searches for all games which include some particular search term or keyword in its title. The term "counter" was chosen as an example to showcase this need. Filtering capabilities based on regex of SPARQL were used to satisfy this user need.

Listing 3: Query for user need (2)

```
SELECT ?title
WHERE {
    ?game  cvgo:title ?title .

    FILTER regex(?title, "counter", "i")
}
```

title
Counter-Strike:GlobalOffensive

Table 13: Query results of user need (2)

The results displayed in Table 13 contain a single column corresponding to the title of the game. Only one game was retrieved, "Counter-Strike: GlobalOffensive" which effectively includes the term "counter" in its title.

(3) What are the most recent games?

The goal of this need is retrieving the games sorted by their release date, with the most recent at the top. The following query demonstrates the **ORDER BY** clause, particularly its usage to sort by date. The opposite, that is, retrieving the oldest games, could be achieved by simply changing the sort order from **DESC** to **ASC**.

Listing 4: Query for user need (3)

```
SELECT ?title ?date
WHERE {
    ?game cvgo:title ?title ;
          vgo:releaseDate ?date .
}
ORDER BY DESC(?date)
```

title	date
Indivisible	2019-10-08
Remnant:FromtheAshes	2019-08-19
MONSTERHUNTER:WORLD	2018-08-09
Rust	2018-02-08
TotalWar:WARHAMMERII	2017-09-28
BlackDesertOnline	2017-05-24
Cities:Skylines	2015-03-10
WarThunder	2013-08-15
EuroTruckSimulator2	2012-10-12
Counter-Strike:GlobalOffensive	2012-08-21
TeamFortress2	2007-10-10

Table 14: Query results of user need (3)

Table 14 showcases the obtained results which contain two columns, the title and release date of each game. All games are retrieved.

(4) What games belong to a genre, are tagged with a tag, and contain a feature?

This user need aims to simulate the filters provided by the search engine in Steam's website. It includes the ability to filter the games returned by either a single or set of genres, tags and features simultaneously. The **IN** clause is utilized in the **FILTER** to allow this flexibility. The criteria chosen in the following query is merely an example and exclusively selects games which include a "Multi-player" feature, are labelled with the "Competitive" tag and belong in the "Action" genre.

Listing 5: Query for user need (4)

```
SELECT ?title
WHERE {
    ?feature cvgo:isFeatureOfGame ?game .
    ?tag cvgo:isTagOfGame ?game .
    ?genre cvgo:isGenreOfGame ?game .
    ?game cvgo:title ?title .

    ?feature foaf:name ?featureName .
    ?tag foaf:name ?tagName .
    ?genre foaf:name ?genreName .

    FILTER(?featureName IN ("Multi-player"))
    FILTER(?tagName IN ("Competitive"))
    FILTER(?genreName IN ("Action"))
}
```

title
TeamFortress2
Counter-Strike:GlobalOffensive

Table 15: Query results of user need (4)

Table 15 demonstrates the results obtained. A single column with the title the game is displayed. Only two games were selected, which fit the previously mentioned criteria.

(5) What games are priced between two price points?

The intention behind this user need is finding what games are priced between a minimum and maximum value. In the following query, the price was set between 5 and 20 euros as an example.

Listing 6: Query for user need (5)

```
SELECT ?title ?price
WHERE {
    ?game cvgo:price ?price ;
          cvgo:title ?title .

    FILTER (5 <= ?price && ?price <= 20)
}
```

title	price
EuroTruckSimulator2	19.99
BlackDesertOnline	9.99

Table 16: Query results of user need (5)

The obtained results, showcased in Table 16, contain two columns for the title and price of the game. Only two games were selected priced at 19.99€ and 9.99€ respectively.

(6) Which game reviews are from users with the most hours played before writing the review?

The motivation behind this need is retrieving the reviews sorted by the number of hours the reviewer played the game before writing it, implying more accurate evaluations of the game.

