Goodreads' Books and Reviews

Diogo Almeida
up201806630@edu.fe.up.pt

Pedro Queirós
up201806329@edu.fe.up.pt

ABSTRACT

Over time, the amount of data available online has grown in an unimaginable rate, reinforcing the need to have mechanisms to connect and gather all the available information. This document concerns the development process of one of those mechanisms: an information system on Goodreads' books and its reviews. To obtain a dataset with relevant and suitable information, data refinement and enrichment were performed. Furthermore, the dataset was analyzed for a better understanding of the available data, with some statistics being made for that same purpose. Additionally, after using *Solr* in order to index the previously processed documents, the evaluation of several system configurations was made, demonstrating that the retrieval system's capacity is highly dependent on the defined indexing schema and the use of weighting filters.

CCS CONCEPTS

• **Information systems → Evaluation of retrieval results**.

KEYWORDS

Dataset , Data Extraction, Data Preparation, Data Cleaning, Book, Review, Domain Conceptual Model, Pipeline, Python, Pandas, Solr, Schema, Query, Evaluation

1 INTRODUCTION

Books have been playing an important role in society since the early times of humanity. Nowadays, books can be read in many different formats and have numerous purposes: to transmit knowledge about a specific subject, to tell a story, to serve as a record for future generations, among others.

The current panorama of book search systems is vast in regards to the information that it is able to retrieve, letting users mainly search for titles, authors, keywords or genres (e.g. Amazon Book Search [1] or WorldCat Search [12]). The main goal of this project is to complement these types of search systems with a search engine that allows users to also search for books based on e.g., reviews, descriptions and awards, in order to provide an easier and better experience when trying to find a book to read that fits their preferences.

This article describes the first and second development phases of this search system and is divided into two major sections that characterize the several steps taken:

The Data Collection section, which details the data preparation process, starts with **Data Extraction and Enrichment**, which describes the process of data gathering and enrichment, as well as detailing the source of the information. This is an essential step to guarantee that relevant data for the problem is used to create the first version of the dataset. The next section is **Data Preparation**, that details the procedure of cleaning and refining the collected data in order to have a consistent and practical dataset that is easier to handle. The **Domain Conceptual Model** section details how data can be conceptually described in the domain. It is followed by the **Possible Search Tasks** section, in which some of the possible queries to do in the system are listed. Finally, the **Dataset Characterization** section details how tools like plots and graphs are essential to better explore and understand all the collected information.

The Information Retrieval section, that describes the implemented information retrieval system and evaluates the results of possible retrieval tasks. In Section 7, we have an **Information Retrieval Introduction**, where we describe the goals of an information retrieval system and detail the three different systems we'll use along with the methods to compare the global performance of these systems. The **Information Retrieval Tool Selection** contains a brief comparison between the two main search engines considered for the project. Afterwards, the system's collections and documents are described in the **Collections and Documents** section. The following section, **Index Processing**, explains the documents' index processing, along with the description and characterization of the filters applied to a newly created field type, used in the most relevant fields. In the **Retrieval Process** section, the system is evaluated by comparing the performance of the three different system configurations on various information needs. Lastly, for the **Information Retrieval Tool Evaluation**, we present some remarks regarding the chosen search tool and explain why it ultimately was the best choice for the project.

Finally, for the **Conclusions and Future Work**, we shortly recap all the sections in this paper and explain our main findings. We also present some of our plans for the future development phases of the project.

2 DATA EXTRACTION AND ENRICHMENT

In order to extract the best information for the project, various open data platforms were consulted. After some research, Kaggle [6] proved to be the best solution. Kaggle is a platform for data scientists and machine learning practitioners to publish datasets and explore and build models. The dataset from Kaggle, **'Goodreads Books - 31 Features'** [8], which has around 150 MB, contains 52,199 books with 31 features each. These features include several relevant aspects about a book, like its title, author, description, publisher, the series it belongs to, various details about its rating (1-5 stars), and more. There are also, for each book, the "recommended_books" feature, that contains books that are recommended by Goodreads and the "books_in_series" feature that contains the books that belong to that book's series, it is important to note that all the books mentioned (by id) in these 2 features are also present in the dataset.

After an initial review of the Kaggle dataset, it was decided to enrich the project's data further with more text-rich fields. Therefore, another dataset was created using some reviews from each book in the Kaggle dataset. These reviews were scraped from the Goodreads website [5] and include the review's author, text and publication date. The information was scraped and exported to a

.csv file using a Python script with the *Requests [9]* and *BeautifulSoup [10]* libraries. This .csv file ended up with a size of around 372 MB, containing around 8 reviews for each book.

3 DATA PREPARATION

Before proceeding with its preparation, the data was first carefully analysed to ensure that only the relevant information for the project was selected based on the possible search tasks that could be done.

Firstly, we removed the columns that don't have relevant information or that have a significant amount of NaN (Not a Number) values, for instance, the "asin" (Amazon Standard Identification Number) column.

Secondly, books that didn't have an original title got this column filled with their title.

Thirdly, the books without all the relevant information, for instance, title, publisher or author, were removed from the dataset, resulting in 35,880 books.

Next, due to bad encoding in the original dataset, there were some strange characters appearing in books that had column values that weren't in English. This was fixed so all characters from all languages appear correctly.

After that, books with descriptions containing less than 25 characters were also removed for not having enough information, leaving 35,655 books.

