Goodreads Books and Reviews

Bruno Sousa, Filipa Durão, Miguel Duarte and Rui Alves
MIEIC, FEUP
Porto, Portugal
{up201604145, up201606640, up201606298, up201606746}@fe.up.pt

***Abstract*—Given the increasing amount of data that is available online, being able to handle big amounts of information and process, index, and search it efficiently is evermore a focus for information systems nowadays. In this paper, the specific case of books and reviews from Goodreads is studied, alongside with additional data on authors extracted via Wikidata. After normalization and intersection, the dataset was ready for characterization, which showed that it was interesting to study it, given the extracted statistics that showed its heterogeneity while also confirming its validity due to having enough data distributed through most of its entries (with a total of 510 thousand unique documents). After indexing the documents using Solr, the evaluation of different system configurations showed that the IR system's quality highly depends on both the indexing operations (such as stop words removal and stemming) and the field weighting configuration (which must be balanced to ensure the retrieval of documents of all different types), achieving an 88% mean average precision in the optimal system configuration for the conceived test set. By modeling the domain as a web ontology, semantically linking the datasets' showed that information can become more understandable at the machine level, allowing the execution of more complex queries involving data aggregation.**

***Keywords*—Goodreads, Book, Author, Book Review, Dataset refining, Data retrieval, Data processing pipelines, Domain modeling, Domain search tasks, Data indexing, IR - Information Retrieval, Solr, Semantic Web, Ontology, Protégé, Cellfie, SPARQL**

I. Introduction

Goodreads is an American social cataloging website that features information about a multitude of books, together with their reviews from online users.

This project aims to create a proof of concept (PoC) of an IR system for the information available in the **Goodreads** website, with a focus on books and book reviews, and the authors that published those books. The construction of a web ontology that models these entities and how they relate to one another will also be explored.

In this paper, the process relative to characterizing, processing, indexing and querying datasets relative to books, their reviews, and their respective authors is described.

Firstly, Part II details the used datasets, as well as their collection and refinement processes. Secondly, Part III elaborates upon the processes used to index and search the information in the used (and refined) datasets. Then, Part IV details the process of conceiving, refining and populating a web ontology for the domain. Finally, Part V elicits some final remarks about this work.

II. Dataset Preparation

A. The Datasets

1) Books: The Books dataset was retrieved from **goodbooks-10k** [1]. This repository features a subset of the existing books on the website (the top 10,000 best-rated books on September 13th, 2017) and was built by scrapping the website's pages.

Regarding the dataset's refinement, firstly **OpenRefine** [2] was used to get a grasp of the data's nature. This dataset was in `CSV` format and contained exactly 10,000 entries. Initially, since each dataset entry featured information that wasn't relevant for the domain, a few attributes were filtered and all duplicate entries were removed. Then, all `null` and empty fields were normalized. Finally, all whitespaces were trimmed.

To thoroughly analyze the dataset, both **pandas** [3] and a set of `Python` scripts were used. It was concluded that 96% of all books in the dataset were written in the last two centuries. Of these, most were written in the past two decades (further information may be found in Figures 17, 18 and 19). Furthermore, the distribution of books by sagas was analyzed. It was concluded that about 76% of the books did not belong to any saga. Furthermore, 14% of the books belong to a saga that features a single book. The remaining 10% belong to a saga with 2 or more books (further information may be found in Figures 20 and 21).

2) Reviews: The Reviews dataset was retrieved from the **UCSD** website [4], where the reviews' information was divided into eight different book genres (such as Romance, Fantasy, ...). This repository features a subset of the existing book reviews in the **Goodreads** website (all reviews prior to 2018) and was built by scrapping their review pages.

Regarding the dataset's refinement, firstly **OpenRefine** was used to get a grasp of the data's nature. Then, **langdetect** [5] was used to understand the reviews text properties and language details. The original dataset featured about 15 million entries in `CSV` format. To reduce its size while maximizing its usefulness, the reviews were filtered to only match books existent in the books dataset. Then, reviews were filtered by date, so that only reviews from 2016 onwards remained. Finally, like in the other datasets, whitespaces were trimmed and the useful attributes were selected. This reduced the number of entries from 15 million to about 500 thousand. In the end, there was a 70% intersection between the books and reviews datasets

To thoroughly analyze the dataset, both **pandas** and a set of `Python` scripts were used. Moreover, to analyze the reviews' text content, the **langdetect** `Python` tool was used. The first conclusion is that the vast majority of the books have less than 50 reviews. The number of reviews through time is roughly linear, even though it is slightly decreasing over time (further information may be found in Figures 22 and 23). As for the languages in which the reviews were written, the English language is the most common (with about 91% of the reviews). About 8% are written in other languages, while the rest of them are unintelligible (further information may be found in Figures 24 and 25). Finally, as for the size of the reviews (using Twitter's maximum post size to reference a "small" review) it was concluded that about half of all reviews are short (about 47%), around 38% are medium-sized (241 to 999 characters) and only 15% are long (over 1000 characters, further information may be found in Figures 26 and 27).

3) Authors: The Authors dataset was built in two steps. Firstly, a list of the authors present in the Books datasets was extracted, using a set of `Python` scripts. Then, for each of those authors the **Wikidata** [6] information was fetched, using the **wptools** [7] `Python` package.

Regarding the dataset's refinement, after gathering all the information about the authors, and similarly to the previous datasets, all null and empty fields were normalized, all whitespaces were trimmed and the useful attributes were selected. In the end, the dataset features bout 3100 entries, and about 76% of the books in the Books dataset have information regarding their author.

To thoroughly analyze the dataset, both **pandas** and a set of `Python` scripts were used. It was concluded that the majority of the authors only wrote one book (about 59%), while 35,5% wrote between 2 and 9 books, and the remaining authors wrote 10 or more books, as visible in Figure 1 (further information may be found in Figure 29).

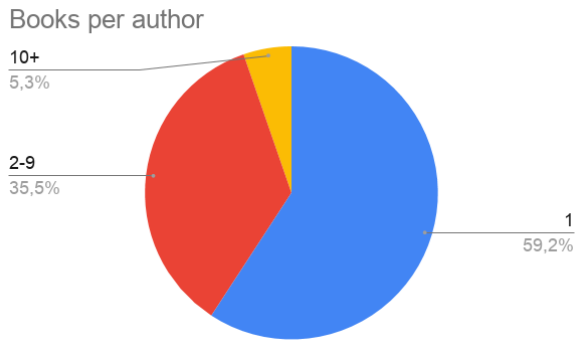


Figure 1: Books per Author.

B. Data Conceptual Model

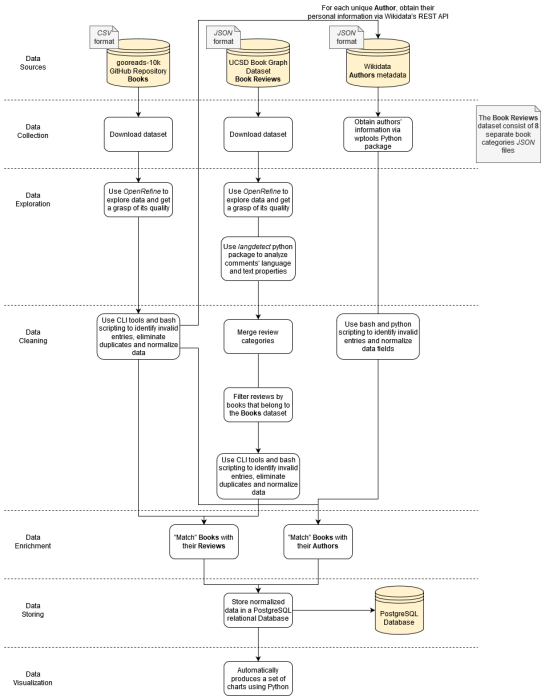


Figure 2: Data processing pipeline.

To treat the data, the pipeline outlined in Figure 2 was used. This pipeline can be separated in the following topics:

- Processing the **goodbooks-10k** dataset
- Processing the **UCSD** dataset
- Obtaining information about book authors' present in the **goodbooks-10k** dataset from **Wikidata**
- Merging books data with their reviews and with their authors

1) Books Dataset: The process began with the download of a `CSV` file from the **goodbooks-10k** repository, which contained the following book metadata:

- Book IDs
  - for **Goodreads**
  - internal to the dataset
- ISBN
- Authors
- Publication Year
- Title (with information on the book's Saga)
- Original Title (book title only)
- Language
- Rating information
  - Average Rating
  - Number of Total Ratings
  - Number of Ratings per Rating Value (1 - 5)
- Number of Text Reviews
- Image

From this metadata, the used attributes were: the **Goodreads** ID, original title, saga (obtained from the title), authors, ISBN, publication year, language and average rating.

After this step, **OpenRefine** was used for initial data visualization. Having acknowledged some problems in the data, some `CLI` tools as well as some specifically developed `bash` scripts were used to remove invalid entries, eliminate duplicates and normalize data.

2) Reviews Dataset: For the Reviews, multiple `JSON` files were used with information about the reviews, divided by book genre. For each entry, the following information was available:

- IDs in **Goodreads**
  - for the Reviewer
  - for the Book
  - for the Review
- Rating
- Review Text
- Creation and Update Date
- Number of votes and comments

From this data, the **Goodreads** Book ID, rating, review text and date were used.

Following this information collection and selection stage, **OpenRefine** was used for initial data visualization. In parallel, the **langdetect** `Python` package was used to identify the review language. The multiple `JSON` files were merged and the book reviews which were not in the books dataset were deleted. Once again, some `CLI` tools as well as some specifically developed `bash` scripts were used to remove invalid entries, eliminate duplicates and normalize data.

3) Authors Information: From the books dataset, the names of the authors of all the books were obtained. The **wptools** `Python` package was used to query **Wikidata** in order to obtain the authors' information. The following information about the authors was extracted:

- sex or gender
- date of birth
- country of citizenship
- place of birth

Afterwards, some scripts were used to identify and remove authors with no information or invalid names and to normalize certain fields that had lists of data instead of individual items.

4) Merging Datasets: Finally, all the datasets were merged using the **Goodreads** book ID to merge reviews and books. Besides that, the authors' names were used to merge books and authors. To analyze our data, the **pandas** `Python` package was used to process the data and generate the graphics presented on the 'The Datasets' section.