Listing 7: Query for user need (6)

```
SELECT ?name ?title ?hours
WHERE {
    ?review cvgo:hours ?hours ;
            cvgo:gameReview ?game .
    ?player cvgo:userReview ?review ;
            foaf:nick ?name .
    ?game cvgo:title ?title .
}
ORDER BY DESC(?hours)
```

name	title	hours
Strix	BlackDesertOnline	12039
JoeyWheeler	TotalWar:WARHAMMERII	3054.4
Arthianne	BlackDesertOnline	2947.1
Locus	TeamFortress2	2390.8
ColCori	Counter-Strike:GlobalOffensive	2064.1
RagostatinDei	MONSTERHUNTER:WORLD	1666.3
Galaxian	Counter-Strike:GlobalOffensive	1230.6
shantiwater	MONSTERHUNTER:WORLD	941.9
elia$	WarThunder	887.2
standardheadache	TotalWar:WARHAMMERII	876.8

Table 17: Query results of user need (6)

Table 17 displays the results from the previous query. It includes three columns for the username of the reviewer and title of the game, which combined uniquely identify the review, and the number of hours played. In actuality, all reviews are retrieved but, for the sake of brevity, only the first 10 results are displayed in the results table.

(7) Which reviewers have the most hours played on average before writing the review?

In this user need, the goal is to retrieve the users who have written at least one review sorted by the average number of

hours played before writing the reviews. This would be useful to gauge, in general terms, the legitimacy of the reviews written by each user. The resulting query demonstrates the usage of the **AVG** function to calculate the average number of hours.

Listing 8: Query for user need (7)

```
SELECT ?name (AVG(?hours) AS ?average) (COUNT
    (?review) AS ?reviews)
WHERE {
    ?review cvgo:hours ?hours .
    ?player cvgo:userReview ?review ;
            foaf:nick ?name .
}
GROUP BY ?name
ORDER BY DESC(?average)
```

name	average	reviews
Strix	12039.0	1
JoeyWheeler	3054.4	1
Arthianne	2947.1	1
Locus	2390.8	1
ColCori	2064.1	1
RagostatinDei	1666.3	1
Galaxian	1230.6	1
shantiwater	941.9	1
elia$	887.2	1
standardheadache	876.8	1

Table 18: Query results of user need (7)

The results, displayed in Table 18, include three columns: the username, average hours and total number of reviews written for each reviewer. The results are similar to those found in the query for the user need **(6)**. This is due to the fact that not a single user has written more than one review, as confirmed by the review counter. This is to be expected when taking into account the small number of reviews loaded in the ontology and high amount of Steam users. However, when considering Steam's large amount of reviews, the results should be distinctly different between the two queries.

(8) What developers have developed the most games?

This user need aims to retrieve the developers sorted by the number of games they've developed in total. The query showcases the **GROUP_CONCAT** and **OPTIONAL** clauses. **GROUP_CONCAT** is used to concatenate the titles of the games developed in a single string and the **OPTIONAL** clause is used to ensure that even developers which have yet to develop a game are included in the results.

Listing 9: Query for user need (8)

```
SELECT ?name (GROUP_CONCAT (?title;separator
    = ", ") AS ?games) (COUNT(?game) AS ?
    total)
WHERE {
    ?developer rdf:type cvgo:Developer ;
               foaf:name ?name .
    OPTIONAL {
        ?developer cvgo:developsGame ?game .
        ?game cvgo:title ?title
    }
}
GROUP BY ?name
ORDER BY DESC(?total)
```

name	games	total
Valve	Counter-Strike:GlobalOffensive, TeamFortress2	2
PearlAbyss	BlackDesertOnline	1
ColossalOrderLtd.	Cities:Skylines	1
HiddenPathEntertainment	Counter-Strike:GlobalOffensive	1
SCSSoftware	EuroTruckSimulator2	1
LabZeroGames	Indivisible	1
CAPCOMCo.	MONSTERHUNTER:WORLD	1

Table 19: Query results of user need (8)

Table 19 shows the obtained results which include three columns: the developer names, the titles and the total amount of games by developer. For brevity reasons, only the first 7 entries are included in the results table although all developers are returned.

(9) What publishers have published games developed by third-party developers?