Books that weren't in the reviews dataset also ended up being removed, resulting in the final 35,347 books. Although the reviews dataset was scraped using the books from the original dataset, it wasn't possible to scrape all of them due to errors, so they had to be removed from the dataset as well.

Finally, the referenced books in the "recommended_books" or "books_in_series" columns that weren't in the dataset (because they were deleted by the previous operations) were also removed. Some columns (awards, genres and their votes, and characters) were extracted to a new .csv file to make it easier to analyse the final data that will be used. These columns were also removed from the original dataset in order not to have duplicated information. All of these operations were made using *Pandas [7]*, a Python library.

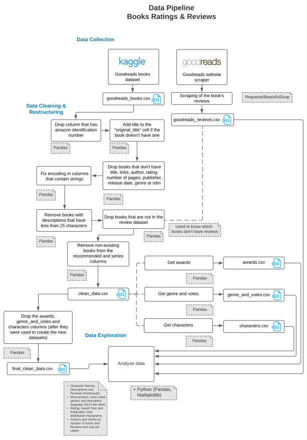

Data Pipeline
Books Ratings & Reviews
Data Collection
kaggle
Goodreads books dataset
goodreads
Goodreads website scraper
goodreads_books.csv
Scraping of the book's reviews
Requests/BeautifulSoup
Data Cleaning & Restructuring
goodreads_reviews.csv
Drop column that has amazon identification number
Add title to the "original_title" cell if the book doesn't have one
Pandas
Pandas
Fix encoding in columns that contain strings
Drop books that don't have title, links, author, rating, number of pages, publisher, release date, genre or isbn
Pandas
Pandas
Remove books with descriptions that have less than 25 characters
Drop books that are not in the review dataset
Used to know which books don't have reviews
Pandas
Pandas
Remove non-existing books from the recommended and series columns
Get awards
awards.csv
Pandas
Pandas
clean_data.csv
Get genre and votes
genre_and_votes.csv
Pandas
Drop the awards, genre_and_votes and characters columns (after they were used to create the new datasets)
Get characters
characters.csv
Data Exploration
Pandas
Pandas
final_clean_data.csv
Analyse data
Character Names, Descriptions and Reviews Wordclouds;
Most present, most voted genres and description language (NLP) bar plots;
Rating, Award Year and Publication Year distribution histograms;
Authors and Series by number of books and Reviews text size pie charts
Python (Pandas, Mathplotlib)

Figure 1: Data Pipeline

4 DOMAIN CONCEPTUAL MODEL

A domain conceptual model was made to represent the main entities of the domain. The book class is the center of the conceptual model and it has the following attributes: Goodreads id, the title, the original title, the links for the book's cover, Goodreads page, Amazon and WorldCat, number of ratings, average rating, number of pages, isbn, isbn13 and the book's description. Every book was written by at least one author and published by one publisher. A book can belong to a series (collection), can have settings (which are the locations where the story unfolds) and can have characters. Each book has several genres which have several votes from the readers. Furthermore, they also have the number of votes for the ratings from 1 to 5 and some reviews made by some Goodreads users. Finally, books still have awards with the respective year when they have won it and recommended books based on their genre or rating.

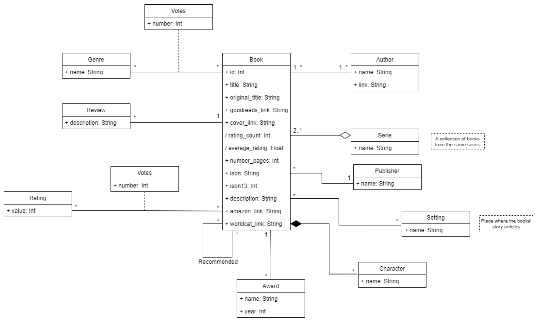


Figure 2: Conceptual Model

5 DATA CHARACTERIZATION

With the aim of better understanding and classifying all the collected data, charts were developed regarding some of the dataset's most interesting and striking features. These features and respective charts are analyzed next.

A. Awards and Books

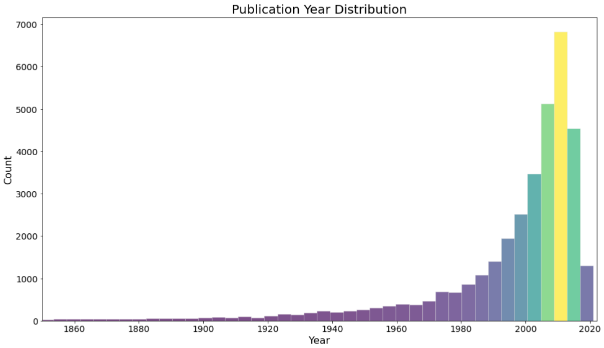


Figure 3: Publication Year Distribution

Regarding the year a book was published, it can be observed in **Figure 3** that the dataset mainly focuses on modern releases, containing many books published after the 2000's. Another interesting note is that there seems to be an increase in included books published from the 1800's onwards.

