Advanced Book Search System: Leveraging Reader Reviews for Efficient Information Retrieval

ANA SOFIA TEIXEIRA, RICARDO NUNES RIBEIRO, SÉRGIO MANUEL CARVALHAIS, and TOMÁS AGANTE MARTINS

ABSTRACT

This paper presents the data collection, preparation, analysis, and querying of a dataset. An initial search was made to find a suitable dataset to be used for the project. After settling with a dataset about book reviews and respective informations, retrieved from Kaggle, there was an initial data analysis of the data and then, data exploration using Python libraries. Afterwards, the data was cleaned and refined using Python scripts. Through the usage of Solr, an information retrieval tool, we define a collection and its respective documents, and describe their indexing process. The defined information needs are queried, while also exploring some of the tool's features, and evaluated according to some relevant metrics that compare the different system's configurations. After this initial testing and evaluation, Solr's searching features were explored to further enhance the system, and finally more testing was done in order to compare the new behaviour to the original system. In the last stage of the project, the schemas and features were explored, so this paper displays both the metrics of based and enhanced systems. In the end, an intuitive and appealing user interface was developed, to provide a good experience when using our system.

KEYWORDS

Book Reviews, data collection, data source, data preparation, information retrieval, search engine.

1 INTRODUCTION

While existing book search systems rely on conventional criteria, our project introduces a novel technical solution. We aim to develop a search engine that enables users to search books based on reader reviews. This article serves as a comprehensive overview of the development phase of our project, focusing on the technical aspects. Our goal is to create a technically robust book search system that enhances the user experience by considering reader reviews and detailed content information.

We begin by sourcing and curating relevant data sets, assess data quality and authority, and conduct exploratory data analysis. Additionally, we document our data processing pipeline. Our work involves characterising data set properties, defining the conceptual model for our book data, and meticulously identifying and characterising information needs for the rest of the development of the project.

After the foundational phase, our project advanced to information retrieval, employing specialised tools and methodologies for efficient data set access. This stage involved developing an information retrieval tool, document analysis, and constructing an index. The emphasis was on refining query execution to enhance accurate information retrieval.

Subsequently, we rigorously evaluated our information retrieval systems, conducting relevance assessments using diverse setups and metrics. We analysed system performance across various configurations and metrics to gauge effectiveness.

These steps were followed by the dataset's schema improvement and repetition of the evaluation methodologies that were previously applied. Also, new information retrieval systems were tested by the usage of several methods like the notion of synonyms, keywords, semantic search, etc.

Finally, a web interface was created to make it so that users can query the database in a intuitive and easy to use way, that took the form of a search engine website, developed using the JavaScript's Vue.JS framework.

2 DATA SOURCES

In our quest to obtain data aligned with our research objectives, Kaggle [1] emerged as the ideal source. The data set of interest, 'Amazon Books Reviews' [2], comprises two key files. The first file details 212,404 distinct books, sourced from the Google Books API. The second file contains a treasure trove of Amazon reviews, totalling 3 million entries from May 1996 to July 2014. The book details file encompasses essential information like title, authors, publisher, and user ratings. In contrast, the reviews file provides insights into book titles, pricing, user details, review scores, and more.

What's noteworthy is that this data set is licensed under Creative Commons Zero (CC0), effectively dedicating it to the public domain and allowing unfettered usage and modification in accordance with copyright law. This open data approach encourages collaboration and innovation within the research community.

3 DATA COLLECTION AND PREPARATION

3.1 Data Preparation Pipeline

For the data collection phase, we first downloaded a .zip folder from the Kaggle website [2]. After having the folder locally on our devices, we unziped it and inside found the two files we required: *Books_ratings.csv* and *books_data.csv*.

From the name of both files, we quickly understood what was contained in them. the first one compiled all the reviews and ratings from the users associated with a given book. We became aware that there were 3 million entries of reviews in this file. The second one was more focused on storing the individual book information (212404 records of books), among them, their authorship and categorisation.

After having collected the data from the data source, we proceeded with some tasks in order to prepare our data for the next phases according to the pipeline depicted in Figure 1.

Authors' address: Ana Sofia Teixeira, up201806629@fe.up.pt; Ricardo Nunes Ribeiro, up202310095@fe.up.pt; Sérgio Manuel Carvalhais, up202007544@fe.up.pt; Tomás Agante Martins, up201704976@fe.up.pt.