The objective of this user need is retrieving the publishers sorted by the number of games they've published which were developed by external developers, that is, other than the publisher themselves. This query introduces the **DISTINCT** clause which is used to ensure the developer names are only selected and counted once for both the resulting concatenated string of developer names and the total amount counter.

Listing 10: Query for user need (9)

```
SELECT (?pName AS ?publisherName) (
    GROUP_CONCAT (DISTINCT(?dName) ;
    separator = ", ") AS ?developers) (COUNT
    (DISTINCT(?dName)) AS ?total)
WHERE {
    ?publisher cvgo:publishesGame ?game ;
               foaf:name ?pName .
    ?game cvgo:title ?title .
    ?developer cvgo:developsGame ?game ;
               foaf:name ?dName .

    FILTER (?publisher != ?developer)
}
GROUP BY ?pName
ORDER BY DESC(?total)
```

publisherName	developers	total
SEGA	FeralInteractive(Linux), FeralInteractive(Mac), CREATIVEASSEMBLY	3
FeralInteractive(Mac)	FeralInteractive(Linux), CREATIVEASSEMBLY	2
FeralInteractive(Linux)	FeralInteractive(Mac), CREATIVEASSEMBLY	2
Ltd.	CAPCOMCo.	1
ParadoxInteractive	ColossalOrderLtd.	1
PerfectWorldEntertainment	GunfireGames	1
Valve	HiddenPathEntertainment	1
505Games	LabZeroGames	1
CAPCOMCo.	Ltd.	1
KakaoGamesEuropeB.V.	PearlAbyss	1

Table 20: Query results of user need (9)

Table 20 shows the obtained results which include three columns: the developer names, the titles and the total amount of games by developer. For brevity reasons, only the first 7 entries are included in the results table although all developers are returned.

Finally, in Table 21 we showcase the average execution times obtained for all of the previously presented queries.

Query	Time (s)
(1) What are the most recommended games by the users?	0,037
(2) What are the games with some term in the title?	0,025
(3) What are the most recent games?	0,032
(4) What games belong to a genre, are tagged with a tag and contain a feature?	0,055
(5) What games are priced between two price points?	0,034
(6) Which game reviews are from users with the most hours played before writing the review?	0,056
(7) Which reviewers have the most hours played on average before writing the review?	0,051
(8) What developers have developed the most games?	0,047
(9) What publishers have published games developed by third-party developers?	0,044

Table 21: Average execution times of queries

4.5 Discussion and future prospects

As it stands, the ontology would be useful for those who want to represent and retrieve data regarding computer video games, like video game digital distribution services such as Steam or Epic Games. If these type of entities were to adopt ontologies in common, linking their data with meaning (semantics) outside of their individual store, it would ease a lot of use cases. For example, it could help publishers publish their games into multiple services at once, or even help players make a purchase judgment on a game based on the reviews given by other players despite the store they bought it from.

A valuable feature of ontologies is their ability to be extended. In the future, the developed ontology could be expanded upon, covering more concepts in the realm of computer video games which extend beyond what is currently present in the dataset in an effort to represent more knowledge of the domain. Video games constitute a very broad and rich domain spanning to numerous concepts surrounding it from distinct areas, such as finance and news. Therefore, we see the opportunity for reusing and connecting with other existing ontologies. Examples of such concepts include identifying directors and producers along with the information about them such as biographic data or complementing the information regarding publishers and developers as companies.

By comparing the two workflows — information retrieval and ontologies — we conclude that they're both powerful tools that fulfil different needs. Information retrieval is used for unstructured data, with a focus on enabling full-text search. On the other hand, ontologies are used to represent and retrieve knowledge in a structured and domain-aware form, defining resources, literals and relations between them. Information retrieval answers user needs expressed in natural language with the use of keywords (e.g. requesting games about a specific theme), while ontologies answer user needs in relational language (e.g. requesting the most recommended games by users based on their reviews' ratings). Despite their differences, these processes can be combined to form "Semantically enhanced Information Retrieval" [9], enabling semantic search — searching by meanings — at the cost of performance due to additional overhead.