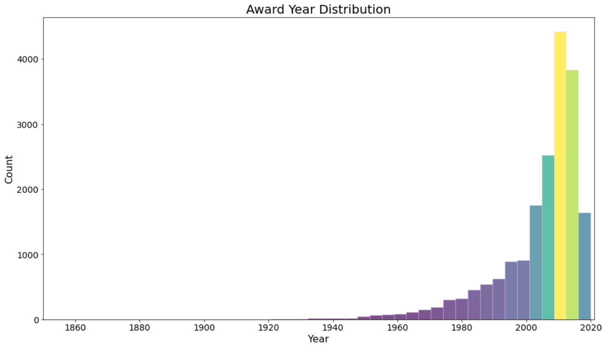


Figure 4: Award Year Distribution

The year an award was given also follows the same trend as the book publication year distribution seen previously, with most awards being handed out after the 2000's. This makes sense, since most books present in the dataset were published around this time.

About 8,948 of the total of 35,347 books (25.3%) were given an award. Since the dataset contains 19,186 awards, a lot of these awarded books have been distinguished multiple times: 4,018 to be exact (44.9% of the awarded books). The fact that a considerable chunk of books got an award is a good indicator of positive book ratings, a feature that will be analyzed later.

B. Book Genres

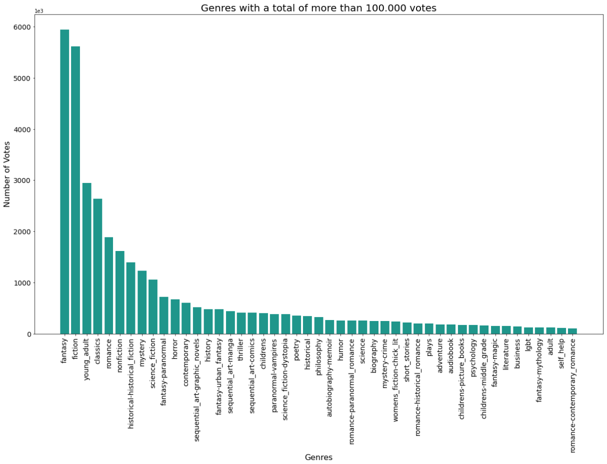


Figure 5: Genres with a total of more than 100000 votes

In the Goodreads website, the source of the books dataset, users can vote in the genres they think the book fits into. Upon analyzing **Figure 5**, the conclusion is that the "fantasy" and "fiction" genres are the most voted, with almost 6,000,000 votes each, 3,000,000 more than the third most voted genre "young_adult".

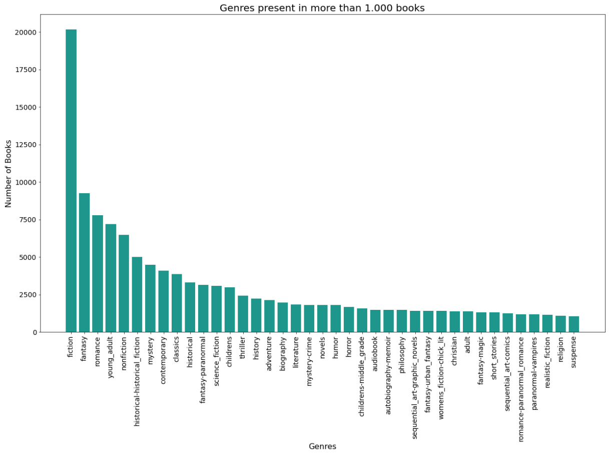


Figure 6: Genres present in more than 1000 books

The bar plot present in **Figure 6** is similar to **Figure 5's**, however, it now counts the genres that are present (have at least one vote) in more than 1,000 books. In this chart, we can observe that the "fiction" genre is present in a lot more books than the "fantasy" genre, although both genres were very close in votes, as analyzed in the previous chart (with "fantasy" actually having more votes than "fiction"). From this we can deduce that, in our dataset, "fantasy" genre books are fewer, but more popular (have more votes and therefore more users reading it) than "fiction" genre books, that need presence in more books to match the amount of votes that the "fantasy" genre has. In fact, the votes to books ratio in the "fantasy" genre is superior to every other genre.

C. Book Ratings

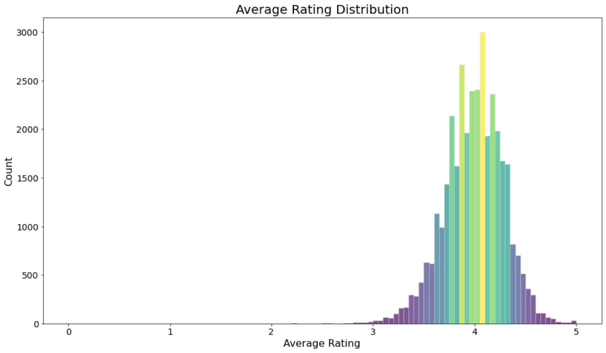


Figure 7: Average Book Rating Distribution

By observing **Figure 7** we can see that, in a classification of 0-5 stars, most books in the dataset have a rating of around 4 stars. This large amount of positive ratings was expected, since in the book awards analysis done before, we saw that there was a good amount of awarded books, which generally means that those books are acclaimed by both critics and readers/users.