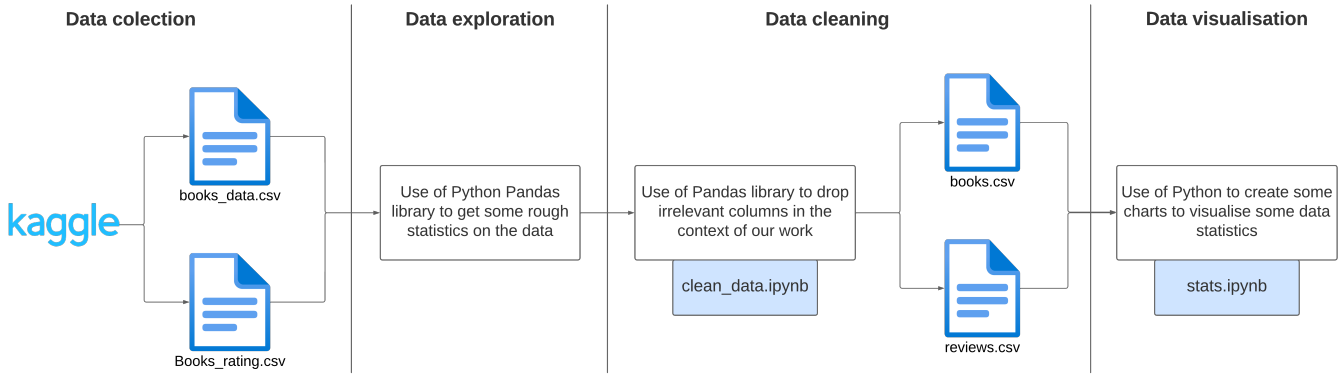

Fig. 1. Data Pipeline

In the data exploration phase, and through the use of the Python Pandas library, we were able to collect some initial data regarding the data sets. For instance, the column names of both tables; the number of missing values; the number of unique values in the columns (e.g.: there were 10883 unique values in the column 'categories' in *books_data.csv*). Later on, in the Data Cleaning phase, we decided that, in the context of our project, there was no need to store some of the information present in the files. Moreover, the files' size was considerably large, therefore, to facilitate the computing operations in our personal computers, we deleted those columns we deemed unnecessary: "Id", "Price", "User_id", "profileName", "review/helpfulness", "review/score" and "review/time" from *Book_rating.csv*; and "image", "previewLink", "infoLink" and "ratingsCount" from *books_data.csv*. Later on, we renamed the remaining columns in *Books_rating.csv* according to the following scheme: "Title" ↦ "book_title", "review/summary" ↦ "summary", "review/text" ↦ "text"; and "Title" ↦ "book_title", leaving all the others with the same name. As for *books_data.csv*, we renamed "Title" to "book_title" and left all others unchanged ("description", "authors", "publisher", "publishedDate", 'categories'). In the end of this phase, we wrote the new tables into new .csv files for organisation purposes, being them *reviews.csv* and *books.csv*.

Lastly, using a Jupyter Notebook, we created some plots in order to have a deeper look at the data at hands (cf. 3.3 Data Characterisation). Figure 1 represents a formal overview of the entire process.

3.2 Conceptual Data Domain Model

Having the data in such a way we could better handle and analyse, we were able to come up with a Conceptual Data Model. The model (Figure 2) identifies the entities we consider relevant in the context of our problem as well as their respective attributes. We point out the following:

- *book* entity
  - book_title: stores the name of the book
- *review* entity
  - summary: holds a summary/preview of the full review
  - text: stores the full review written by the user
- *author* entity
  - author: stores the name of the author
- *category* entity
  - category: records the name of the category the book falls into
- *publisher* entity
  - publisher: holds the name of the company who responsible for publishing the book

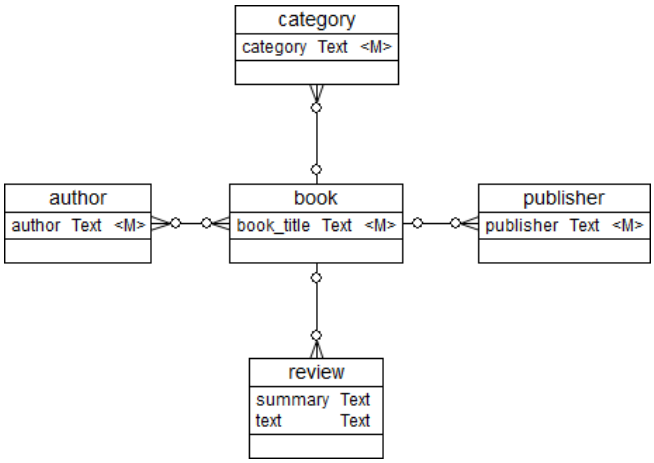

Fig. 2. Conceptual Data Model

4 DATASET CHARACTERISATION

In terms of our collection, the .csv files we obtained were extensive. They contained information regarding full-text reviews with reference to the books reviewed and also every information imaginable about the book, like Amazon internal reference number, no. of pages or the full book. To ease out our task and for the presentation of our topic to be clearer, we produced some charts using the python scripts in our pipeline.

Given that the main focus of this project is regarded with user reviews, we decided to better analyse this field in our data set. To

do so, we created some charts that reflect the characteristics we consider to be more relevant.

4.1 Lexical Diversity

The chart in Figure 11 (c.f. Annexes) gives an overall idea of how many different words reviewers use in their reviews. From the chart, we highlight that more than 1.2 million reviews (around 40% of the whole data set) have a lexical diversity ratio between 65% and 75%, meaning most words in the review text field were only written once.

4.2 Most Common Words

Another useful analysis we found useful was to explore what were the most common words that were written in the whole data set (Figure 12, c.f. Annexes). In line with out expectations, "book" is, in fact, the most frequent word in the database - more than 5 million repetitions -, followed by "read", "one", "story", and, closing the top five most common words, "like".

4.3 Word Count Distribution in Reviews

As the Figure 13 (c.f. Annexes) chart demonstrates, most reviews contain between 50 and 100 words (around 25% of all reviews). A closer look at the first four intervals (Figure 14) (c.f. Annexes) allows us to see that the most frequent number of words in reviews is around 25 words. However, it came to our attention the fact that there is a significant amount of reviews that contain more than 400 words.

4.4 Number of Sentences Distribution in Reviews

Lastly, in Figure 15 (c.f. Annexes), we plotted the distribution of the number of sentences written in the reviews. As the figure shows, around 14% of the reviews are made up of roughly 3 sentences, being this the most common sentence size in the data set.

5 PROSPECTIVE SEARCH TASKS

5.1 Description

The objective of the project is to provide the user with an advanced book search so that they can find the reading they are looking for based on several factors. As such, the user must be able to find books not only by their title, but also through their author, publisher, category, theme, reviews and ratings, all combined. An example of a possible search would be science fiction books that do not address the topic of space.

5.2 Information needs

To perform this search, all the information contained in the files obtained after cleaning the data from Kaggle website will be necessary. Thus, we will have at our disposal technical information about the books (such as title, author, publisher, publication date, category, description) contained in the books.csv file and also reviews and ratings of these in the reviews.csv file.

Based on this, some examples of these queries are:

- Retrieve books that talk about artificial intelligence.
- Retrieve books that support fights against racism.
- Retrieve novels set in post-apocalyptic worlds.
- Retrieve books related to American Civil War.

6 INFORMATION RETRIEVAL

In the pursuit of effective information retrieval for our book search system, the choice of the right tool is paramount. This section delves into the tool we've employed and the methodologies adopted to ensure accurate and efficient retrieval of information from our extensive data set.