5 Conclusions

The objective of this work was threefold: (1) research and compile relevant data about video games into a single refined dataset; (2) index the dataset in a full-text search tool to perform information retrieval on and evaluate the results; (3) design, populate and query an ontology for this domain.

After prior research attempts for existing public datasets regarding the topic ended in unsatisfactory results, the team ultimately decided to generate one based on the data available on Steam. Choosing Steam as the source of data restricted the domain from all video games to exclusively computer video games. In order to extract the data, web crawling and scraping techniques were used and, subsequently, most of the effort was spent on the development of the scraper. One of the biggest challenges was fine-tuning the scraper in order to achieve a refined and coherent outcome. In the end, the dataset represents a comprehensive sample of Steam's video games offerings released on the PC platform, serving as a base for the rest of the work.

For the information retrieval process, *Solr* was selected as a search tool because of its documentation and community support, as well as its performance and simple interface. The indexing stage proved to be difficult in this project, as indexing nested information like games and their reviews is not trivial in *Solr*. However, we managed to keep this hierarchy while indexing, only to be faced with another unforeseen challenge – boosting fields from a parent while retrieving child documents, and vice versa. Consequently, this indexing method was replaced with the default flat approach, which flattens the children (user reviews) in the parent (games) and thus, even though both games and reviews were indexed, the documents became only the video games. The indexing process explored some of *Solr*'s analyzer capabilities, such as modifying the

schema to accommodate English language indexing improvements, like stemming and synonyms definition. After indexing the dataset, three information needs were defined, differing from the previously presented retrieval tasks as the goal was to explore the tool's full-text search capabilities. Each information need was translated into a number of queries and multiple information retrieval approaches which varied through field boosting, phrase searching, and synonym usage. For each information need, its approaches were then compared and evaluated based on metrics like precision-at-k, average precision, and recall. Some information need results were also compared to Steam's search system. The results revealed that the variations from the default retrieval approaches displayed improvements in the considered metrics for the selected information needs. As future work, we propose exploring other tools, such as *Elasticsearch*, and documenting the process of indexing, retrieval, evaluation, and comparing results and conclusions.

The process of developing an ontology started with its modelling, from the conceptual data model to an OWL file, integrating with other ontologies such as the *Video Game Ontology*. As an ontology editor, we've used *Protégé* which we concluded that, although a powerful and intuitive tool for modelling, it's very limited in the realm of populating ontologies because of its memory management issues and lack of support, and slightly limited in querying as it doesn't provide much functionality or statistics. For importing data, we've used *Protégé*'s *Cellfie* plugin which allowed us to import spreadsheet files, meaning the dataset had to be converted from JSON-lines to CSV. Additionally, a set of rules had to be defined for the plugin to map the data to each class or property. Querying was a straightforward process: we defined 9 user needs, implemented queries in SPARQL that satisfied them and executed them towards the populated ontology. The querying execution was initially performed inside *Protégé*, with the *SPARQL Query* plugin, but it was lacking statistics features, so we used *ARQ* as a query engine, allowing us to retrieve execution times. The results reveal the developed ontology has the potential of representing rich knowledge in the domain of video games, being able to answer common user needs and possibly be adopted by other entities like digital distribution services. As future work, we propose expanding the domain of the ontology and suggest further populating the ontology with more optimized tools, such as *GraphDB*.

References

[1] Abid Ahmed and Jonathan Tong. 2019. Esports: The next billion dollar industry. Retrieved November 23, 2019 from https://www.lexology.com/library/detail.aspx?g=d452955d-26e1-4574-bf62-3988f6c8ab9d

[2] Emhimed Alatrish. 2012. Comparison of Ontology Editors. *ERAF Journal on Computing* 4 (01 2012), 23–38.