D. Authors

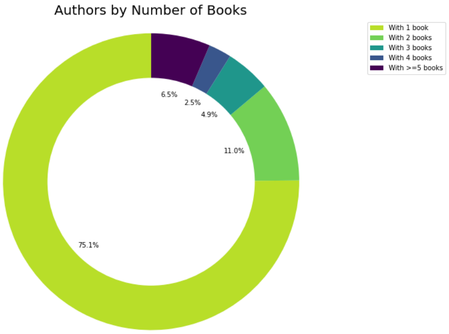


Figure 8: Authors by Number of Books

From observing **Figure 8** we conclude that the majority of authors present in the dataset (75.1%) only published 1 book. Since the dataset contains 35,347 books and 19,171 authors, this means 14,405 books were written by authors that only wrote 1 book. Therefore, 20,942 books, which is more than half of the books dataset (59.2%), were written by only 4,781 authors, which averages at about 4/5 books per author, despite that, according to the pie chart, only 6.5%+2.5%=9% (1716) of authors have written more than 3 books. This might happen because of the large percentage (46.2%) of books that belong to a series and are, consequently, written by the same author.

E. Book Descriptions and Reviews

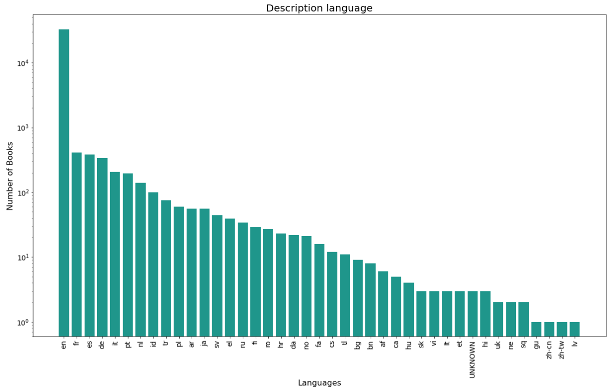


Figure 9: Book Descriptions Language

In **Figure 9**, we explore the different languages the descriptions were written in. The language for the descriptions was detected using the Python library *spacy-langdetect [3]*. From the chart, which uses a logarithmic scale, we conclude that the vast majority of descriptions are in English, with French, Spanish, German and other languages following after with less descriptions. Analyzing the book description language was an attempt of obtaining a rough approximation of each book's original idiom, although some of the descriptions end up being translated to English in the website, especially for popular books.

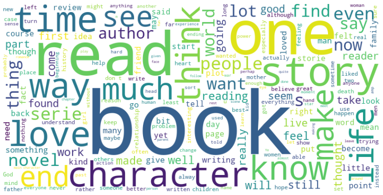


Figure 10: Book Reviews Word Cloud

For the book reviews, we decided to generate a word cloud plot (**Figure 10**) to try and see their general tone. Naturally, there are a lot of words associated with the elements of books such as "read", "character", "story" or even the word "book" itself. What is interesting is the large amount of positive words, such as "love", "good", "great" or "well", which might be related to the high rating distribution the books in the dataset have, as analysed before.

6 POSSIBLE SEARCH TASKS

Following an extensive data analysis of the main features that made the group acknowledge the dataset's relevant information, we can now consider some possible search tasks to be used:

- Search for the biography of an historical figure;
- Search for comic books featuring a specific character as the protagonist;
- Search for non-fictional books about the research and work of one of history's greatest minds;
- Search for fantasy children's book that are easy to read;

7 INFORMATION RETRIEVAL INTRODUCTION

The main goal of an information retrieval system is to find documents within a collection that are relevant in order to satisfy a user information need, while retrieving as few non-pertinent documents as possible. This research work specifically focuses on the most common retrieval task: *ad hoc* search, where the information need is provided through a user-initiated query that is performed on all documents of the collection.

With this objective in mind we will be using *Solr*, a popular open source search platform that offers various features which enhance the retrieval process. The platform allows the definition of custom filters for different fields in order to create dynamic searches, while also providing the possibility to grant weights to specific fields to boost their relevance. Having said that, to study and evaluate the results of possible search tasks, the following three systems will be utilized:

- A baseline system, which only performs basic matching on key fields;
- A system where the main textual fields are indexed with the usage of filters;
- A system that uses weights in order to boost relevant fields in the retrieval process.

Moreover, a set of information needs, defined in the previous Section 6, will be translated to textual queries and tested in each system. Each of these queries will then be evaluated based on their Precision @ 10 (*P@10*) and Average Precision (*AvP*) values. Therefore, *P@10* measures the precision, translating into the fraction of relevant retrieved documents, and *AvP* is the average of the precision values obtained for the set of the top K documents existing after each relevant document is retrieved.

Finally, to compare the global performance of all three systems we will use the Mean Average Precision (*MAP*), which is obtained by averaging the *AvP* values off all the systems.

8 INFORMATION RETRIEVAL TOOL SELECTION

After an initial examination, the two main tools that were considered for information retrieval were *Apache Solr*[11] and *Elasticsearch*[4]. Both of these tools are open-source, are built on top of *Lucene* [2] and offer a great variety of features, for instance, distributed full-text search, near real-time indexing, high availability and support for *NoSQL* data.

Despite being the older tool, *Solr* ended up being our final choice, as mentioned in the previous Section 7. This is because *Solr* is best suited for search applications that use significant amounts of static data, which perfectly fits the problem at hand, as described in the previous sections.

While Solr has a lot of advantages and useful features, such as support for rich-text documents and complex search queries, it also has a significant disadvantage: its poor documentation and lack of examples/tutorials. This was especially troublesome because most of the already lacking examples and documentation were all directed towards schemas in *XML*, making it harder for the group to implement the schema using *JSON*, as presented in the class tutorials.