C. Domain Conceptual Model

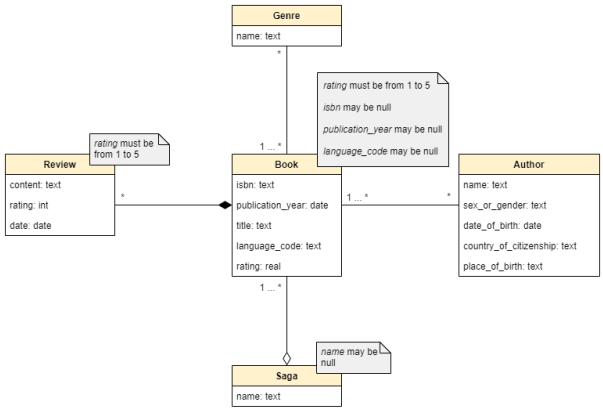


Figure 3: Domain Conceptual Diagram.

The domain of the project and how different entities relate with one another is modeled in Figure 3.

The domain's primary recovery unit is the Books. They represent a connection point amongst all other entities in the domain. The system's recovery units are Books, Reviews and Authors.

The remaining entities do not represent "direct" recovery units within the system. A Genre is a concept that gathers groups according to their topic and writing style and a Saga is a collection of Books that are interconnected.

D. Possible Search Tasks

In the scope of retrieving information from the stored data, the following search tasks were considered as Possible Search Tasks:

1) Search for books rated over R, filtered by genre G
2) Search for books that were co-authored by authors A_1 and A_2
3) Search for reviews of the most well-rated book in the saga S
4) Search for reviews between dates D_1 and D_2 of books that were authored by A
5) Search for medium-sized reviews in books written by authored A that are not from genre G
6) Search for authors that published over N books, filtered by their country of citizenship C
7) Search for authors who have written at least N books rated over R
8) Search for entities that have a section of text T in one of their fields (possibly giving different weights to different fields)

These Search Tasks were considered in order to belong to one (or more) of the following "categories":

- Filter by attributes in the entity
- Filter by relationships between entities
- Filter by attributes of other entities
- Filter by text searching in several attributes at once

III. Information Retrieval

Having completed the dataset preparation step, the following step consists of making its information accessible, so that it may satisfy users' information needs. Thus, this chapter features the developed work in the information retrieval system configuration, preparation, usage and refinement, discussing the taken approach, and the advantages and drawbacks of the implemented solution.

A. Tool Selection

The selected tool was **Solr** [8]. In terms of features, it is able to index data in several different formats, apply filters to different fields in order to make querying and indexing more efficient, and easily query data using several filters, varying weights, etc.

However, it also has a few limitations, as its documentation is highly dependant on the used version, which makes searching for further details bothersome. Additionally, this documentation faces a severe lack of practical examples which would greatly improve the usage experience.

It is worth mentioning that some *ad hoc* exploratory work was made using **Elasticsearch** [9]. However, the bulk of the work and investigation on tooling was made with **Solr**.

B. Collections and Documents

After completing the dataset preparation phase, the information was organized into three different datasets: Books, Authors and Reviews, which are thoroughly detailed in Section II-B.

To prepare the datasets to be imported into **Solr**, these were merged into a single `JSON` file, where each entry features a single document of any of the three types. Then, the collection was imported into **Solr** using the `post` tool:

```
post
  -c goodreads
  -format solr
  goodreads.json
```

This collection (with the 3 types of documents) is indexed in a single core. In order to do this, the created schema does not have any required attributes (all are optional) so that different entities simply have different non-null attributes. As such, this allows the retrieving of all of the dataset's entities using only one core.

C. The Indexing Process

The indexed fields in the three types of documents are listed in Tables I, II and III.

Table I: Book document fields.

Field	Type	Indexed
title	text_general	Y
id	string	N
isbn	string	Y
language_code	string	Y
publication_year	plongs	Y
book_rating	pfloat	Y
authors	string	Y

Table II: Author document fields.

Field	Type	Indexed
author_name	text_general	Y
sex_or_gender	string	Y
date_of_birth	string	Y
place_of_birth	text_general	Y
country_of_citizenship	plongs	Y

Table III: Review document fields.

Field	Type	Indexed
review_text	text_general	Y
id	string	N
date	string	Y
review_rating	pfloat	Y
book_id	string	N
book_name	string	Y

The initial task consisted of deciding which fields were to be searched upon. After experimentation, it was concluded that all fields should be indexed, except the identifier fields (such as the Book document type `id` field and Review document type `id` and `book_id` fields) - these fields are merely internal identifiers used by **Goodreads** and feature no semantic meaning.

Among the indexed fields, some were deemed worthy to be further processed. Although **Solr** features a set of default field types, they were considered to be either too specific or too broad for the required use case. Thus, a field type named *text_general* was created. When indexed, fields of this type are tokenized and processed according to a set of filters [10]:

- Stop words removal - using a list of common English connectors and prepositions that do not add discriminative power to user queries;
- Lower case conversion - converting all letters to the same case results in matching more results that may satisfy the user's needs;
- English possessive removal - removing trailing singular possessives from words;
- Stemming - Using Porter's stemming algorithm for English;
- Hyphenated words reconstructing - reconstructing hyphenated words that have been tokenized as two tokens because of a line break or other intervening whitespace.

An analysis on the results of using the aforementioned filters will be discussed in Sections III-E and III-F.

It is worth mentioning that, although it would be possible to use only a subset of these filters in different fields, this proved to slightly deteriorate the quality of the results. Moreover, the addition of these filters did not negatively impact the indexing process duration.

D. Retrieval Process

After completing the indexing of the documents, the next step was to decide how the documents should be retrieved. The retrieval process involved two major phases: selecting a query parser, and selecting and optimizing the parameters of the selected parser.

Among the many query parsers [11] offered by **Solr**, the explored ones were:

- The Standard query parser;
- The DisMax (Maximum Disjunction) query parser [12];
- The Extended DisMax query parser.

An *ad hoc* evaluation of the advantages and drawbacks of each option led to the choice of focusing research on the DisMax query parser since it is able to process simple queries and supports weighting each field of the indexed documents.

Regarding the available DisMax parameters, the ones used were:

- q - the query to search for in the documents
- qf - the list of document's fields, each including a weight to represent the importance of the field, to be searched for the query. The fields chosen for the query were $review\_text$, book's $title$, $author\_name$, and author's $place\_of\_birth$ as these were the fields that a common user in this domain would mostly search for.

After analysing the results with default qf weights, three different field weight configurations were conceived, as seen in Table IV.

Table IV: Field weight configurations.

Config.	review_text	title	author_name	place_of_birth
FW1	1.100	0.900	0.900	0.900
FW2	0.750	2.000	2.000	1.000
FW3	0.825	2.750	2.450	1.375

The *FW1* configuration aims to prioritize results from reviews documents. This configuration was implemented due to the fact that a large portion of the typical user queries is related to a certain topic they were interested to read about, and the text from other users' reviews provides most of the information about topics present in the book.

This configuration, however, didn't feature optimal results when users wanted to search for either a specific book or a specific author. Typically, reviews' text does not include the name of the book nor the authors' full name, so most of the retrieved results included these terms when the reviewer stated a resemblance with other books or authors (*e.g.* "the story is very similar to [`book_title`]", "the book is influenced by [`author_name`]'s work"), which resulted in the system returning a set of non-relevant results.

To mitigate this problem, two other field weight configurations were conceived. *FW2* aims to prioritize results of the Book or Author types (by applying a bigger weight to their `book_title` and `author_name` fields, respectively), leading to better results where the user's information needs should be fulfilled by documents of these types. To further enhance the results, the *FW3* configuration was conceived, where the same fields were adjusted based on the results obtained for the information needs of the developed test set. This led to an ideal balance among the different weighted fields, achieving a weighting configuration that aims to allow the retrieval of relevant documents of the three distinct types.

E. Evaluation Methodology

In order to evaluate the achieved information retrieval system in a systematic manner, the three different configurations showcased in Table V were conceived.

Table V: System configurations.

Configuration	Tokenization	Filtering	Stop Words removing
IR1	Y	N	N
IR2	Y	Y	N
IR3	Y	Y	Y

Firstly, configuration *IR1* aims to study the behavior of the system with only basic tokenization. Secondly, configuration *IR2* aims to understand the impact of applying the filters described in Section III-C (except for the stop words removal). Finally, configuration *IR3* aims to analyze the results of using a set of stop words.

To evaluate each system configuration, for each field weight configuration described in Section III-D, eight information needs were conceived. These were then expressed and queries and submitted to the system. For evaluation purposes, the first 20 results were taken into account, being deemed either relevant or non-relevant.

The results will, then, be evaluated according to the following metrics:

- Precision at `K` (from 1 to 20)
- Recall at `K` (from 1 to 20)
- Interpolated precision-recall (at 11 recall points, from 0% to 100%, with increments of 10%)
- Average precision (AvP)
- Mean average precision (MaP), for each system / field weight configuration pair (as shown in Equation 1, where `Q` is the total number of queries)

$$\frac{\sum_{q=1}^{Q} AvP(q)}{Q} \tag{1}$$

F. Results

The following three examples illustrate the nature of the information needs used to test the system.

Information Need (*IN1*): Understanding people's opinions on religion / faith-related books
Query (*Q1*): Religion OR Faith

The average precision (AvP) results for *IN1* obtained for each system / field weight configuration pair are visible in Table VI.

Table VI: Average precision results for information need *IN1*.

AvP	FW1	FW2	FW3
IR1	24%	65%	64%
IR2	85%	55%	57%
IR3	93%	64%	64%

For this information need, *FW1* showed as particularly low AvP value in *IR1*. However, the results significantly improved ib *IR2* (with the addition of filters). The quality

of the results peaked in *IR3* (with the addition of the stop words list). It is worth mentioning that, for this information need, the best results were obtained using the field weights configuration *FW1*, due to the fact that this configuration targets mostly reviews text content (which is the main goal of this information need).