[3] Jaime Arguello. 2013. Evaluation Metrics. Retrieved November 20, 2019 from https://ils.unc.edu/courses/2013_spring/inls509_001/lectures/10-EvaluationMetrics.pdf

[4] Entertainment Software Association. 2019. Entertainment Software Rating Board. Retrieved November 23, 2019 from https://www.esrb.org/

[5] James Batchelor. 2018. Global games market value rising to $134.9bn in 2018. Retrieved October 8, 2019 from https://www.gamesindustry.biz/articles/2018-12-18-global-games-market-value-rose-to-usd134-9bn-in-2018

[6] Valve Corporation. 2019. Steam. Retrieved October 8, 2019 from https://store.steampowered.com

[7] Valve Corporation. 2019. Steam Search. Retrieved October 10, 2019 from https://store.steampowered.com/search/?category1=998

[8] John Domingue, Dieter Fensel, and James A. Hendler. 2011. *Introduction to the Semantic Web Technologies.* Springer Berlin Heidelberg, Berlin, Heidelberg, 1–41. https://doi.org/10.1007/978-3-540-92913-0_1

[9] Miriam Fernández, Iván Cantador, Vanesa López, David Vallet, Pablo Castells, and Enrico Motta. 2011. Semantically enhanced Information Retrieval: An ontology-based approach. *Journal of Web Semantics* 9, 4 (2011), 434 – 452. https://doi.org/10.1016/j.websem.2010.11.003 JWS special issue on Semantic Search.

[10] Apache Software Foundation. 2019. ARQ - A SPARQL Processor for Jena. Retrieved January 1, 2020 from https://jena.apache.org/documentation/query/

[11] Apache Software Foundation. 2019. Solr. Retrieved November 23, 2019 from https://lucene.apache.org/solr/

[12] Python Software Foundation. 2019. Python Programming Language. Retrieved October 10, 2019 from https://www.python.org/

[13] Epic Games. 2018. Epic Games Store. Retrieved January 3, 2020 from https://epicgames.com

[14] Kris Graft. 2013. Valve gives players a louder voice with Steam Reviews. Retrieved October 10, 2019 from https://www.gamasutra.com/view/news/205643/Valve_gives_players_a_louder_voice_with_Steam_Reviews.php

[15] Josef Hardi, Martin O'Connor, Csongor Nyulas, and Tania Tudorache. 2018. Protégé plugin for creating OWL ontologies from spreadsheets. Retrieved January 1, 2020 from https://github.com/protegeproject/cellfie-plugin

[16] Erik Kain. 2019. Putting Grand Theft Auto V's 110 Million Copies Sold Into Context. Retrieved November 23, 2019 from https://www.forbes.com/sites/erikkain/2019/05/14/putting-grand-theft-auto-vs-110-million-copies-sold-into-context/

[17] Scrapinghub Ltd. 2019. A Fast and Powerful Scraping and Web Crawling Framework. Retrieved October 10, 2019 from https://scrapy.org/

[18] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval.* Cambridge University Press, New York, NY, USA.

[19] Xavier Morera. 2019. Elasticsearch vs. Solr - Choosing Your Open Source Search Engine. Retrieved November 23, 2019 from https://www.searchtechnologies.com/blog/solr-vs-elasticsearch-top-open-source-search

[20] Elastic NV. 2019. Elasticsearch. Retrieved November 23, 2019 from https://www.elastic.co/

[21] Stanford University School of Medicine. 2019. Protégé. Retrieved January 1, 2020 from https://protege.stanford.edu/

[22] Janne Parkkila, Filip Radulovic, María Poveda, and Daniel Garijo. 2014. The Video Game Ontology. Retrieved January 1, 2020 from http://vocab.linkeddata.es/vgo/

[23] Andre Perunicic. 2018. Steam Scraper. Retrieved October 8, 2019 from https://github.com/prncc/steam-scraper

[24] Timothy Redmond, Jennifer Vendetti, Matthew Horridge, and Ignazio Palmisano. 2018. Protege Desktop plug-in that provides support for writing and executing SPARQL queries. Retrieved January 1, 2020 from https://github.com/protegeproject/sparql-query-plugin

[25] Yakov Shafranovich. 2005. RFC 4180 - Common Format and MIME Type for Comma-Separated Values (CSV) Files. Retrieved January 1, 2020 from https://tools.ietf.org/html/rfc4180