6.1 Information Retrieval Tool

We opted for Apache Solr [5] as our information retrieval tool due to its robust features and capabilities in handling large datasets. Apache Solr is an open-source search platform built on Apache Lucene, offering advanced search and indexing functionalities. Some of the key features and advantages of using Solr in our project include:

- **Full-Text Search:** Solr provides powerful full-text search capabilities, allowing us to search through extensive book data efficiently.
- **Customizable Ranking:** Solr allows the customization of ranking strategies, ensuring that search results are relevant and aligned with user preferences.
- **Rich Query Language:** Solr offers a powerful and expressive query language that supports complex queries, enhancing our ability to retrieve specific information from the dataset.
- **Schema Flexibility:** Solr's schema design allows us to tailor the data structure to our specific requirements, accommodating various fields and data types present in our book dataset.

6.2 Document Analysis

Our collection is made up of several documents that, in our context, represent a book. It serves as the focal point for information retrieval, containing a variety of fields that capture diverse aspects of a book's metadata and content.

To enhance the search experience and provide users with comprehensive information about books rather than isolated reviews, we adopted a strategy to aggregate both the textual reviews and summaries into their corresponding books. This aggregation process ensures that queries return books as primary results, encapsulating the collective insights from both reviews and summaries. By combining these textual elements, our retrieval system aims to present a holistic view of each book, allowing users to make informed decisions based on a consolidated representation of reviews and content summaries.

6.3 Index Building Process

In the index building process, we defined a schema with specific filters [7] and tokenizers [8] to enhance the accuracy of our information retrieval system. Based on the characteristics of our data set, where most fields are text based (only the published year and the unique identifier of the book are numbers) we decided to create our custom field types as follows:

1. Name: text
Solr Field Type: TextField
Index Analyzer:

- Tokenizer:
  - StandardTokenizer

- Filters:
  - ASCIIFolding
  - LowerCase

Query Analyzer:

- Tokenizer:
  - StandardTokenizer
- Filters:
  - ASCIIFolding
  - LowerCase

2. Name: int
Solr Field Type: TrieIntField
Index Analyzer: N/A
Query Analyzer: N/A

The "text" field type in Solr incorporates ASCII folding and lowercasing during both indexing and query processing [4], enhancing the robustness of text-based searches. This configuration ensures that search operations are case-insensitive and can handle diacritics and special characters effectively. Additionally, the "text" field type utilises the standard tokenizer during both indexing and query processing. The standard tokenizer breaks the text into words, removing punctuation and providing a foundational step in the text analysis process. Together with ASCIIFolding and LowerCase filters, the "text" field type in Solr is well-equipped for handling diverse textual data in a search context.

Bellow is a table with the matching between the field in each 'book' document and the custom created field types:

Field	Type	Indexed	Multivalued
book_id	int	true	false
book_title	text	true	false
description	text	true	false
authors	text	true	true
publisher	text	true	false
categories	text	true	true
publishedYear	int	true	false
reviews	text	true	true
summary	text	true	false
text	text	true	false

Table 1. Solr Schema Fields

The Solr schema, outlined in Table 1, defines the important details for each book in our system. Despite being a space consuming decision, we opted for the indexation of all fields as it would, on the other hand, allow for faster searching processes. The 'Multivalued' column specifies if the field is an array with more than one value inside it or not.

6.4 Query Execution

After successfully indexing our collection and refining our information needs, we formulated specific queries to address distinct aspects of our search requirements. The choice of queries aligns with our defined information needs, aiming to test the effectiveness and relevance of our information retrieval system. Each query is tailored to retrieve books based on a particular theme or topic.

Here are the formulated queries corresponding to our redefined information needs:

1. **Retrieve books that talk about artificial intelligence:**

```
q = artificial intelligence
```

2. **Retrieve books that support fights against racism:**

```
q = fight fights against racism discrimination
```

3. **Retrieve novels set in post-apocalyptic worlds:**

```
q = novel AND post-apocalyptic world
```

4. **Retrieve books related to the American Civil War:**

```
q = "american civil war"
```

To enhance the precision of our queries, we utilised the Extended DisMax (eDisMax) query parser (`defType=edismax`) [6]. The eDisMax query parser in Solr is an extended version of the DisMax parser, offering enhanced features for better control over query parameters. It allows for a more flexible and customise approach to query parsing, enabling us to specify multiple fields for querying (`qf`) and assigning different weights to each field. Utilising the eDisMax query parser ensures that the retrieved books closely align with the information needs, providing a more effective and relevant search experience.

The parameters `q` and `qf` in the eDisMax setup play a crucial role. The `q` parameter specifies the user's query, and the `qf` parameter designates the fields to be queried and their respective weights. By adjusting these parameters, we can fine-tune the search process, considering the importance of each field in the retrieval process.

7 EVALUATION

In this section, we evaluate the performance of our information retrieval systems, considering different retrieval setups and metrics. We implemented two distinct retrieval configurations, the Base System and the Enhanced System.

The evaluation process encompasses a detailed relevance assessment, involving manual scrutiny of each retrieved document to ensure alignment with the specified queries.

For a comprehensive evaluation, we employ key metrics, including Average Precision, Precision @ 10, and Precision-Recall Curve, to gauge the effectiveness of our system in meeting the defined information needs. The subsequent sections provide a detailed analysis of the results obtained for each information need, offering insights into the precision and relevance of the retrieved documents.

7.1 Relevance Evaluation Process

Relevance evaluation involved a demanding process, starting with the retrieval of the first 40 documents from the Solr interface based on the specified queries. Each document was then manually analysed to assess its relevance in the scope of the query and the defined

information needs. This scrutiny considered various factors, including the title and the content of reviews, summaries, and book descriptions.

To supplement the manual evaluation, external sources such as Google were consulted to gather additional context and detailed information, especially in cases where the reviews or book descriptions were ambiguous or lacked clarity. The combination of manual analysis and external validation ensured a thorough and reliable assessment of the relevance and precision of the retrieved documents.

7.2 Retrieval Setups

For our information retrieval system, we implemented two distinct retrieval setups to evaluate their impact on the search results:

Base System:

```
qf = book_title reviews.text reviews.summary
description categories
```

In the base system, we assigned equal weights to key fields, including book title, reviews, summaries, and descriptions. This setup aimed to provide a balanced approach to document scoring, considering various aspects of a book's metadata and content.