Information Need (*IN2*): Finding historical books about the roman empire era
Query (*Q2*): (Rome or Roman) Empire

The average precision (AvP) results for *IN2* obtained for each system / field weight configuration pair are visible in Table VII.

Table VII: Average precision results for information need *IN2*.

AvP	**FW1**	**FW2**	**FW3**
IR1	87%	57%	65%
IR2	94%	75%	65%
IR3	99%	91%	85%

For this information need, an improvement pattern similar to *IN1* was observed, achieving significant improvements by adding analyzer filters and a stop words list. However, there was a significant improvement when using weight fields configuration *FW2* and *FW3*, due to the fact that these two configurations target mostly book titles and author names (which is the main goal of this information need).

Information Need (*IN3*): Finding historical books about the USA civil war and their authors
Query (*Q3*): (USA civil war) OR (Union AND Confederate)

The average precision (AvP) results for *IN3* obtained for each system / field weight configuration pair are visible in Table VIII.

Table VIII: Average precision results for information need *IN3*.

AvP	**FW1**	**FW2**	**FW3**
IR1	73%	37%	37%
IR2	88%	64%	43%
IR3	86%	58%	48%

The obtained results were quite similar to the ones obtained for *IN2*. However, the precision values were lower, since the collection features a fewer amount of documents on the American civil war domain.

Based on the aforementioned examples, it is possible to conclude that better results were achieved when using system configuration *IR3*, especially when using field weight configuration *FW1*, since this weight configuration targets documents of all types, with emphases on reviews (useful for most typical information needs).

The mean average precision (MaP), however, is a better metric to understand the quality of the system, since it takes into account the results obtained in all the queries of a given query set (as visible in Equation 1). The MaP results obtained for each system / field weight configuration pair using the conceived eight information needs are visible in Table IX.

Table IX: Mean average precision results for each *IR*/*FW* configuration pair.

MaP	**FW1**	**FW2**	**FW3**
IR1	60%	53%	53%
IR2	87%	65%	59%
IR3	88%	59%	56%

The obtained MaP results corroborate the conclusions obtained by analysing the *IN1*, *IN2* and *IN3* information needs - the system configuration that achieved the overall better results was *IR3*, using field weight configuration *FW1*.

All the results obtained from the analysis of the different configurations may be found in Annex VII.

G. Tool Evaluation

As **Solr** was the only tool used, there is no objective, empirical way to evaluate the tool and compare it with other information retrieval libraries or frameworks. However, it is possible to draw a few conclusions from the experience obtained when using it to implement this IR system:

- The documentation is very limited, making it hard to learn how to perform certain tasks since there are very few practical examples
- The configuration and customization of the tool was not straightforward nor user-friendly (especially during the indexing process)
- Nevertheless, once the learning curve is overcome, **Solr** offers many different options for all the needed IR tasks, including multiple ways to both index and query documents
- It allows the definition of complex queries and a very fast query response time

Overall, while **Solr** does have its disadvantages, it still allows the implementation of a good information retrieval system. The previously discussed results show that it is possible to achieve positive results using this tool (given that the indexing and querying processes are properly configured).

IV. Semantic Web

In the past few decades, the World Wide Web [13] has been adopted as the primary information source around the world. The semantic web [14] provides a way to develop a data store (built on a given vocabulary and a set of rules for data handling) on the Web. It aims to address and solve the lack of structuring of documents (that poses a problem for machine-interpretation) by semantically correlating documents and their entities.

This chapter features the developed work on conceiving an ontology that models the **Goodreads** domain using **Protégé** [15], with focus on the conceiving and configuration processes, as well as the populating and querying tasks.

A. Domain semantics

Although the conceptual domain has already been modeled and refined (as seen in Figure 3), it does not fully capture how entities relate to one another. Thus, a second version was developed, in order to facilitate building the ontology, showcased in Figure 4.

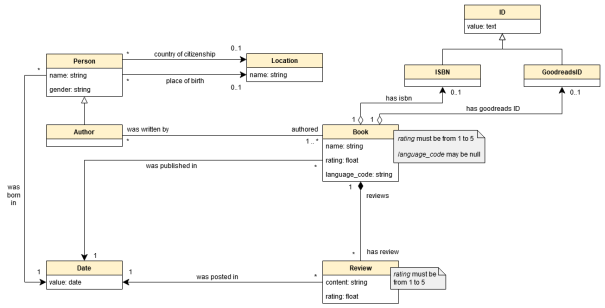


Figure 4: Domain Conceptual Diagram (iteration 2).

The added semantic relations provide a meaningful description on how these entities are related. Moreover, it is worth mentioning that a set of attributes were extracted to specific entity classes (such as `Date`, `Location` and `ID`), since this approach allows more flexibility in the ontology is to be integrated with other ones.

B. Existing Ontologies and Standards

There are multiple ontologies that model book stores and/or collections. However, due to the specificity of the domain, there is no ontology that includes the concepts of books, their authors and their reviews.

Thus, it was deemed that it would be more relevant to search for an ontology that was both more flexible and more generic than the aforementioned ontology-types.

Dublin Core [16] is a standard closely related to the domain of this project and consist of a lightweight RDFS [17] vocabulary for describing generic metadata. It is used mainly to describe digital resources (such as videos, images and web pages), as well as physical resources (such as books and CDs). It is widely used in libraries, universities and document-heavy fields such as law.

Of the fifteen featured metadata fields, the following were deemed relevant for the domain [18]:

- Creator - "An entity primarily responsible for making the resource."
- Date - "A point or period of time associated with an event in the lifecycle of the resource."
- Description - "An account of the resource."
- Language - "A language of the resource."
- Subject - "The topic of the resource."
- Title - "A name given to the resource."

It is worth mentioning that each Dublin Core element is optional and may be repeated. Although the aforementioned metadata fields are appropriate to describe both a book and a book review, they are inadequate to describe an author entity. These metadata fields have the prefix `dc`, and the following were selected and used when populating the ontology (further detailed in Section IV-D):

- Book
  - `dc:title`
  - `dc:date`
  - `dc:identifier`
  - `dc:language`
  - `dc:creator`
- Review
  - `dc:date`
  - `dc:identifier`

C. Building the ontology

1) Classes: The class schema in Figure 5 was defined to structure the ontology, representing the different entities of the domain.

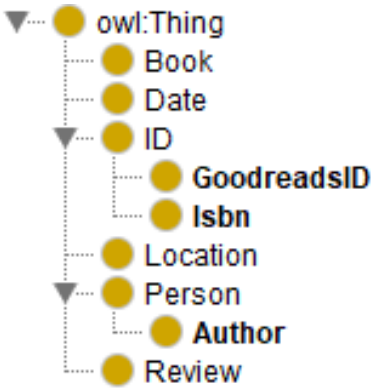


Figure 5: Ontology Classes.

Even though the *Person* class has only one subclass (the *Author* class), this approach improves the ontology's future extensibility by allowing the creation of other *Person*-type subclasses (for example, a *Reviewer* class).

It is worth mentioning that both Date and Location were assigned a class, which may be extended and integrated with specific ontologies in the future.

2) Object and Data Properties: To introduce associations between the multiple classes, the Object Properties presented in Figure 6 were created. These properties represent the semantics of how classes relate to one another.

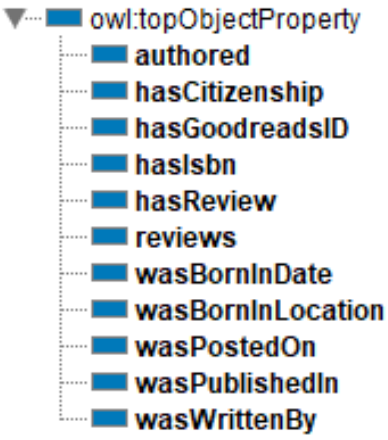


Figure 6: Ontology Object Properties.

The data properties presented in Figure 7 are used to characterize the attributes of the different entities in the domain, such as a person's name or gender.

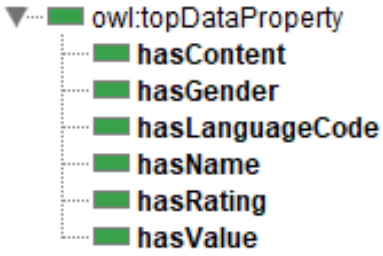


Figure 7: Ontology Data Properties.

3) Property Restrictions: All object properties are Asymmetric, meaning that these properties can be used to connect an individual to another, but those individuals cannot be connected the other way around by the same property. Moreover, all object properties are Irreflexive, which means that no individuals can be connected to themselves via any of the properties.

The `wasWrittenBy` and the `authored` object properties are inverse of each other. This means that, if an individual `B` was written by `A`, then `A` authored `B`. The same can be said about the `reviews` and `hasReview` object properties: If an individual `R` reviews `B`, then `B` has a review `R`.

Finally, all data properties are functional.

D. Populating the Ontology

The dataset preparation step resulted, as detailed in Part II, in three `JSON` datasets (`books.json`, `authors.json` and `reviews.json`). However, **Protégé** does not feature a way of populating an ontology from files in this format. Thus, a set of parsing scripts were implemented to produce a version of the datasets in `CSV` format.

Protégé includes the **Cellfie** plugin [19], which allows the creation of axioms [20] to populate the ontology from an **Excel** workbook.

Thus, after iteratively updating the format of the produced `CSV` datasets, they were merged into a single `XLSX` file, containing one sheet for each of the three entity datasets, as shown in Figure 8.

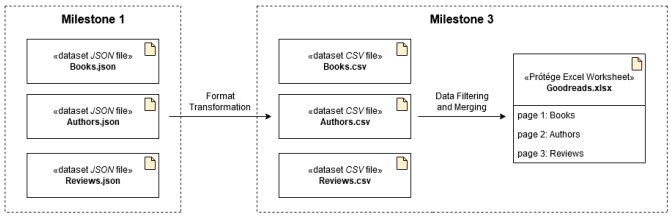


Figure 8: Milestone 3 dataset transformations.

Then, a set of transformation rules were defined to produce the axioms and required transformations that generate the individuals, their type constraints, and other facts (such as the data and object properties of each individual). The set of rules is shown in Figure 9.