[26] Andrew Smith. 2019. Why Epic Games Store-Exclusive Games Are Good For Developers. Retrieved January 3, 2020 from https://www.gamespot.com/articles/why-epic-games-store-exclusive-games-are-good-for-/1100-6472443/

[27] Dean Takahashi. 2018. Games market expected to hit $180.1 billion in revenues in 2021. Retrieved October 10, 2019 from https://venturebeat.com/2018/04/30/newzoo-global-games-expected-to-hit-180-1-billion-in-revenues-2021/

[28] W3C. 2012. OWL 2 Web Ontology Language. Retrieved January 4, 2020 from https://www.w3.org/TR/2012/REC-owl2-profiles-20121211/

[29] Ian Ward. 2019. Documentation for the JSON Lines text file format. Retrieved November 18, 2019 from http://jsonlines.org/

[30] Wikipedia. 2019. Steam. Retrieved October 8, 2019 from https://pt.wikipedia.org/wiki/Steam

A Scraped entities examples

A.1 Game

Listing 11: *Assassin's Creed Odyssey* game example


```
{
    "id": 812140,
    "title": "Assassin's Creed Odyssey",
    "genres": ["Action", "Adventure", "RPG"],
    "developer": ["Ubisoft Quebec", "Ubisoft Montreal", "Ubisoft Bucharest", "Ubisoft Singapore", "
        Ubisoft Montpellier", "Ubisoft Kiev", "Ubisoft Shanghai"],
    "publisher": ["Ubisoft"],
    "franchise": "Assassin's Creed",
    "release_date": "05-10-2018",
    "features": ["Single-player", "Steam Achievements", "Steam Trading Cards", "Captions available", "In-
        App Purchases", "Partial Controller Support"],
    "tags": ["Open World", "RPG", "Assassin", "Singleplayer", "Action", "Historical", "Adventure", "
        Stealth", "Story Rich", "Third Person", "Parkour", "Female Protagonist", "Sexual Content", "
        Choices Matter", "Violent", "Atmospheric", "Great Soundtrack", "Nudity", "Multiplayer", "Gore"],
    "description": "Choose your fate in Assassin's Creed Odyssey. From outcast to living legend, embark
        on an odyssey to uncover the secrets of your past and change the fate of Ancient Greece.",
    "price": 59.99,
    "sentiment": "Very Positive",
    "percent_positive": 84,
    "n_reviews": 28825,
    "early_access": false,
    "about": "(...) TRAVEL TO ANCIENT GREECE From lush vibrant forests to volcanic islands and bustling
        cities, start a journey of exploration and encounters in a war torn world shaped by gods and men.
        (...)"
}
```


A.2 Review

Listing 12: Review example submitted by *BGy*


```
{
    "product_id": 812140,
    "recommended": true,
    "date": "12-08-2019",
    "text": "Ubisoft really wasn't joking around when it called the latest Assassin's Creed game an
        Odyssey. (...)",
    "hours": 70.9,
    "username": "BGy",
    "products": 158,
    "found_helpful": 33,
    "found_funny": 2,
    "early_access": false
}
```