Enhanced System:

```
qf = book_title^2 reviews.text^2 reviews.summary
description categories
```

Both systems have as backbone the same schema (as detailed in Section 6.3). However, in the enhanced system, we introduced different weightings to the book title and reviews, placing higher importance on these fields by using squared weights (^2). This setup was designed to amplify the influence of textual content, specifically book titles and reviews, in the scoring process. The goal was to investigate whether emphasising certain fields would lead to more precise and relevant search results.

These retrieval setups served as the foundation for our evaluation, allowing us to compare the performance and effectiveness of the base and enhanced systems in our problem context.

7.3 Results Evaluation Overview

In our evaluation, we employ 3 key metrics to assess the performance of our information retrieval system. These metrics offer quantitative insights into the system's effectiveness in meeting user information needs. Here's a brief overview of the metrics used:

- **Average Precision (AP):** This metric computes the average precision across all relevant documents retrieved by the system, offering an aggregate measure of precision. It indicates the system's ability to consistently rank relevant documents higher in the result set.
- **Precision @ 10 (P@10):** Precision at 10 evaluates the precision of the system by focusing on the top 10 retrieved documents. This metric provides insight into the system's immediate performance in presenting relevant results within the initial subset of the ranking.
- **Precision-Recall Curve:** The precision-recall curve depicts the relationship between precision and recall across various decision thresholds. It serves as a graphical representation of the system's effectiveness in balancing precision and recall, offering insights into its performance under different retrieval scenarios.

These metrics collectively offer a comprehensive evaluation of our information retrieval system, allowing us to gauge its precision, relevance, and overall effectiveness in fulfilling user information needs.

- **Information Need 1:** Books about Artificial Intelligence
  Query:
  `q = artificial intelligence`
  Metrics Comparison:

System	Average Precision	Precision at 10 (P@10)
Base System	0.759010	0.700000
Enhanced System	0.858932	0.900000

Table 2. Metrics Comparison for Books about Artificial Intelligence

  Results Description: The enhanced system does better than the base system in retrieving books related to artificial intelligence, as indicated by a higher average precision. The enhanced system can maintain precision at 1.0 for higher recall values than the base system before dropping from that level (c.f. Table 2). However, at a recall of about 0.60, the precision of the enhanced system drops sharply to levels lower than those of the base system. The precision-recall curve in Fig. 16 in the annexes is a visual illustration of this description.
- **Information Need 2:** Books that support fights against racism
  Query:
  `q = fight fights against racism discrimination`
  Metrics Comparison:

System	Average Precision	Precision at 10 (P@10)
Base System	0.658527	0.600000
Enhanced System	0.707906	0.700000

Table 3. Metrics Comparison for Books that support fights against racism

  Results Description: The enhanced system outperforms the base system in retrieving books related to to fights against racism, as indicated by higher Average Precision. The enhance system is able to maintain precision equal to 1.0 for overall values of recall than the base system (c.f. Table 3). However, at around the [0.1;0.15] and [0.50;0.60] intervals of recall values, the precision of the enhanced system equals that of the base system. The Precision-Recall Curve in Fig. 17 in the annexes visually illustrates this description.
- **Information Need 3:** Novels in Post-Apocalyptic Worlds
  Query:

`q = novel AND post-apocalyptic`

Metrics Comparison:

System	Average Precision	Precision at 10 (P@10)
Base System	0.821265	0.700000
Enhanced System	0.759569	0.700000

Table 4. Metrics Comparison for Novels in Post-Apocalyptic Worlds

Results Description: The base system's precision sharply declines at around 0.05 recall (c.f. Table 4), falling below the precision values of the enhanced system. The enhanced system notably maintains higher precision for increased recall values compared to the base system. This trend is visually depicted in Fig. 18 in the annexes.

- **Information Need 4:** Books about the American Civil War
  Query:

  `q = "american civil war"`

  Metrics Comparison:

System	Average Precision	Precision at 10 (P@10)
Base System	0.982544	1.000000
Enhanced System	0.968378	1.000000

Table 5. Metrics Comparison for Books about the American Civil War

Results Description: Considering that the query performs an exact search, as evidenced by the quotation marks (q = "american civil war"), this example presents more accurate results. However, in this case, we notice that the base system is better in the retrieval of books related to the American Civil War than the enhanced system, since its Average Precision is higher (c.f. Table 5), although the Precision @ 10 values are the same. This happens because, in some reviews, critics mention the "American Civil War" to locate the book in a timeline or to comment on something related to that topic. The *boost*, being applied to *reviews.text*, ends up giving priority to these results in the search, despite them not being directly related to what is intended. We can clearly see in the Precision-Recall Curve in Fig. 19 present in the annexes the differences between both performances.

8 IMPROVED INFORMATION RETRIEVAL SYSTEMS

In this phase of our project, we aimed to enhance the existing information retrieval system by introducing various improvements. We decided to build upon the Enhanced System from the previous sections, incorporating the following new features and filters to optimise the search experience. That is, each process described below was incorporated on top of the enhanced system individually, not in a cumulative way.

Overview of Improvements

- **Edge NGram and Porter Stem Filters**
- **Synonym Expansion**
- **Keyword Extraction with KeyBERT**
- **Semantic Search with Vector Embeddings**
- **User Interface Development**

8.1 Edge NGram and Porter Stem Filters

This section introduces the Edge NGram and Porter Stem filters [7], outlining their functionalities and the anticipated outcomes when applied to the Solr search system.

Edge NGram Filter: The Edge NGram filter is designed to break down words into smaller units, creating a set of sub sequences of a token known as "n-grams." Specifically, it generates n-grams starting from the beginning (edge) of the word. The parameters `minGramSize` (set to 3) and `maxGramSize` (set to 15) define the minimum and maximum lengths of the generated n-grams. For example, with a word like "search," the Edge NGram filter will produce the following n-grams "sea", "sear", "searc" and "search".

Porter Stem Filter: The Porter Stem filter is a linguistic normalisation technique that reduces words to their base or root form, known as the "stem". It helps in consolidating different forms of a word to a common base, increasing the system's ability to match variations of a term. For instance, words like "running" and "runner" have the same stem "run".