9 COLLECTIONS AND DOCUMENTS

The relevant data for the information system is mainly split in two .csv files, regarding the books and their reviews. There were two approaches we considered in order to insert this data into *Solr*: the first one consisted in creating two types of documents, books and reviews, in the same core, creating the necessity of using several schemas or a single complex schema in order to be compatible with the different types of data. For this approach, we were also forced to use the *join* operation in queries, which wasn't ideal; the second one involved having only one type of document containing the books and their reviews in one of the fields, allowing the use of a single flexible schema compatible with the different types of data. However, merging the reviews into one field of the books dataset meant that we could only use the text content field from the reviews, having to discard the author and the date, which wasn't an issue since they weren't relevant for searching. Since the first approach made querying much harder, we decided to opt for the second approach. With this in mind, the two .csv files were merged and converted to a *JSON* file, so they could be put into *Solr* along with the flexible schema created, making the information ready to be queried on.

10 INDEXING PROCESS

One of the most crucial stages in Information Retrieval is Indexing, which minimizes the documents to the relevant terms contained in them. Before beginning this process, we analyzed which fields from each document were appropriate for indexing. For a field to be suitable it has to be useful for searching books, fulfilling the previously defined information needs, by serving as a query parameter. The documents were indexed using *Solr's* Post tool and the final schema can be consulted in **Table 1**

Table 1: Schema Field Types

Field	Type	Indexed
title	standard_text	Yes
link	string	No
series	standard_text	Yes
cover_link	string	No
author	standard_text	Yes
author_link	string	No
rating_count	pint	No
average_rating	pfloat	No
five_star_ratings	pint	No
four_star_ratings	pint	No
three_star_ratings	pint	No
two_star_ratings	pint	No
one_star_ratings	pint	No
number_of_pages	pint	Yes
date_published	standard_text	Yes
publisher	standard_text	Yes
original_title	standard_text	Yes
genre_and_votes	standard_text	Yes
isbn	string	Yes
isbn13	string	Yes
settings	standard_text	Yes
characters	standard_text	Yes
awards	standard_text	Yes
amazon_redirect_link	string	No
worldcat_redirect_link	string	No
recommended_books	standard_text	Yes
books_in_series	standard_text	Yes
description	standard_text	Yes
reviews	standard_text	Yes

Considering the project's context, it is important to modify *Solr's* indexing pipeline for the fields with unstructured data. The fields with native values were defined using the default *Solr* field types, such as *string*, *pfloat* and *pint*. Furthermore, all of the important text fields e.g., title, series, author, description and reviews were defined using an analyzer pipeline, which contained both tokenizers and filters for the enhanced processing of each individual token.

Having said that, a new field type for this textual content, ***standard_text***, was created using the following filters:

- *ASCIIFoldingFilterFactory*, which converts alphabetic, numeric, and symbolic *Unicode* characters which are not in the Basic Latin *Unicode* block to their *ASCII* equivalents, if one exists.

- *LowerCaseFilterFactory*, which converts any uppercase letters in a token to the equivalent lowercase token.
- *SynonymFilterFactory*, which does synonym mapping. Each token is looked up in the list of synonyms and if a match is found, then the synonym is emitted in place of the token.

This custom field type uses these filters for both indexing and querying, while also making use of *Solr's Standart Tokenizer*, which splits the text field into tokens, treating whitespace and punctuation as delimiters.

11 RETRIEVAL PROCESS

In order to evaluate the performance of the different systems detailed in Section 7, as well as the impact of filters and weights/boosts, we defined four different information needs. Each information need contains a basic description, a user story to explain its context, its relevance judgement to settle if a document is relevant or not and the query that represents it. For the query section, the meaning of the abbreviations used for the *eDismax* parameters is the following:

- **Query (q)**;
- **Filter Query (fq)**;
- **Query Field with optional boost (qf)**: The fields where the search is made;
- **Phrase boosted Field (pf)**: Assigns a boost depending on the proximity of the searched words;
- **Phrase boost Slope (ps)**: Maximum amount of tokens wanted between between the searched words.

With the goal of evaluating each system performance for each information need we analyzed the top 10 query results and their relevance, allowing us to calculate the *P@10* and *AvP*, and draw conclusions based on these values afterwards.

One important detail to note is that the query fields vary with each information need. These fields and their respective weights (for System 3), which were determined by following an *Ad Hoc* approach, are presented before every information need.

A. Bibliographies of Adolf Hitler

Table 2: Weights Distribution (System 3)

Field (qf)	Weight
title	3
genre	2
description	1
author	3

Information Need: Biographies about Adolf Hitler.
User Story: "As a History student studying about World War II, I want to find books and biographies that portray the life of Adolf Hitler".
Relevance Judgement: In this information need we intend to search for books that talk about Adolf Hitler, more specifically, biographies about the dictator. Since the book is a biography, the book has to have "Biography" as one of its genres or have the word "biography" in the title or description. It is also important for "Hitler" to be mentioned in the book's title or be the book's author. Mentions of this word may also appear in the description, despite having less importance.
Query:

- **q**: hitler and biography
- **qf**: title, genre_and_votes, description, author

Table 3: Bibliographies of Adolf Hitler Information Need Results

Rank	System 1	System 2	System 3
AvP	0.781429	0.781429	0.836376
P@10	0.6	0.6	0.7

Result Analysis: The first two systems have the same performance while the third one has slightly better results as we can see in **Table 3** and **Figure 15** of the attachments. This last system has a better precision than the other two, which means that relevant results will appear sooner. Since the word "Hitler" appears in many descriptions of books about, e.g., the Second World War or the Holocaust, Systems 1 and 2, that give a book's description the same importance as the title or the author will be more likely to show these irrelevant results. On the other hand, System 3, that weighs the fields in favor of the book's title, genre and author, will certainly find biographies written by or about Hitler more easily.