B Ontology

B.1 Visual graph

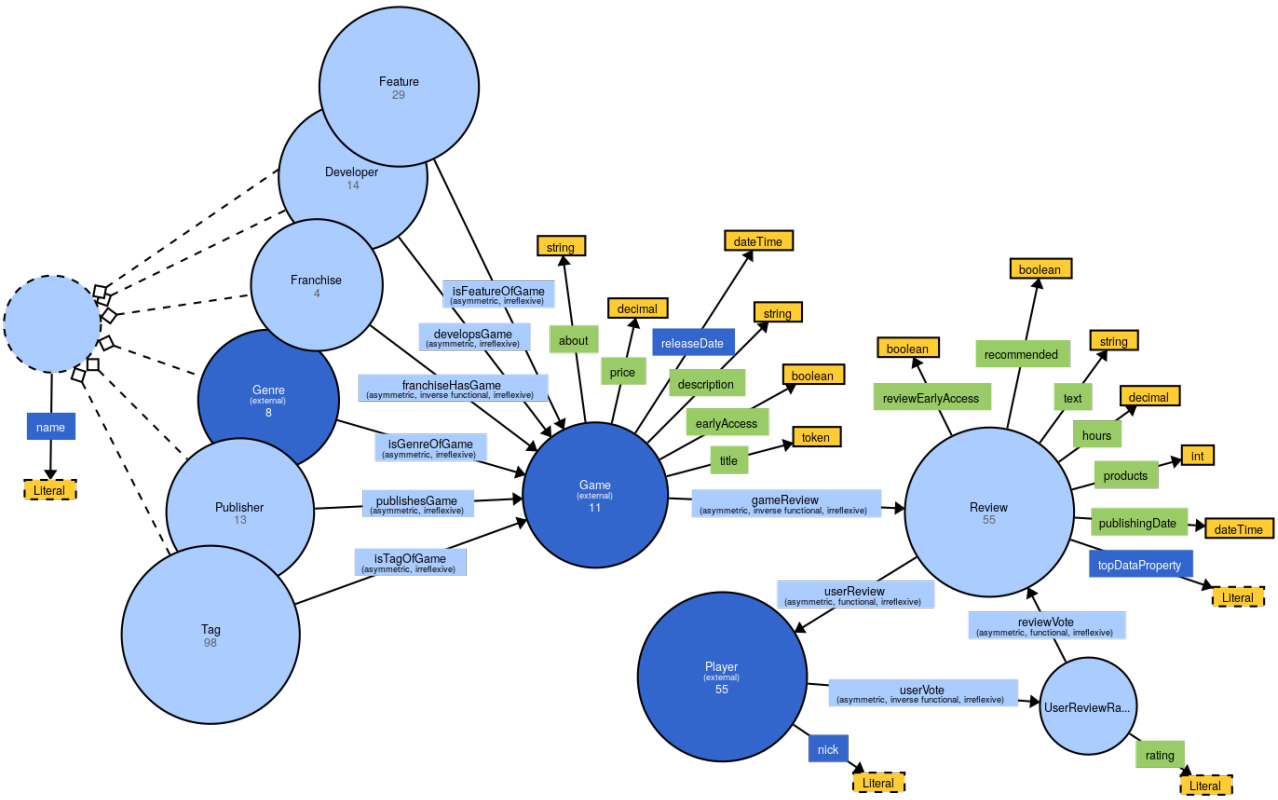

Figure 7: Ontology visual graph

B.2 Object Properties

Object Property	Domain	Range	Functional	Inverse Functional	Transitive	Symetric	Asymmetric	Reflexive	Irreflexive
developsGame	Developer	Game					✓		✓
publishesGame	Publisher	Game					✓		✓
franchiseHasGame	Franchise	Game		✓			✓		✓
isFeatureOfGame	Feature	Game					✓		✓
isGenreOfGame	Genre	Game					✓		✓
isTagOfGame	Tag	Game					✓		✓
gameReview	Game	Review		✓			✓		✓
userReview	Review	Player	✓				✓		✓
reviewVote	UserReviewRating	Review	✓				✓		✓
userVote	UserReviewRating	Player	✓				✓		✓

Table 22: Object Properties, their domain, range and characteristics

B.3 Data Properties

Data Property	Domain	Range
about	Game	xsd:string
description	Game	xsd:string
earlyAccess	Game	xsd:boolean
price	Game	xsd:decimal
releaseDate	Game	xsd:dateTime
title	Game	xsd:token
hours	Review	xsd:decimal
products	Review	xsd:int
publishingDate	Review	xsd:dateTime
recommended	Review	xsd:boolean
reviewEarlyAccess	Review	xsd:boolean
text	Review	xsd:string
foaf:nick	Player	rdfs:Literal
foaf:name	Developer, Publisher, Franchise, Feature, Genre, Tag	rdfs:Literal
rating	UserReviewRating	{"Funny", "Helpful"}

Table 23: Data Properties, their domain and range