Sheet Name	Start Column	End Column	Start Row	End Row	Rule	Comment
books	E	E	2	+	Individual: @E* Types: Date Facts: hasValue @E*(xsd:string)	Date (Book)
books	A	A	2	+	Individual: @A* Facts: authored @C*(xsd:string)	Author (authored)
authors	D	D	2	+	Individual: @D* Types: Location Facts: hasValue @D*(xsd:string)	Location (Author, citizenship location
books	B	B	2	+	Individual: @B* Types: GoodreadsID Facts: hasValue @B*(xsd:string)	GoodreadsID
authors	E	E	2	+	Individual: @E* Types: Location Facts: hasValue @E*(xsd:string)	Location (Author, birth location)
authors	A	A	2	+	Individual: @A* Types: Author Facts: hasName @A*(xsd:string) Facts: hasGender @B*(xsd:string) Facts: wasBornInDate @C* Facts: hasCitizenship @D* Facts: wasBornInLocation @E*	Author
reviews	A	A	2	+	Individual: @A* Types: Review Facts: hasRating @D*(xsd:float) Facts: hasContent @E*(xsd:string) Facts: reviews @C* Facts: wasPostedOn @F* Annotations: dc:date @F* Annotations: dc:identifier @A*	Review
books	C	C	2	+	Individual: @C* Types: Book Facts: hasName @C*(xsd:string) Facts: hasLanguageCode @F*(xsd:string) Facts: hasRating @G*(xsd:float) Facts: wasWrittenBy @A* Facts: hasGoodreadsID @B* Facts: hasIsbn @D* Facts: wasPublishedIn @E* Annotations: dc:creator @A* Annotations: dc:title @C* Annotations: dc:identifier @D* Annotations: dc:date @E* Annotations: dc:language @F*	Book
reviews	F	F	2	+	Individual: @F* Types: Date Facts: hasValue @F*(xsd:string)	Date (Review)
reviews	C	C	2	+	Individual: @C* Facts: hasReview @A*	Book (hasReview)
authors	C	C	2	+	Individual: @C* Types: Date Facts: hasValue @C*(xsd:string)	Date (Author)
books	D	D	2	+	Individual: @D* Types: Isbn Facts: hasValue @D*(xsd:string)	Isbn

Figure 9: Cellfie transformation rules.

The application of the listed rules resulted in the creation of about 1,200 axioms (consisting of creating individuals, type assertions, format assertions, and others), which resulted in populating the ontology with about 200 individuals. Figure 10 displays an example of the axioms generated when creating the "The Hunger Games" book-type individual.

```
Individual: TheHungerGames
TheHungerGames Type Book
TheHungerGames hasGoodreadsID 2767052
TheHungerGames hasIsbn 439023483
TheHungerGames hasLanguageCode “eng” ^^xsd:string
TheHungerGames hasName “The Hunger Games” ^^xsd:string
TheHungerGames wasWrittenBy SuzanneCollins
SuzanneCollins authored TheHungerGames
TheHungerGames wasPublishedIn 2008
TheHungerGames hasRating 4.34f
```

Individual
Object property
Data property
Data property assertion

Figure 10: Generated axioms for a book-type individual.

It is worth mentioning that, since **Protégé** is quite resource-heavy, the axiom generation step may take up to 10 seconds (even though the number of generated axioms is in the order of a few hundred).

E. Implemented Queries and Results

Before implementing the queries *per se*, a list of querying goals was conceived. Of the list of candidates, the following ones were selected to be expressed as **SPARQL** [21] queries:

1) Obtain a list of books and their number of reviews (a metric that shows how discussed they are in the community), in descending order (most popular first)
2) Obtain authors alongside with the books they wrote
3) Find the top 3 most reviewed authors of all time
4) Find the top 5 best-rated books of all time
5) For each book, find its best and worse reviews (rating-wise)
6) Find the best-rated authors of all time (based on the average rating of the books they authored)
7) Find the 5 books that were most recently reviewed (trending books)
8) Find all the reviews on books written by American female authors

The following three examples illustrate a few of the developed **SPARQL** queries.

Goal (*OWL_G3*): Find the top 3 most reviewed authors of all time
Query (*OWL_Q3*): Figure 11
Result (*OWL_R3*): Figure 12

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.semanticweb.org/rui/ontologies/2020/11/goodreads#>

SELECT ?author_name (count(?review) as ?n_book_reviews)
WHERE {
?author a :Author .
?author :hasName ?author_name .
?author :authored ?book .
?book :hasReview ?review.
}
GROUP BY (?author_name)
ORDER BY DESC(?n_book_reviews)
LIMIT 3
```

Figure 11: *OWL_Q3* SPARQL Query.

author_name	n_book_reviews
"Suzanne Collins"^^<http://www.w3.org/2001/XMLSchema#string>	"22"^^<http://www.w3.org/2001/XMLSchema#integer>
"Dan Brown"^^<http://www.w3.org/2001/XMLSchema#string>	"8"^^<http://www.w3.org/2001/XMLSchema#integer>
"Jane Austen"^^<http://www.w3.org/2001/XMLSchema#string>	"8"^^<http://www.w3.org/2001/XMLSchema#integer>

Figure 12: *OWL_R3* SPARQL Query result.

Goal (*OWL_G5*): For each book, find its best and worse review (rating-wise)
Query (*OWL_Q5*): Figure 13
Result (*OWL_R5*): Figure 14

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.semanticweb.org/rui/ontologies/2020/11/goodreads#>

SELECT ?book_name ?rating_diff ?min_review_rating ?max_review_rating {
    BIND(?max_review_rating - ?min_review_rating AS ?rating_diff) .
    {
    SELECT ?book_name (min(?review_rating) AS ?min_review_rating) (max(?review_rating) AS
           ?max_review_rating)
    WHERE {
       ?book a :Book .
       ?book :hasName ?book_name .
       ?book :hasReview ?review .
       ?review :hasRating ?review_rating .
    }
    GROUP BY ?book_name ?review_interval
    ORDER BY ?min_review_rating ?max_review_rating
    }
}
```

Figure 13: *OWL_Q5* SPARQL Query.

book_name	max_review_rating	min_review_rating	rating_diff
Twilight	4.0	0.0	4.0
Angels & Demons	5.0	0.0	5.0
Catching Fire	5.0	1.0	4.0
Divergent	5.0	2.0	3.0
Mockingjay	5.0	2.0	3.0
Het Achterhuis ...	5.0	3.0	2.0
The Hunger Games	5.0	3.0	2.0
Pride and Prejudice	5.0	3.0	2.0
Man som hatar kvinnor	5.0	3.0	2.0
To Kill a Mockingbird	5.0	4.0	1.0

Figure 14: *OWL_R5* SPARQL Query result (Datatypes omitted).

Goal (*OWL_G6*): Find the best-rated authors of all time (based on the average rating of the books they authored)
Query (*OWL_Q6*): Figure 15
Result (*OWL_R6*): Figure 16

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.semanticweb.org/rui/ontologies/2020/11/goodreads#>