B. Comic books whose protagonist is Spider-Man

Table 4: Weights Distribution (System 3)

Field (qf)	Weight
title	3
genre	2
description	1
characters	2

Information Need: Comic books featuring Spider-Man as the main character.
User Story: "As a Spider-Man fan, I want to find comic books whose protagonist is Spider-Man".
Relevance Judgement: For this information need we intend to retrieve comic books in which Spider-Man appears as the main character. Therefore, the words "Spider-Man" (or its synonyms) or "Peter Parker" should most importantly appear in the title, which most likely means the character is the protagonist. The superhero's name may also appear in the description or in the "characters" field, although with less importance than the title. Finally, since we're searching for a comic book, one of the book's genres must be "Comics".
Query:

- **q**: comics AND ("spider-man" OR "Peter Parker")
- **qf**: title, genre_and_votes, description, characters

Table 5: Comic books whose protagonist is Spider-Man Information Need Results

Rank	System 1	System 2	System 3
AvP	0.971429	0.893519	1
P@10	0.8	0.7	0.9

Result Analysis: The three systems have similar average performances, as it can be seen in **Table 5** and **Figure 16** of the attachments. Although we were expecting very high results, since we were using the book's title as a search field, which usually features the main character's name, some books where Spider-Man appears as a side character still appeared, not being so relevant for the goal of the query. The third system has a better performance since it assigns a higher weight to the title and characters, along with the use of the synonyms file. System 2 has a slightly lower performance than System 1 because the use of synonyms alone, especially in the book's description, can be misleading.

C. Non-fictional books about the life's work of Albert Einstein

Table 6: Weights Distribution (System 3)

Field (qf)	Weight
title	3
genre	2
description	1
author	3
characters	3

Information Need: Non-fictional books about the research and studies of Albert Einstein.
User Story: "As a person investigating Albert Einstein's work, I want to find non-fictional books about his research, both Scientific and Philosophical".
Relevance Judgement: With this information need we want to find books that detail Albert Einstein's career work, be it Scientific or Philosophical. First of all, because we want books that explain and explore Einstein's work, the scientist should either be the book's author or have his name feature in either the title or the book's characters (or both). The name can also appear in the description although it is less relevant. Additionally, since we are looking for non-fictional books, this genre must be part of the book's genres, along with "Science" or "Philosophy".
Query:

- **q**: einstein AND -fiction AND (philosophy OR science)
- **qf**: title, genre_and_votes, description, characters

Table 7: Non-fictional books about the life's work of Albert Einstein Information Need Results

Rank	System 1	System 2	System 3
AvP	0.582275	0.565608	0.92
P@10	0.6	0.6	0.6

Result Analysis: The third system has a better average performance than the other two models as we can observe in **Table 7** and **Figure 17** of the attachments. Einstein's name comes up frequently in books, being often referred to as an inspiration for modern scientists or even used to describe an intelligent person. A system that focuses more on the books' titles and their authors will show relevant results sooner, which is what System 3 does. The first and the second systems give the same importance to the books' title, genre, description and author, frequently "catching" these books that mention "Einstein" outside of the information need's context, resulting in worse performance.

D. Fantasy children's books set in a medieval era that are easy to read

Table 8: Weights Distribution (System 3)

Field	Weight
review (qf)	2
review (pf)	5
review (ps)	5

Information Need: Easy to read children-friendly books set in the medieval era.
User Story: "As a parent, I want to find children-friendly books set in the medieval era that are easy to read".
Relevance Judgement: For this information need, relevant books portrait easy-to-read fantasy children's books, set in a medieval era. First of all, the relevant books must feature "Fantasy" and "Childrens" as two of their genres. Since the books are set in a medieval era, the description must mention the word "kingdom". Finally, pertinent books must also include the words "easy" and "read" with at most five words of distance (possible adjectives or connectors in between) in at least one of their reviews.
Query:

- **q**: easy read
- **fq**:
  - **genre_and_votes**: fantasy
  - **genre_and_votes**: childrens
  - **description**: kingdom
- **qf**: reviews
- **pf**: reviews
- **ps**: reviews

Table 9: Fantasy children's books set in a medieval era that are easy to read Information Need Results

Rank	System 1	System 2	System 3
AvP	0.493386	0.50496	0.862857
P@10	0.6	0.6	0.6

Result Analysis: Once again, the third system has the best performance compared to the other two, as we can see in **Table 9** and **Figure 18** of the attachments. In order to find books that are "easy to read" we are required to search beyond the book's title, description or genre, since these don't usually offer an opinion on the readability or quality of a book. Therefore, to obtain this information, we have to search in the users' reviews. The first two systems cannot match the words "easy" and "read" with adjectives or connectors between, e.g., "easy to read" or "easy and fast read". However, System 3 can match strings with up to five words between "easy" and "read", assigning different weights based on the number of words between, making it better for the goal of this query.

E. General Conclusions

Taking into account all the results from the multiple information needs across the different systems, we can now calculate the MAP (*Mean Average Precision*) of these three systems.