Expected Impact on Search Results: The application of the Edge NGram and Porter Stem filters is expected to have the following outcomes:

- **Edge NGram Filter:** By breaking down words into smaller units, this filter facilitates partial and wildcard searches. In theory, users should obtain relevant results even when entering partial terms, enhancing the system's flexibility in handling user queries.
- **Porter Stem Filter:** Normalising words to their base forms ensures that variations of a term are treated as the same, improving recall in the search process. Users can find documents containing different forms of a word more effectively.

These filters contribute to a more robust and user-friendly search experience by addressing common challenges associated with partial term matching and linguistic variations.

8.2 Synonym Expansion

This section delves into the concept of synonym expansion within the Solr search system, providing an overview of the tools employed, namely WordNet [10] and the Datamuse API [3]. Additionally, it explores the methodology used for extracting synonyms and evaluates the subsequent impact on search results.

Synonym Extraction Tools:

- WordNet (NLTK): A lexical database of the English language providing synonyms and relationships between words.
- Datamuse API: An API that returns a set of relevant words based on queries, allowing us to dynamically fetch synonyms.

Synonym Extraction Methodology: We used a Python notebook to interact with this tools. For the purpose of our work, we limited the set of synonyms to the be acquired based on the words we were using in our queries to make the whole processing faster. We obtained the set of synonyms from each of the tools mentioned above and then concatenated them into a single set. We create a *synonyms.txt* file were one line contained the synonyms for each word processed. We later added the SynonymGraphFilterFactory Filter to the schema.

Expected Impact on Search Results: The incorporation of synonym expansion into the Solr search system was expected to enhance the system's recall by including documents that may contain synonymous terms to the ones given in the query. This improvement is particularly valuable in addressing variations in user queries and accommodating a wider range of vocabulary.

8.3 Keyword Extraction with KeyBERT

KeyBERT [9] is a Python module that utilises BERT embeddings to extract keywords from a given text. This section provides an overview of KeyBERT, the process of extracting keywords, and the impact on search relevance.

KeyBERT leverages BERT (Bidirectional Encoder Representations from Transformers) embeddings, which capture contextual information and semantic meaning in the text. This enables KeyBERT to identify and rank keywords based on their importance in the context of the entire document.

Keyword Extraction Process: The process of extracting keywords using KeyBERT involved the following steps:

(1) Installation: KeyBERT can be installed using Python package managers like pip.
(2) Initialisation: Initialise the KeyBERT model.
(3) Text Representation: Convert the input text (reviews.text field in our case) into vectorized representations using BERT embeddings.
(4) Keyword Extraction: Extract keywords from the vectorized representations, considering their importance in the context.

Impact on Search Relevance: The schema of the Solr index was updated to include a new field named *keywords* where the top 10 keywords extracted using KeyBERT are stored for each book. To this new field is assigned a weight of 5 in the query configuration field `qf`, indicating its increased importance in the search process.

8.4 Semantic Search with Vector Embeddings

This section delves into the implementation of semantic search within the Solr search system through the incorporation of vector embeddings. It provides comprehensive details on the process of generating vector embeddings and elucidates the role of the newly introduced 'vector' field in enhancing semantic search capabilities.

Generation of Vector Embeddings: Vector embeddings were generated for the selected fields — *book_title*, *reviews.text*, *authors*, *reviews.summary*, *description*, and *categories*. The embedding process involved utilising the SentenceTransformer library, specifically employing the 'all-MiniLM-L6-v2' model.

The SentenceTransformer [11] library facilitates the transformation of textual data into dense vector representations, capturing the semantic relationships between different pieces of text. The selected model, 'all-MiniLM-L6-v2,' is known for its efficiency and effectiveness in generating meaningful embeddings for various natural language processing tasks.

Addition of the vector Field. To leverage the generated vector embeddings for semantic search, a new field named *vector* was introduced to the Solr schema. This field serves as a repository for the dense vector representations associated with each document. The schema was accordingly updated to incorporate this *vector* field.

Expected Impact on Search Results: The addition of the *vector* field contributes to the improvement of semantic search capabilities within the Solr system. By capturing the semantic nuances of different textual elements, the vector embeddings enable more nuanced and context-aware search results. Users can expect a more refined search experience that goes beyond traditional keyword matching, allowing for the retrieval of documents with similar semantic meanings.

The adoption of vector embeddings aligns with contemporary advancements in natural language processing, enhancing the overall effectiveness of the Solr search system in delivering relevant and contextually rich search results.

8.5 User Interface Development

The development of an enhanced user interface utilising Vue.js[13] in conjunction with FastAPI[12] aimed at optimising the user experience within the book search system while seamlessly interfacing with the Solr API. This section illuminates the design and implementation aspects of this user interface, emphasising its pivotal role in enhancing accessibility and usability.

The user interface predominantly centres around Vue.js, a dynamic JavaScript framework, and FastAPI, a robust web framework. Vue.js acts as the primary tool for crafting visually intuitive pages, presenting vital book-related information to users. Meanwhile, FastAPI operates as a streamlined server, effectively managing requests directed towards the Solr API.

At the forefront of the interface lies the home page, serving as the entry point to the system (c.f. Annexes Figure 20). Featuring a simple and efficient search bar powered by Vue.js, users seamlessly input queries. Query results promptly display as a concise book list, ensuring swift and efficient browsing (c.f. Annexes Figure 21). Upon selecting a book from this list, users access detailed book information through a popup page, designed using Vue.js for uninterrupted exploration (c.f. Annexes Figure 22). FastAPI acts as an intermediary server, simplifying interactions between the frontend and the Solr API, ensuring a smooth and straightforward querying process.

Note on Data Reduction

In implementing the keyword extraction strategy, we acknowledge that we worked with a reduced sample size of the original data. Despite the potential impact on results, this decision was made to facilitate processing on our devices. The reduction from 212,404 books to 5,000 (2.35%) and from 3,000,000 reviews to 79,363 (2.65%)

allowed us to gain experience with these method within resource constraints. Readers should consider this factor when interpreting the search outcomes.