SELECT ?author_name (AVG(?book_rating) as ?author_rating)
WHERE {
?author a :Author .
?author :hasName ?author_name .
?author :authored ?book .
?book :hasRating ?book_rating .
}
GROUP BY ?author_name
ORDER BY DESC(?author_rating)
```

Figure 15: *OWL_Q6* SPARQL Query.

author_name	author_rating
"Harper Lee"^^<http://www.w3.org/2001/XMLSchema#string>	"4.25"^^<http://www.w3.org/2001/XMLSchema#float>
"Jane Austen"^^<http://www.w3.org/2001/XMLSchema#string>	"4.24"^^<http://www.w3.org/2001/XMLSchema#float>
"Veronica Roth"^^<http://www.w3.org/2001/XMLSchema#string>	"4.24"^^<http://www.w3.org/2001/XMLSchema#float>
"Suzanne Collins"^^<http://www.w3.org/2001/XMLSchema#string>	"4.22"^^<http://www.w3.org/2001/XMLSchema#float>
"Stieg Larsson"^^<http://www.w3.org/2001/XMLSchema#string>	"4.11"^^<http://www.w3.org/2001/XMLSchema#float>
"Anne Frank"^^<http://www.w3.org/2001/XMLSchema#string>	"4.10"^^<http://www.w3.org/2001/XMLSchema#float>
"Dan Brown"^^<http://www.w3.org/2001/XMLSchema#string>	"3.85"^^<http://www.w3.org/2001/XMLSchema#float>
"Stephenie Meyer"^^<http://www.w3.org/2001/XMLSchema#string>	"3.57"^^<http://www.w3.org/2001/XMLSchema#float>

Figure 16: *OWL_R6* SPARQL Query result.

A full list of the developed **SPARQL** queries may be found in Annex VIII.

F. Tool Evaluation

Protégé allows quick and thorough creation of an ontology out of the box, without the need for installing other pieces of software or plugins. Its user interface is also quite easy to navigate.

However, from a programmatic standpoint, it lacks ways of populating an ontology from different file formats, as well as mechanisms that make this process more automatic - The only mechanism provided by **Protégé** is the **Cellfie** plugin, that lacks documentation and support, while at the same time being very resource-heavy.

Moreover, the **Protégé** documentation is often outdated (most documentation refers to older versions than the supported one) and lacks practical examples and tutorials.

Thus, the easiest task to complete was, even though requiring a number of refinement iterations, the ontology schema definition (that is, the classes, object and data properties definition). The hardest task was the ontology populating, since the lack of documentation resulted in a slow *ad hoc* trial and error approach to define the **Cellfie** transformation rules.

Nevertheless, this tool is still the best option to quickly build a web ontology since it has a small learning curve and works well out of the box.

G. Ontology Applications

This ontology was built with the intent to represent the **Goodreads** domain (together with information about the books' authors). Thus, it could be applied to the **Goodreads** website, together with authors information from **WikiData**.

However, with minor refactoring, most of the classes, object and data properties could be applied to similar domains, such as (a) music CDs, their reviews and musicians, (b) movies, their reviews and screenwriters, (c) plays, their reviews, and play-writers, among others.

It is worth mentioning that the ontology was conceived with the intent for future expansibility, that is, it allows the addition of new entities, classes and properties without the need for major changes in the current version of the schema. For example, it would be straightforward to add a new *ReviewAuthor* class, as a subclass of the *Person* class, further extending the domain of the ontology to a new concept. Thus, it is important to maintain a degree of flexibility when conceiving an ontology in order to enable future expansion and allow possible integrations.

V. Conclusions

In this paper, the chosen datasets and how they were processed is described, as well as the system conceptualization designed for the project. Then, the steps taken to configure the IR system itself is detailed, with emphasis on the indexing process and the querying results.

Regarding the datasets preparation phase, the datasets were downloaded and processed. The refinements included whitespace trimming, duplicate entries removal and missing/null fields normalization. Then, the datasets' intersection percentage was analyzed, obtaining results above 70%. Furthermore, the information contained in the datasets was studied, and from that, charts were traced to better visualize the data. From there, it was concluded that both the books and the reviews have a favorable distribution over time and a balanced number of reviews per book. In this report, the Domain and Data Conceptual Models may also be found. The Domain Conceptual model describes how the different entities relate to each other. The Data Conceptual Model describes the pipeline used to extract and treat the datasets.

Regarding the IR system configuration phase, firstly a *ad-hoc* comparison between the **Solr** and **Elasticsearch** technologies was made. Although both technologies offered similar features, the developed work was focused on the former. Then, the datasets were merged and imported to **Solr**, indexing each imported document. In this process, a set of fields was subjected to a list of additional operations, which consisted of the removal of stop words, stemming, capitalization normalization, among others. Regarding the retrieval process, a set of field weighting configurations were studied, where each proved to achieve better results in specific information need cases. Regarding the system's evaluation, a set of three system configurations were conceived. The configuration that achieved the best results was the one which applied a set of operations to the document's fields of interest, while using a stop words list, with a mean average precision of 88%.

Finally, regarding the conceived semantic web ontology, the original conceptual model was adapted to better fit the needs to build the ontology. Research made on domain-related ontologies showed that, given the specificity of the domain, no ontologies exist that reflect exactly the context of this work. However, a set of the **Dublin Core** standard metadata fields was applicable, although not fully adopted. Afterward, the ontology itself was built using **Protégé** allowed to quickly and easily create the domain's classes, object and data properties, although this schema was reiterated multiple times. After migrating the dataset to an **Excel** workbook `XLSX` format, the **Cellfie** plugin was used to populate the ontology by defining a set of transformation rules that generated axioms to create individuals, properties and assert constraints. Finally, a set of querying goals were formulated as **SPARQL** queries, which due to the meaningful naming of the schema's classes and

object properties, allowed to expressively describe and satisfy these user needs.

It is worth mentioning that Information Retrieval (IR) and Semantic Web (SW) serve different purposes: While IR is intended to suffice textual-bases user information needs on a given domain, SW aims to make information more accessible and understandable to computers by semantically linking entities, thus allowing higher complexity queries that may involve data aggregation or joining.

As future work, the further expansion of the developed web ontology would be of interest by adding the concepts of review authors, book publishers, and book stores. It would also benefit from integrating other web ontologies, such as W3C Geospatial Ontologies [22] (to describe locations) and ontologies used to accurately describe people.

References

[1] Z. Zając, "Goodbooks 10k," 13th Sep 2017, version 1. Data retrieved from goodbooks-10k GitHub repository, https://github.com/zygmuntz/goodbooks-10k.

[2] D. Huynh, "Open Refine," 23rd Oct 2020. [Online]. Available: https://openrefine.org/

[3] W. McKinney, "pandas - Python Data Analysis Library," 22nd Oct 2020. [Online]. Available: https://pandas.pydata.org/

[4] M. Wan, "UCSD Book Graph - Reviews," 2017, data retrieved from https://sites.google.com/eng.ucsd.edu/ucsdbookgraph/reviews.

[5] N. Shuyo, "langdetect," 24th Oct 2020. [Online]. Available: https://pypi.org/project/langdetect/

[6] W. Foundation, "Wikidata," 26th Oct 2020. [Online]. Available: https://www.wikidata.org/wiki/Wikidata:Main_Page

[7] S. Siznax, "wptools," 25th Oct 2020. [Online]. Available: https://pypi.org/project/wptools/

[8] Apache, "Apache Solr," 14th Nov 2020. [Online]. Available: https://lucene.apache.org/solr/

[9] Elastic, "Elasticsearch," 15th Nov 2020. [Online]. Available: https://www.elastic.co/what-is/elasticsearch

[10] Apache, "Apache Solr Filter Descriptions," 18th Nov 2020. [Online]. Available: https://lucene.apache.org/solr/guide/6_6/filter-descriptions.html

[11] A. S. Foundation, "Query Syntax and Parsing," 15th Nov 2020. [Online]. Available: https://lucene.apache.org/solr/guide/6_6/query-syntax-and-parsing.html

[12] Apache, "Apache Solr DixMax query parser," 21th Nov 2020. [Online]. Available: https://lucene.apache.org/solr/guide/8_6/the-dismax-query-parser.html

[13] w3, "World Wide Web History," 10th Dec 2020. [Online]. Available: https://www.w3.org/History.html

[14] w3, "Semantic Web," 11th Dec 2020. [Online]. Available: https://www.w3.org/standards/semanticweb/

[15] Stanford, "Protégé," 20th Dec 2020. [Online]. Available: https://protege.stanford.edu/

[16] DublinCore, "Dublin Core," 7th Dec 2020. [Online]. Available: https://dublincore.org/

[17] w3, "W3 - RDFs," 19th Dec 2020. [Online]. Available: https://www.w3.org/2001/sw/wiki/RDFS

[18] Wikipedia, "Dublin Core," 7th Dec 2020. [Online]. Available: https://en.wikipedia.org/wiki/Dublin_Core

[19] J. Hardi, "cellfie plugin," 22nd Dec 2020. [Online]. Available: https://github.com/protegeproject/cellfie-plugin

[20] Stanford, "Protégé Axioms," 23rd Dec 2020. [Online]. Available: https://protegewiki.stanford.edu/wiki/Protege4AxiomAnnotations

[21] W3C, "SPARQL Query Language for RDF," 30th Dec 2020. [Online]. Available: https://www.w3.org/TR/rdf-sparql-query/

[22] W3C, "W3C Geospatial Ontologies," 29th Dec 2020. [Online]. Available: https://www.w3.org/2005/Incubator/geo/XGR-geo-ont-20071023/

VI. Annex A - Datasets Analysis

A. Books Dataset Analysis

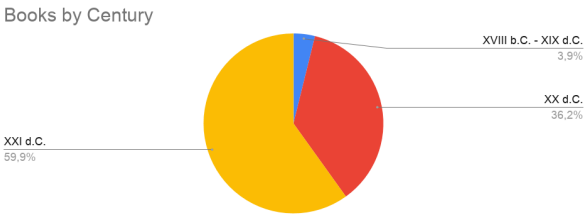


Figure 17: Books by Century.

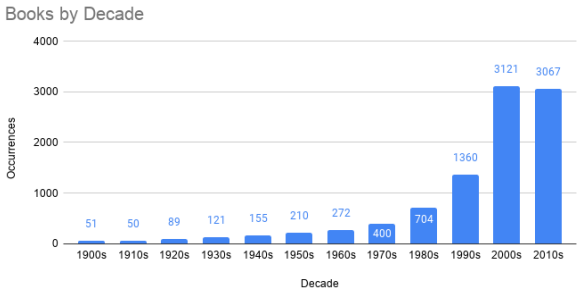


Figure 18: Books by Decade.

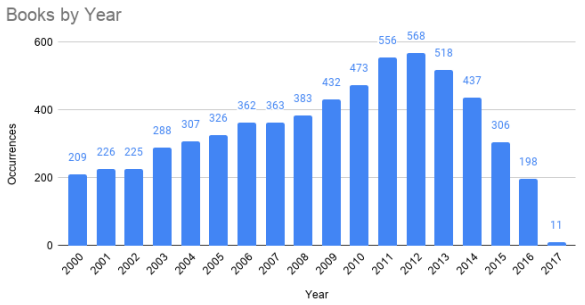


Figure 19: Books by Year.

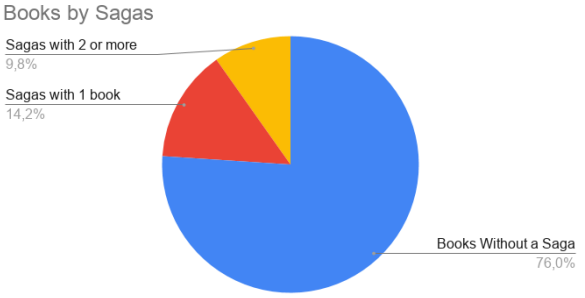


Figure 20: Books by Saga.

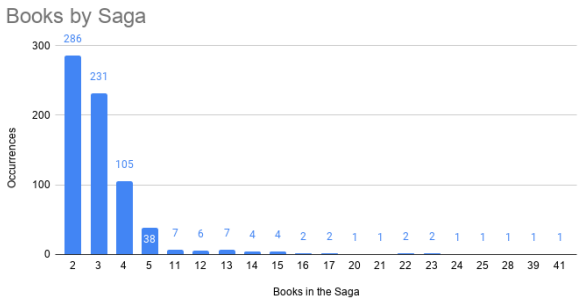


Figure 21: Books by Saga.

B. *Reviews Dataset Analysis*

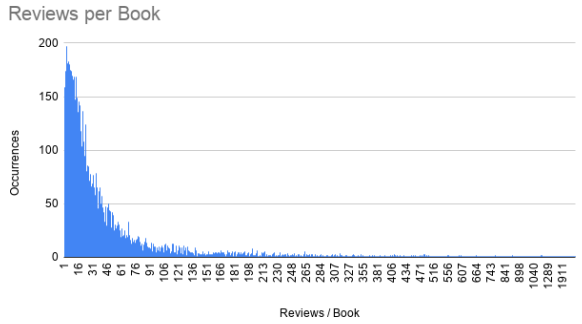


Figure 22: Reviews per Book.

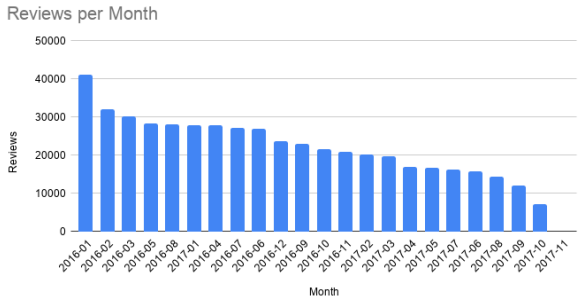


Figure 23: Reviews per Month.

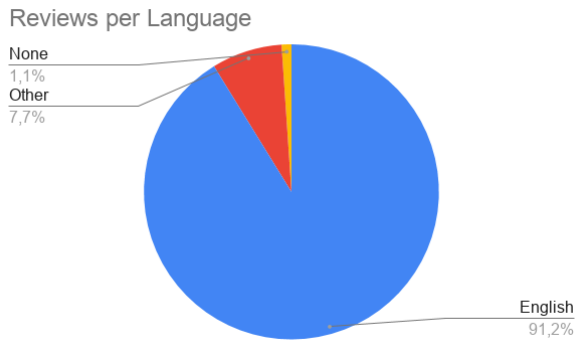


Figure 24: Reviews per Language.

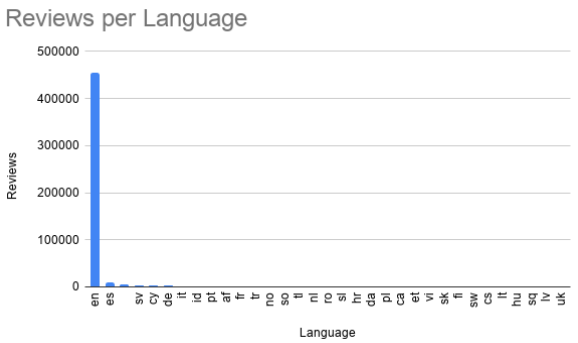


Figure 25: Reviews per Language.

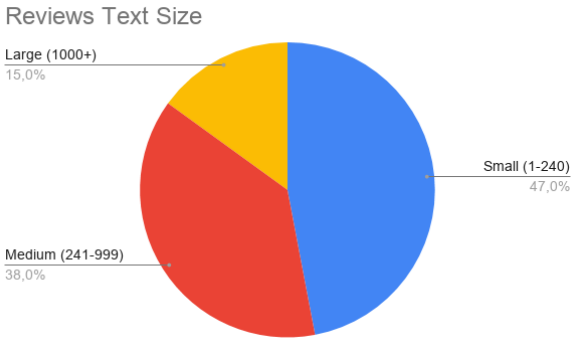


Figure 26: Reviews' size distribution.

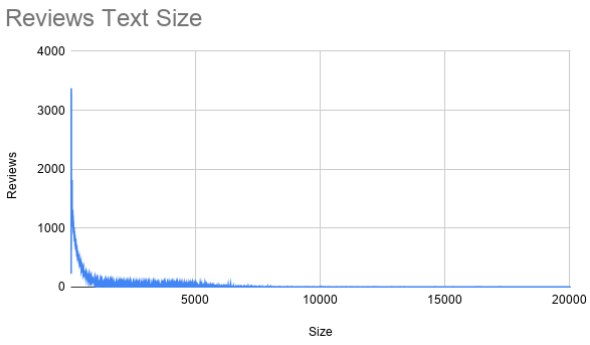


Figure 27: Reviews' size distribution.

C. Authors Dataset Analysis

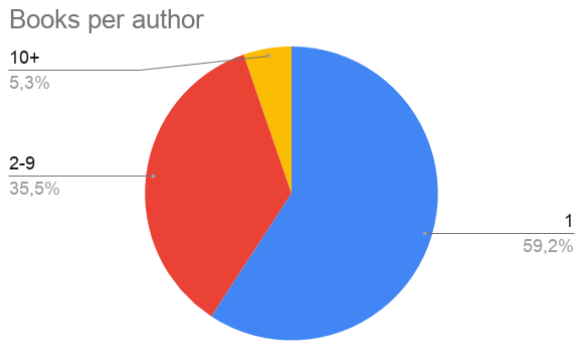


Figure 28: Books per Author.

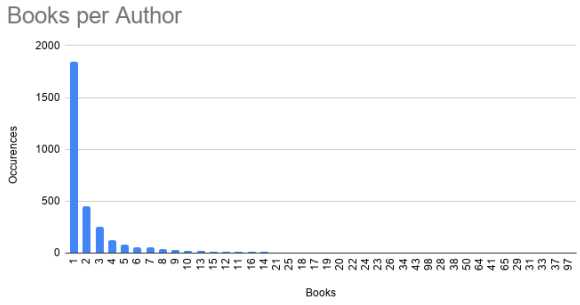


Figure 29: Books per Author.

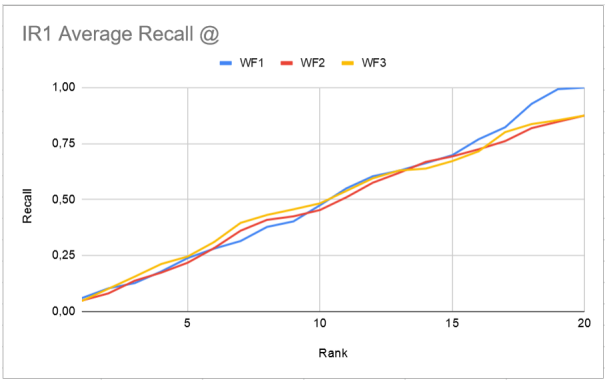


Figure 31: *IR1* average recall at K.

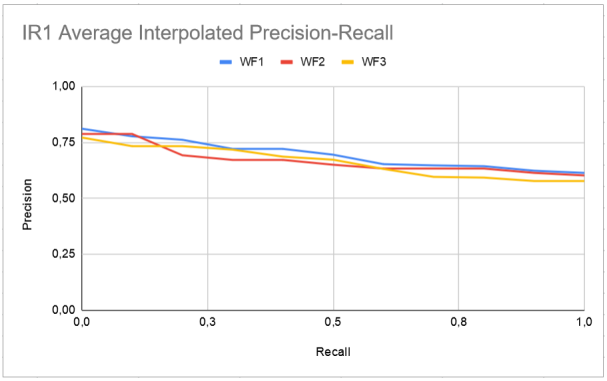


Figure 32: *IR1* average interpolated precision-recall.

VII. Annex B - IR System Configurations Analysis

A. System configuration IR1

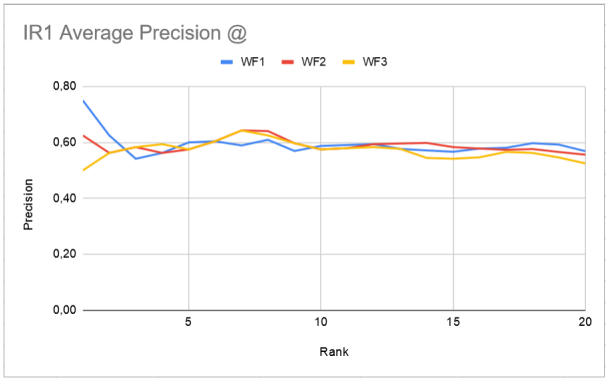


Figure 30: *IR1* average precision at K.

B. System configuration IR2

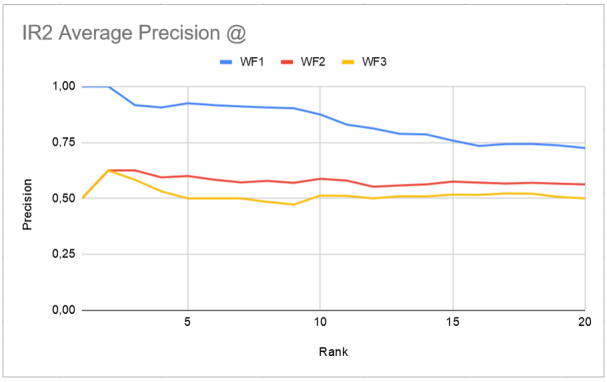


Figure 33: *IR2* average precision at K.

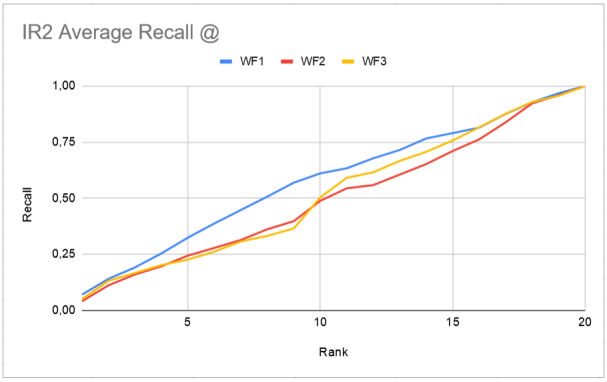


Figure 34: *IR2* average recall at K.

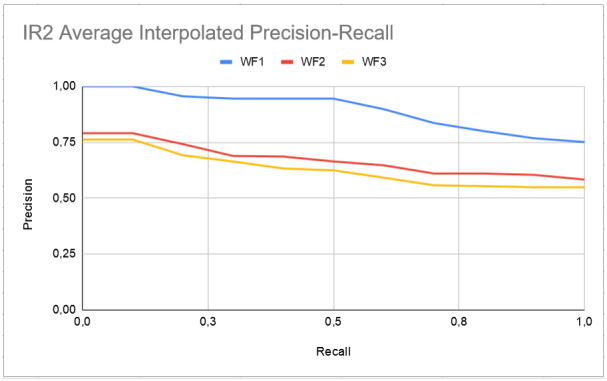


Figure 35: *IR2* average interpolated precision-recall.

C. System configuration IR3

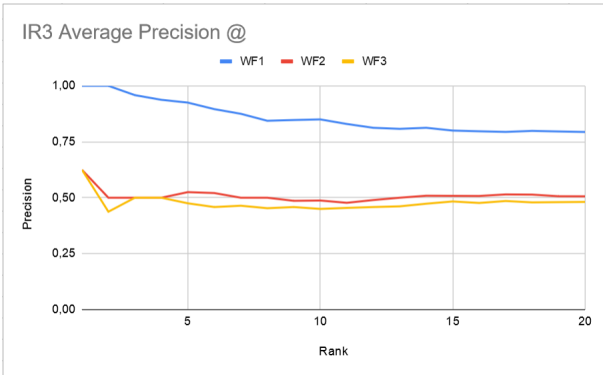


Figure 36: *IR3* average precision at K.

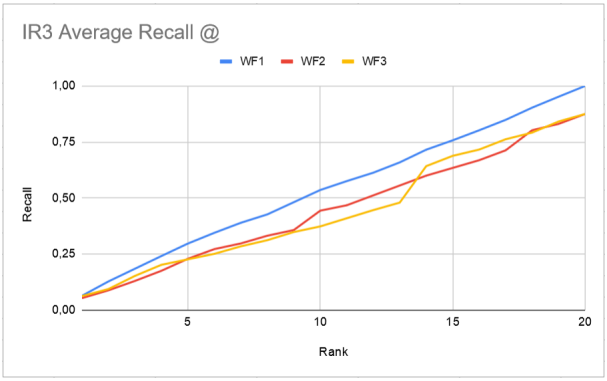


Figure 37: *IR3* average recall at K.

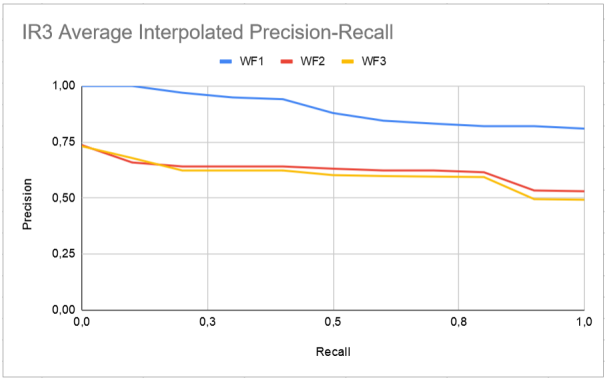
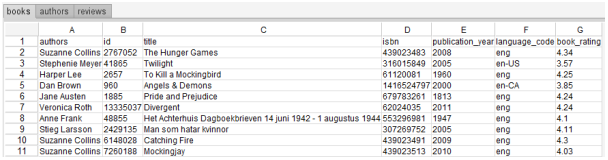


Figure 38: *IR3* average interpolated precision-recall.

VIII. Annex C - Web Ontology

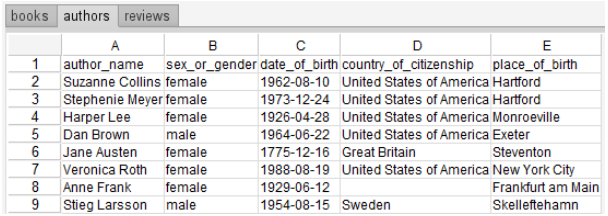
A. *Cellfie Input Excel workbook*



books | authors | reviews

	A	B	C	D	E	F	G
1	authors	id	title	isbn	publication_year	language_code	book_rating
2	Suzanne Collins	2767052	The Hunger Games	439023483	2008	eng	4.34
3	Stephenie Meyer	41865	Twilight	316015849	2005	en-US	3.57
4	Harper Lee	2657	To Kill a Mockingbird	61120081	1960	eng	4.25
5	Dan Brown	960	Angels & Demons	1416524797	2000	en-CA	3.85
6	Jane Austen	1885	Pride and Prejudice	679783261	1813	eng	4.24
7	Veronica Roth	13335037	Divergent	62024035	2011	eng	4.24
8	Anne Frank	48855	Het Achterhuis Dagboekbrieven 14 juni 1942 - 1 augustus 1944	553296981	1947	eng	4.1
9	Stieg Larsson	2429135	Man som hatar kvinnor	307269752	2005	eng	4.11
10	Suzanne Collins	6148028	Catching Fire	439023491	2009	eng	4.3
11	Suzanne Collins	7260188	Mockingjay	439023513	2010	eng	4.03

Figure 39: **Excel** workbook - books page.



books | authors | reviews

	A	B	C	D	E
1	author_name	sex_or_gender	date_of_birth	country_of_citizenship	place_of_birth
2	Suzanne Collins	female	1962-08-10	United States of America	Hartford
3	Stephenie Meyer	female	1973-12-24	United States of America	Hartford
4	Harper Lee	female	1926-04-28	United States of America	Monroeville
5	Dan Brown	male	1964-06-22	United States of America	Exeter
6	Jane Austen	female	1775-12-16	Great Britain	Steventon
7	Veronica Roth	female	1988-08-19	United States of America	New York City
8	Anne Frank	female	1929-06-12		Frankfurt am Main
9	Stieg Larsson	male	1954-08-15	Sweden	Skelleftehamn

Figure 40: **Excel** workbook - authors page.

books | authors | reviews

	A	B	C	D
1	id	book_id	book_name	review_rating review_text
2	461149	2767052	The Hunger Games	3 Easy read and entertaining, but socially, there were so many flaws with the world-building. But the
3	462390	2767052	The Hunger Games	3 ** spoiler alert ** OMG. Everyone loves Kathisis and she has no idea why. I have no idea either.
4	462603	2767052	The Hunger Games	4 I had avoided this one for a while– a blockbuster dystopian YA novel that appeared to be a rehashi
5	462744	2767052	The Hunger Games	5 When I was in the hospital they had a bookshelf full of books. Instead of trying a bunch of books to
6	462749	2767052	The Hunger Games	5 Listened to this one on audio book. Highly recommend doing that. The scholastic production reall
7	463375	2767052	The Hunger Games	4 The Hunger Games, by Suzanne Collins 4.5 out of 5 stars SPOILERS! But really I doubt you have
8	463862	2767052	The Hunger Games	5 Loved this! Was like nothing I'd ever read before, I couldn't put it down. I really connected with chan
9	464001	2767052	The Hunger Games	3 The first book of the Hunger Games trilogy
10	57010	41865	Twilight	4 My review on thinning for a while is different from the review that I would leave right after reading th
11	57753	41865	Twilight	3 The first three books in the four-part vampire saga. An engrossing read, but I hated Edward from th
12	58477	41865	Twilight	3 ** spoiler alert ** I finished reading Twilight today. Took me about a day to read as it is such a quic
13	58755	41865	Twilight	4 Always better than the movie version, it has a lot more details that I like.
14	60253	41865	Twilight	3 Lets be real, this is garbage – but it's entertaining garbage. It's like eating cotton candy, it's terrible

Figure 41: **Excel** workbook - reviews page.

B. SPARQL Queries

SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.semanticweb.org/rui/ontologies/2020/11/goodreads#>

SELECT ?book_name (count(?review) as ?n_reviews)
WHERE {
?book a :Book .
?book :hasName ?book_name.
?book :hasReview ?review .
}
GROUP BY ?book_name
HAVING (?n_reviews > 1)
ORDER BY DESC(?n_reviews)
```