Table 10: MAP values for the three Information Systems

System 1	**System 2**	**System 3**
0.707130	0.686379	0.904808

Table 10 contains the MAP values for each system, and will allow us to compare the systems on a more general scope in order to get a better overview of their performance.

By observing the results, we can conclude that System 3 is an improvement of both Systems 1 and 2, which was expected because of the schema and different weights assigned to each field for each query, allowing a more accurate and precise search depending on the context of the information need. We can also see that the results from System 1, which is schema-less, are slightly better than the results from System 2. Although there isn't a huge difference, we can conclude that a system with this particular schema, while being better at finding and matching keywords, is more vulnerable to ambiguity, frequently matching keywords in fields that aren't that relevant for that particular information need (since there is no weight), which is precisely what we can observe in the information needs present in Subsections B and C, where the query matched more keywords in the description than in the title, thus being less relevant.

12 INFORMATION RETRIEVAL TOOL EVALUATION

Following the Information Retrieval phase, we have some comments on *Solr*, the tool used for this process:

- As mentioned before, *Solr's* documentation is very limited, making it difficult to learn how to perform certain tasks since there are very few practical examples and tutorials, especially for the chosen schema format: *JSON*
- The initial setup and configuration of the tool was not user-friendly, with the lack of any clear documentation making it significantly more difficult.
- However, once we got the platform setup correctly, *Solr* offers many useful features, allowing the definition of complex queries while having a very fast query response time.

Overall, although *Solr* does take some time to learn and configure, it still allows for the implementation of a good information retrieval system, as the previously discussed results show.

13 CONCLUSIONS AND FUTURE WORK

In this paper we presented and detailed the procedures done during the development of a book search system that aims to provide a richer search experience, letting its users search for books based on e.g., reviews, descriptions or awards.

In the initial sections of the paper we detailed the data characterization process that ultimately led to the creation of our dataset, explaining the operations done to both integrate different sources of information and to clean data. The result was a much better structured dataset that could be easily integrated into a search system.

The next stage focused on the information retrieval process. We began by researching about different search tools and comparing them to see which one fitted our problem better, with the final choice being *Solr*. Following that, we detailed our indexing process, where we explained the schema, filters and tokenizers we used in order to optimize the system and achieve better results.

Finally, we explored the actual information retrieval process, where we selected four different information needs and compared their performance through the precision scores from the first ten documents across three different systems, revealing the performance boost the added schema and weights were causing. We concluded that System 3 was having better results due to the previously performed optimizations.

Regarding future work, we will focus on further improving our queries by, for example, creating a new field type to allow a search by ranges (useful when searching for a book with a specific number of pages) or using Natural Language Processing to better detect certain keywords in text-rich fields. We will also try to enhance the overall search experience by creating a graphical user interface that integrates the search system.

REFERENCES

[1] *Amazon Book Search.* URL: https://www.amazon.com/advanced-search/books (visited on 11/16/2021).

[2] *Apache Lucene - Welcome to Apache Lucene.* URL: https://lucene.apache.org/ (visited on 12/14/2021).

[3] Abhijit Balaji. *Spacy Python Library.* URL: https://spacy.io/universe/project/spacy-langdetect (visited on 11/15/2021).

[4] *Elasticsearch: The Official Distributed Search Analytics Engine.* URL: https://www.elastic.co/elasticsearch/ (visited on 12/01/2021).

[5] Goodreads.com. *Goodreads | Meet your next favorite book.* URL: https://www.goodreads.com/ (visited on 11/03/2021).

[6] Kaggle. *Your Machine Learning and Data Science Community.* URL: https://www.kaggle.com/ (visited on 10/27/2021).

[7] Wes McKinney et al. *Pandas Python Library.* URL: https://pandas.pydata.org/ (visited on 11/02/2021).

[8] Austin Reese. *Goodreads Books - 31 Features.* July 2020. URL: https://www.kaggle.com/austinreese/goodreads-books (visited on 11/02/2021).

[9] Kenneth Reitz et al. *Requests Python Library.* URL: https://docs.python-requests.org/en/latest/ (visited on 11/07/2021).

[10] Leonard Richardson et al. *BeautifulSoup Python Library.* URL: https://beautiful-soup-4.readthedocs.io/en/latest/ (visited on 11/07/2021).

[11] *Welcome to Apache Lucene.* URL: https://lucene.apache.org/ (visited on 12/14/2021).

[12] *WorldCat Book Search.* URL: https://www.worldcat.org/ (visited on 11/16/2021).

14 ATTACHMENTS

The following pages contain the attachments for all the mentioned figures in this paper, it also contains additional plots and tables used during research.