9 NEW SYSTEMS EVALUATION

In this section, we assess the performance and effectiveness of the newly implemented information retrieval systems. The evaluation is based on the same key metrics as in the Section 7., providing a quantitative analysis of the systems' ability to meet specific information needs. The results offer insights into the impact of each enhancement on precision and recall, contributing to a comprehensive understanding of the strengths and potential areas for improvement in the newly developed systems. We added a precision-recall curve for each information need where the systems' performance can be compared between each other and the Enhanced System from Section 7. We also provide a Rank-Relevance Table to support our key metrics' results for each information need in the annexes section of the document (tables 17 to 20).

The same process as the one described in section 7.1 for evaluation of results' documents relevance was used.

9.1 Retrieval Setups

The modifications described in Section 8. require the definition of new retrieval setups:

New Schema System:

```
qf = book_title^2 reviews.text^2 reviews.summary
description categories
```

Additions to the schema:

1. Name: text
Solr Field Type: TextField
Index & Query Analyzers:

- Filters:
  - EdgeNGramFilterFactory
  - PorterStemFilterFactory

Synonyms System:

```
qf = book_title^2 reviews.text^2 reviews.summary
description categories
```

Additions to the schema:

1. Name: text
Solr Field Type: TextField
Index & Query Analyzer:

- Filters:
  - SynonymGraphFilterFactory

Keywords System:

```
qf = book_title^2 reviews.text^2 reviews.summary
description categories reviews^5
```

Additions to the schema:

In the system, we added a new field *keywords* of type text so as to apply to it the same filters as the other fields. To the `qf` parameter of the queries, we added the field *reviews* with a weight of 5 to give a bigger importance to this field than all the others.

Field	Type	Indexed	Multivalued
keywords	text	true	false

Table 6. Solr Schema Fields

Embeddings System:

```
qf = book_title^2 reviews.text^2 reviews.summary
description categories
```

Additions to the schema:

1. Name: vector
Solr Field Type: DenseVectorField
Vector Dimension: 384
Similarity Function: *Cosine*
KNN Algorithm: HNSW

Field	Type	Indexed	Stored
vector	vector	true	true

Table 7. Solr Schema Fields

9.2 Results Evaluation Overview

- **Information Need 1:** Books about Artificial Intelligence
  Query:
  `q = artificial intelligence`
  Metrics Comparison:

System	Average Precision	Precision at 10 (P@10)
Enhanced System	0.858932	0.900000
New Schema System	0.777723	0.800000
Synonyms System	0.000000	0.000000
Keywords System	0.565476	0.400000
Embeddings System	0.774552	0.700000

Table 8. Metrics Comparison for Books about Artificial Intelligence

Results Description: We can see that the performance of the evaluated systems in retrieving books related to artificial intelligence varies. The Enhanced System and New Schema System performed well with high precision scores (0.859 and 0.778 Average Precision, respectively) (c.f. Table 8). The New Schema System looses a bit of performance related to the Enhanced System. We believe this is due to the Edge NGram Filter: there were a lot of results that are related to art and that talk about artists. Solr breaks "artificial" in to an n-grams, one of them being "art". The Synonyms System performed poorly (0 Average Precision) due to the fact that synonyms cover a greater number of different results in their search.

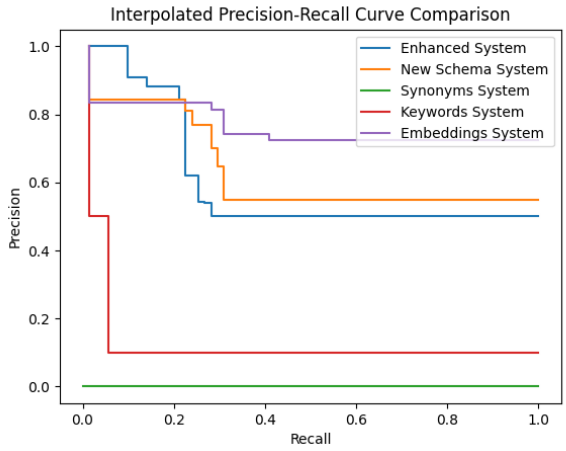

Fig. 3. Precision-Recall Curve for Books about Artificial Intelligence

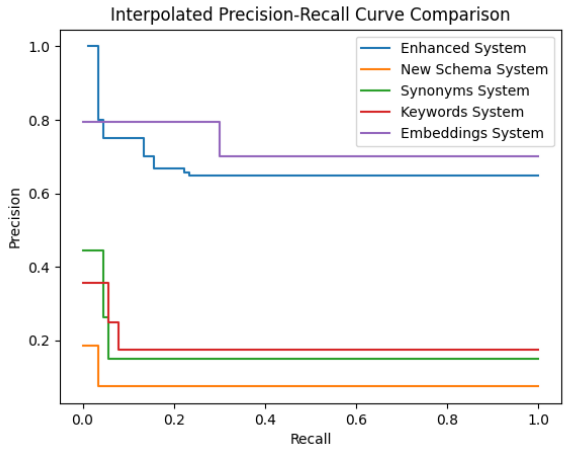

Fig. 4. Precision-Recall Curve for Books that support fights against racism

For instance we noticed a most of results were related to governmental intelligence services (like the CIA, MI5 or KGB). The Keywords System was very precise for low recall values however the curve sharply declines to values of around 0.1 for recall values greater than approximately 0.05. The Embeddings System showed moderate to high performance. Fig. 3 visually illustrates this description.

- **Information Need 2:** Books that support fights against racism
  Query:
  `q = fight fights against racism discrimination`
  Metrics Comparison:

System	Average Precision	Precision at 10 (P@10)
Enhanced System	0.707906	0.700000
New Schema System	0.140422	0.000000
Synonyms System	0.303251	0.400000
Keywords System	0.254040	0.200000
Embeddings System	0.700967	0.600000

Table 9. Metrics Comparison for Books that support fights against racism

Results Description: In the context of retrieving books on fighting against racism, Embeddings System performed well with Average Precisions of 0.701 and P@10 scores 0.6 (c.f. Table 9. The Synonyms System had moderate performance with an Average Precision of 0.303 and a P@10 score of 0.4. We noticed this system returned a lot of results that were related to wars. We confirmed that "war" as one of the synonyms retrieved for the word "fight". The Keywords System showed room for improvement, achieving an Average Precision of 0.254 and a P@10 score of 0.2. The New Schema System struggled, with an Average Precision of 0.140 and a P@10 score of 0. Fig. 4 visually illustrates this description.