book_name	n_reviews
"Divergent"^^<http://www.w3.org/2001/XMLSchema#string>	"8"^^<http://www.w3.org/2001/XMLSchema#integer>
"Angels & Demons "^^<http://www.w3.org/2001/XMLSchema#string>	"8"^^<http://www.w3.org/2001/XMLSchema#integer>
"Twilight"^^<http://www.w3.org/2001/XMLSchema#string>	"8"^^<http://www.w3.org/2001/XMLSchema#integer>
"Mockingjay"^^<http://www.w3.org/2001/XMLSchema#string>	"8"^^<http://www.w3.org/2001/XMLSchema#integer>
"The Hunger Games"^^<http://www.w3.org/2001/XMLSchema#string>	"8"^^<http://www.w3.org/2001/XMLSchema#integer>
"Pride and Prejudice"^^<http://www.w3.org/2001/XMLSchema#string>	"8"^^<http://www.w3.org/2001/XMLSchema#integer>
"Het Achterhuis Dagboekbrieven 14 juni 1942 - 1 augustus 1944"^^<http://www.w3.org/2001/X	"7"^^<http://www.w3.org/2001/XMLSchema#integer>
"To Kill a Mockingbird"^^<http://www.w3.org/2001/XMLSchema#string>	"7"^^<http://www.w3.org/2001/XMLSchema#integer>
"Catching Fire"^^<http://www.w3.org/2001/XMLSchema#string>	"6"^^<http://www.w3.org/2001/XMLSchema#integer>
"Man som hatar kvinnor"^^<http://www.w3.org/2001/XMLSchema#string>	"6"^^<http://www.w3.org/2001/XMLSchema#integer>