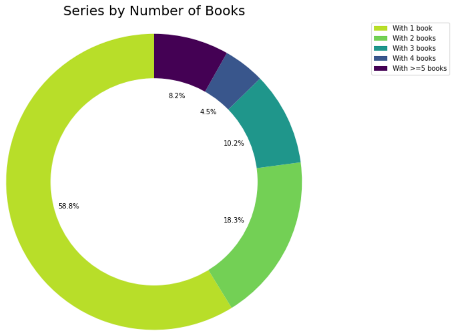


Figure 11: Series by Number of Books

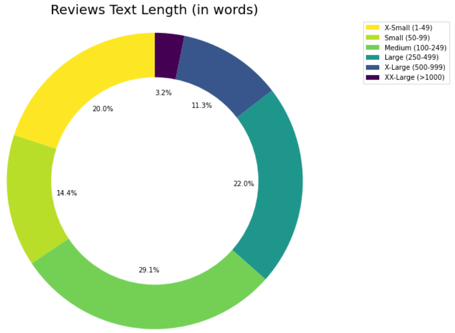


Figure 12: Reviews Text Length

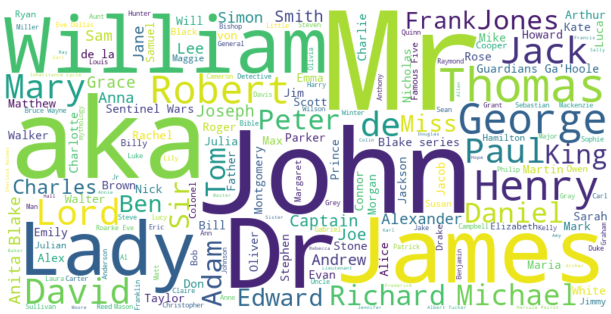


Figure 13: Book Characters Word Cloud

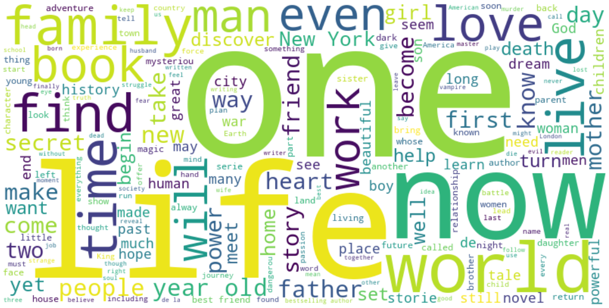
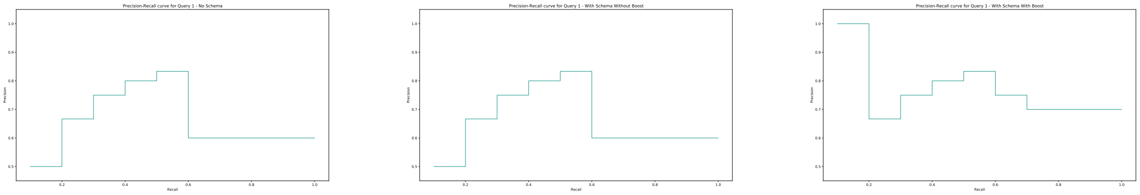


Figure 14: Book Descriptions Word Cloud

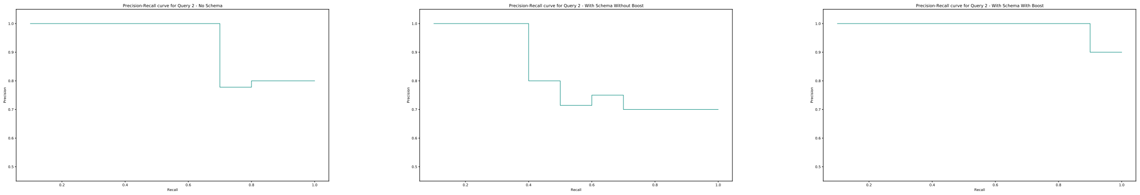


(a) Without schema

(b) With schema without weights

(c) With schema with weights

Figure 15: Bibliographies of Adolf Hitler - Evaluation on 3 systems

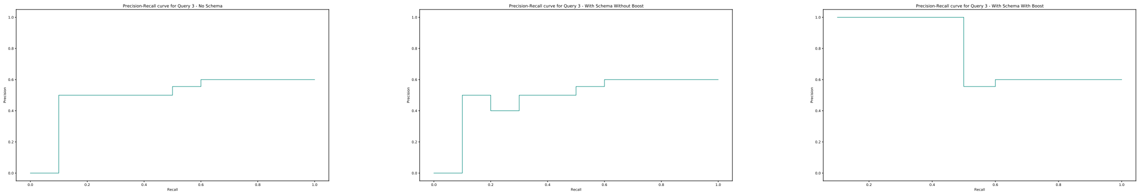


(a) Without schema

(b) With schema without weights

(c) With schema with weights

Figure 16: Comic books whose protagonist is Spider-Man - Evaluation on 3 systems

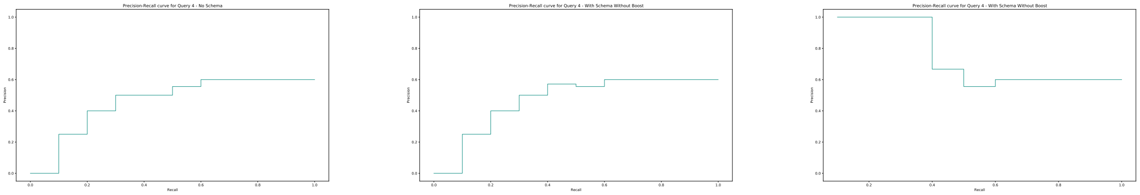


(a) Without schema

(b) With schema without weights

(c) With schema with weights

Figure 17: Non-fictional books about the life’s work of Albert Einstein - Evaluation on 3 systems



(a) Without schema

(b) With schema without weights

(c) With schema with weights

Figure 18: Fantasy children’s books set in a medieval era that are easy to read - Evaluation on 3 systems