- **Information Need 3:** Novels in Post-Apocalyptic Worlds
  Query:
  `q = novel AND post-apocalyptic`
  Metrics Comparison:

System	Average Precision	Precision at 10 (P@10)
Enhanced System	0.845534	0.800000
New Schema System	0.716314	0.700000
Synonyms System	0.616292	0.600000
Keywords System	0.805556	0.300000
Embeddings System	0.529027	0.600000

Table 10. Metrics Comparison for Novels in Post-Apocalyptic Worlds

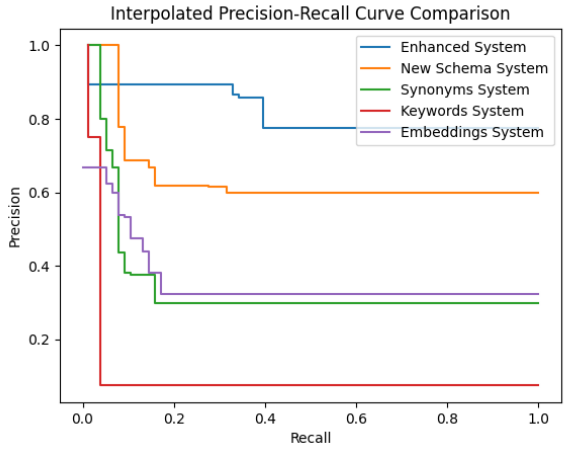

Fig. 5. Precision-Recall Curve for Novels in Post-Apocalyptic Worlds

Results Description: The New Schema System also showed decent performance with an Average Precision of 0.716 and a P@10 score of 0.7 as stated in Table 10. The Synonyms System achieved moderate performance with an Average Precision of 0.616 and a P@10 score of 0.6, suggesting it provided some relevant recommendations. The Keywords System had a relatively high Average Precision of 0.806 but a lower P@10 score of 0.3, indicating potential room for improvement in retrieving the most relevant books within the top 10 results. The Embeddings System had an Average Precision of 0.529 and a P@10 score of 0.6, suggesting it performed moderately but could also benefit from some refinement. We can clearly see in the Precision-Recall Curve in Fig. 5 the differences between this systems performances.

- **Information Need 4:** Books about the American Civil War
  Query:

  ```
  q = "american civil war"
  ```

  Metrics Comparison:

System	Average Precision	Precision at 10 (P@10)
Enhanced System	0.968378	1.000000
New Schema System	1.000000	1.000000
Synonyms System	1.000000	1.000000
Keywords System	0.385366	0.400000
Embeddings System	0.931871	1.000000

Table 11. Metrics Comparison for Books about the American Civil War

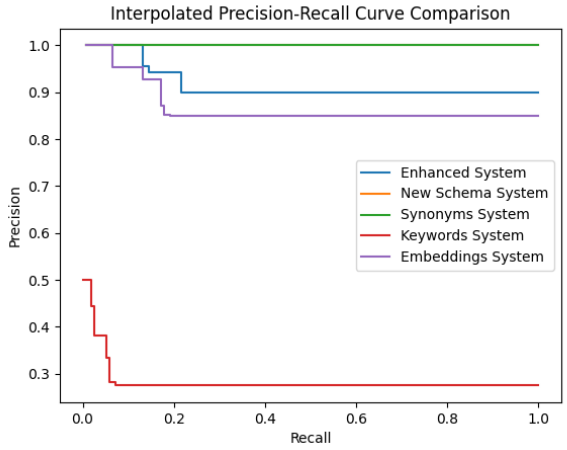

Fig. 6. Precision-Recall Curve for Books about the American Civil War

Results Description: The New Schema System, and Synonyms System all demonstrated perfect performance with an Average Precision and Precision at 10 (P@10) score of 1.000 as shown in Table 11. This indicates that these systems were highly effective in recommending relevant books on the American Civil War. The Keywords System had a lower Average Precision of 0.385 and a P@10 score of 0.4, suggesting that it struggled to provide highly precise recommendations within the top 10 results for this specific topic. The Embeddings System had a very high Average Precision of 0.932 and a perfect P@10 score of 1.000, indicating strong performance in retrieving relevant books related to the American Civil War. We can clearly see in the Precision-Recall Curve in Fig. 6 the differences between this systems performances.

Note on Data Reduction Effects:

The decision to work with a reduced sample size of the original data (from 212,404 books to 5,000, approximately 2.35%) was made to accommodate processing constraints on our devices. This reduction, however, may have contributed to lower-than-expected results.

It is likely, but not definitively certain, that the data reduction had a negative impact on the effectiveness of the keyword extraction system. The observed decrease in performance should be interpreted in light of this reduction in the dataset. Future evaluations with a larger dataset could provide more insights into the system's capabilities when operating with a fuller set of data.

10 COMBINATION OF IMPROVEMENTS

Following the evaluation of various setups and a thorough analysis of metric results, we sought to explore the performance of a combined approach. Thus, we introduced a Combined System that amalgamates the Enhanced System, the New Schema System, and the Synonyms System. This integration aims to leverage the individual merits of each system, offering a holistic and potentially synergistic enhancement to our information retrieval capabilities.

10.1 Retrieval Setup

Combined System:

```
qf = book_title^2 reviews.text^2 reviews.summary
description categories
```

Schema Text Field :
1. Name: text
Solr Field Type: TextField
Index & Query Analyzer:

- Tokenizer:
  - StandardTokenizer
- Filters:
  - ASCIIFolding
  - LowerCase
  - PorterStemFilterFactory
  - EdgeNGramFilterFactory
  - SynonymGraphFilterFactory

Query Analyzer:
The schema of this new system reflects and incorporates the changes from the 3 mentioned systems.

10.2 Results Evaluation Overview

After having completed the setup of the new system, we analysed and plotted the key metrics to compared the Enhanced System

(comparison basis), the New Schema System, the Synonyms System and the Combined System.