Figure 42: *OWL_Q1* SPARQL Query and result.

SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.semanticweb.org/rui/ontologies/2020/11/goodreads#>

SELECT ?author_name ?book_name
WHERE {
?author a :Author .
?author :hasName ?author_name .
?author :authored ?book .
?book :hasName ?book_name .
}
ORDER BY ?author_name
```

author_name	book_name
"Anne Frank"^^<http://www.w3.org/2001/XMLSchema#string>	"Het Achterhuis Dagboekbrieven 14 juni 1942 - 1 augustus 1944"^^<http://www.w3.org/2001/X
"Dan Brown"^^<http://www.w3.org/2001/XMLSchema#string>	"Angels & Demons "^^<http://www.w3.org/2001/XMLSchema#string>
"Harper Lee"^^<http://www.w3.org/2001/XMLSchema#string>	"To Kill a Mockingbird"^^<http://www.w3.org/2001/XMLSchema#string>
"Jane Austen"^^<http://www.w3.org/2001/XMLSchema#string>	"Pride and Prejudice"^^<http://www.w3.org/2001/XMLSchema#string>
"Stephenie Meyer"^^<http://www.w3.org/2001/XMLSchema#string>	"Twilight"^^<http://www.w3.org/2001/XMLSchema#string>
"Stieg Larsson"^^<http://www.w3.org/2001/XMLSchema#string>	"Man som hatar kvinnor"^^<http://www.w3.org/2001/XMLSchema#string>
"Suzanne Collins"^^<http://www.w3.org/2001/XMLSchema#string>	"The Hunger Games"^^<http://www.w3.org/2001/XMLSchema#string>
"Suzanne Collins"^^<http://www.w3.org/2001/XMLSchema#string>	"Catching Fire"^^<http://www.w3.org/2001/XMLSchema#string>
"Suzanne Collins"^^<http://www.w3.org/2001/XMLSchema#string>	"Mockingjay"^^<http://www.w3.org/2001/XMLSchema#string>
"Veronica Roth"^^<http://www.w3.org/2001/XMLSchema#string>	"Divergent"^^<http://www.w3.org/2001/XMLSchema#string>

Figure 43: *OWL_Q2* SPARQL Query and result.

SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.semanticweb.org/rui/ontologies/2020/11/goodreads#>

SELECT ?author_name (count(?review) as ?n_book_reviews)
WHERE {
?author a :Author .
?author :hasName ?author_name .
?author :authored ?book .
?book :hasReview ?review.
}
GROUP BY (?author_name)
ORDER BY DESC(?n_book_reviews)
LIMIT 3
```

author_name	n_book_reviews
"Suzanne Collins"^^<http://www.w3.org/2001/XMLSchema#string>	"22"^^<http://www.w3.org/2001/XMLSchema#integer>
"Dan Brown"^^<http://www.w3.org/2001/XMLSchema#string>	"8"^^<http://www.w3.org/2001/XMLSchema#integer>
"Jane Austen"^^<http://www.w3.org/2001/XMLSchema#string>	"8"^^<http://www.w3.org/2001/XMLSchema#integer>

Figure 44: *OWL_Q3* SPARQL Query and result.

SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.semanticweb.org/rui/ontologies/2020/11/goodreads#>

SELECT ?book_name ?book_rating
WHERE {
?book a :Book .
?book :hasName ?book_name .
?book :hasRating ?book_rating .
}
ORDER BY DESC(?book_rating)
LIMIT 5
```

book_name	book_rating
"The Hunger Games"^^<http://www.w3.org/2001/XMLSchema#string>	"4.34"^^<http://www.w3.org/2001/XMLSchema#float>
"Catching Fire"^^<http://www.w3.org/2001/XMLSchema#string>	"4.3"^^<http://www.w3.org/2001/XMLSchema#float>
"To Kill a Mockingbird"^^<http://www.w3.org/2001/XMLSchema#string>	"4.25"^^<http://www.w3.org/2001/XMLSchema#float>
"Divergent"^^<http://www.w3.org/2001/XMLSchema#string>	"4.24"^^<http://www.w3.org/2001/XMLSchema#float>
"Pride and Prejudice"^^<http://www.w3.org/2001/XMLSchema#string>	"4.24"^^<http://www.w3.org/2001/XMLSchema#float>

Figure 45: *OWL_Q4* SPARQL Query and result.

SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.semanticweb.org/rui/ontologies/2020/11/goodreads#>

select ?book_name ?rating_diff ?min_review_rating ?max_review_rating {

        bind(?max_review_rating - ?min_review_rating as ?rating_diff)
        {
        SELECT ?book_name (min(?review_rating) as ?min_review_rating) (max(?review_rating) as ?max_review_rating)
        WHERE {
                ?book a :Book .
                ?book :hasName ?book_name .
                ?book :hasReview ?review .
                ?review :hasRating ?review_rating .
        }
        GROUP BY ?book_name ?review_interval
        ORDER BY ?min_review_rating ?max_review_rating
        }
}
```

book_name	rating_diff	min_review_rating	max_review_rating
"Twilight"^^<http://www.w3.org/2001/XMLSchen "4.0"^^<http://www.w3.org/2001/XMLSchema#fl "0.0"^^<http://www.w3.org/2001/XMLSchema#fl "4.0"^^<http://www.w3.org/2001/XMLSchema#fl			
"Angels & Demons "^^<http://www.w3.org/2001 "5.0"^^<http://www.w3.org/2001/XMLSchema#fl "0.0"^^<http://www.w3.org/2001/XMLSchema#fl "5.0"^^<http://www.w3.org/2001/XMLSchema#fl			
"Catching Fire"^^<http://www.w3.org/2001/XMLS "4.0"^^<http://www.w3.org/2001/XMLSchema#fl "1.0"^^<http://www.w3.org/2001/XMLSchema#fl "5.0"^^<http://www.w3.org/2001/XMLSchema#fl			
"Divergent"^^<http://www.w3.org/2001/XMLSche "3.0"^^<http://www.w3.org/2001/XMLSchema#fl "2.0"^^<http://www.w3.org/2001/XMLSchema#fl "5.0"^^<http://www.w3.org/2001/XMLSchema#fl			
"Mockingjay"^^<http://www.w3.org/2001/XMLScl "3.0"^^<http://www.w3.org/2001/XMLSchema#fl "2.0"^^<http://www.w3.org/2001/XMLSchema#fl "5.0"^^<http://www.w3.org/2001/XMLSchema#fl			
"Het Achterhuis Dagboekbrieven 14 juni 1942 - "2.0"^^<http://www.w3.org/2001/XMLSchema#fl "3.0"^^<http://www.w3.org/2001/XMLSchema#fl "5.0"^^<http://www.w3.org/2001/XMLSchema#fl			
"The Hunger Games"^^<http://www.w3.org/200 "2.0"^^<http://www.w3.org/2001/XMLSchema#fl "3.0"^^<http://www.w3.org/2001/XMLSchema#fl "5.0"^^<http://www.w3.org/2001/XMLSchema#fl			
"Pride and Prejudice"^^<http://www.w3.org/200 "2.0"^^<http://www.w3.org/2001/XMLSchema#fl "3.0"^^<http://www.w3.org/2001/XMLSchema#fl "5.0"^^<http://www.w3.org/2001/XMLSchema#fl			
"Man som hatar kvinnor"^^<http://www.w3.org/2 "2.0"^^<http://www.w3.org/2001/XMLSchema#fl "3.0"^^<http://www.w3.org/2001/XMLSchema#fl "5.0"^^<http://www.w3.org/2001/XMLSchema#fl			
"To Kill a Mockingbird"^^<http://www.w3.org/20("1.0"^^<http://www.w3.org/2001/XMLSchema#fl "4.0"^^<http://www.w3.org/2001/XMLSchema#fl "5.0"^^<http://www.w3.org/2001/XMLSchema#fl			

Figure 46: *OWL_Q5* SPARQL Query and result.

SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.semanticweb.org/rui/ontologies/2020/11/goodreads#>

SELECT ?author_name (AVG(?book_rating) as ?author_rating)
WHERE {
?author a :Author .
?author :hasName ?author_name .
?author :authored ?book .
?book :hasRating ?book_rating .
}
GROUP BY ?author_name
ORDER BY DESC(?author_rating)
```

author_name	author_rating
"Harper Lee"^^<http://www.w3.org/2001/XMLSchema#string>	"4.25"^^<http://www.w3.org/2001/XMLSchema#float>
"Jane Austen"^^<http://www.w3.org/2001/XMLSchema#string>	"4.24"^^<http://www.w3.org/2001/XMLSchema#float>
"Veronica Roth"^^<http://www.w3.org/2001/XMLSchema#string>	"4.24"^^<http://www.w3.org/2001/XMLSchema#float>
"Suzanne Collins"^^<http://www.w3.org/2001/XMLSchema#string>	"4.2233334"^^<http://www.w3.org/2001/XMLSchema#float>
"Stieg Larsson"^^<http://www.w3.org/2001/XMLSchema#string>	"4.11"^^<http://www.w3.org/2001/XMLSchema#float>
"Anne Frank"^^<http://www.w3.org/2001/XMLSchema#string>	"4.1"^^<http://www.w3.org/2001/XMLSchema#float>
"Dan Brown"^^<http://www.w3.org/2001/XMLSchema#string>	"3.85"^^<http://www.w3.org/2001/XMLSchema#float>
"Stephenie Meyer"^^<http://www.w3.org/2001/XMLSchema#string>	"3.57"^^<http://www.w3.org/2001/XMLSchema#float>

Figure 47: *OWL_Q6* SPARQL Query and result.

SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.semanticweb.org/rui/ontologies/2020/11/goodreads#>

SELECT ?book_name ?review_date
WHERE {
?book a :Book .
?book :hasName ?book_name .
?book :hasReview ?review .
?review :wasPosted2n ?review_date_ent .
?review_date_ent :hasValue ?review_date
}
ORDER BY DESC(?review_date)
LIMIT 5
```

book_name	review_date
"Angels & Demons "^^<http://www.w3.org/2001/XMLSchema#string>	"2017-10-14"^^<http://www.w3.org/2001/XMLSchema#string>
"To Kill a Mockingbird"^^<http://www.w3.org/2001/XMLSchema#string>	"2017-10-01"^^<http://www.w3.org/2001/XMLSchema#string>
"The Hunger Games"^^<http://www.w3.org/2001/XMLSchema#string>	"2017-09-08"^^<http://www.w3.org/2001/XMLSchema#string>
"Pride and Prejudice"^^<http://www.w3.org/2001/XMLSchema#string>	"2017-08-19"^^<http://www.w3.org/2001/XMLSchema#string>
"Divergent"^^<http://www.w3.org/2001/XMLSchema#string>	"2017-08-05"^^<http://www.w3.org/2001/XMLSchema#string>

Figure 48: *OWL_Q7* SPARQL Query and result.

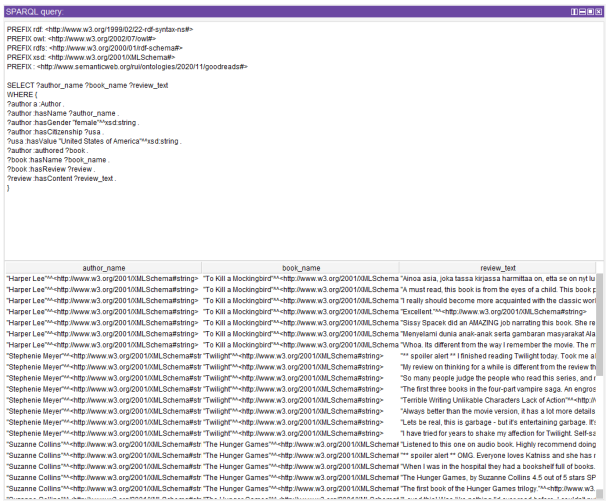


Figure 49: *OWL_Q8* SPARQL Query and result.