- **Information Need 1:** Books about Artificial Intelligence
  Query:
  `q = artificial intelligence`
  Metrics Comparison:

System	Average Precision	Precision at 10 (P@10)
Enhanced System	0.858932	0.900000
New Schema System	0.777723	0.800000
Synonyms System	0.000000	0.000000
Combined System	0.000000	0.000000

Table 12. Metrics Comparison for Books about Artificial Intelligence

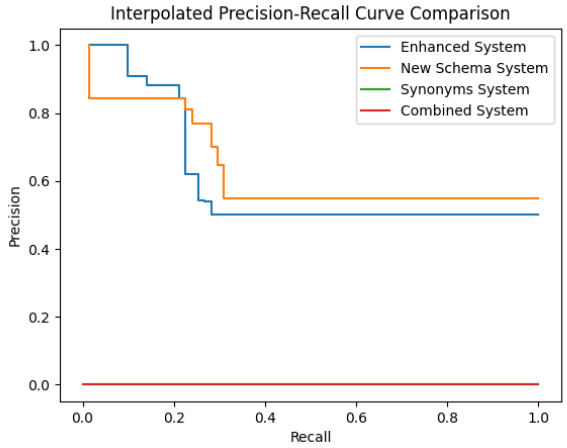

Fig. 7. Precision-Recall Curve for Books about Artificial Intelligence

- **Information Need 2:** Books that support fights against racism
  Query:
  `q = fight fights against racism discrimination`
  Metrics Comparison:

System	Average Precision	Precision at 10 (P@10)
Enhanced System	0.707906	0.700000
New Schema System	0.140422	0.000000
Synonyms System	0.303251	0.400000
Combined System	0.303251	0.400000

Table 13. Metrics Comparison for Books that support fights against racism

- **Information Need 3:** Novels in Post-Apocalyptic Worlds
  Query:
  `q = novel AND post-apocalyptic`

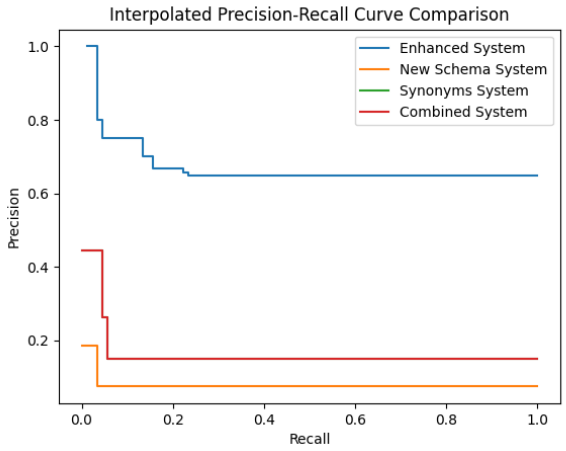

Fig. 8. Precision-Recall Curve for Books that support fights against racism

System	Average Precision	Precision at 10 (P@10)
Enhanced System	0.845534	0.800000
New Schema System	0.716314	0.700000
Synonyms System	0.616292	0.600000
Combined System	0.616292	0.600000

Table 14. Metrics Comparison for Novels in Post-Apocalyptic Worlds

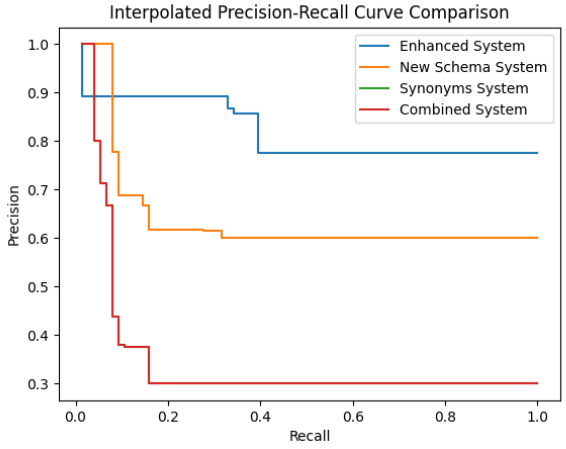

Fig. 9. Precision-Recall Curve for Novels in Post-Apocalyptic Worlds

Metrics Comparison:

- **Information Need 4:** Books about the American Civil War
  Query:
  `q = "american civil war"`
  Metrics Comparison:

System	Average Precision	Precision at 10 (P@10)
Enhanced System	0.968378	1.000000
New Schema System	1.000000	1.000000
Synonyms System	1.000000	1.000000
Combined System	1.000000	1.000000

Table 15. Metrics Comparison for Books about the American Civil War

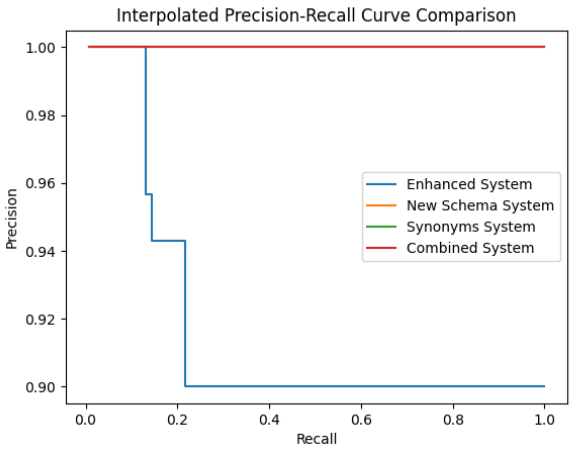

Fig. 10. Precision-Recall Curve for Books about the American Civil War

10.3 Conclusion of the results

After carefully analysing the results, we conclude that the combined system is clearly dependent on the implementation of the synonyms filter. For all information needs, the average precision and P@10 values are always the same in both systems.

11 MEAN AVERAGE PRECISION

Mean Average Precision calculates the average value of the Average Precision Values of each system in all information needs. It is a good metric for comparing systems performances. From the table 16, we can clearly see that the best of the tested systems is the Enhanced System with a mean average precision of around 0.845. Next is the base system and the last place is shared by both the Synonyms System and the Combines System.

System	Mean Average Precision
Enhanced System	0.845187
Base System	0.811680
Embeddings System	0.734105
New Schema System	0.658615
Keywords System	0.502610
Synonyms System	0.479886
Combined System	0.479886

Table 16. Mean Average Precision Per System

12 CONCLUSION

In the initial sections of the paper, regarding the Data Preparation phase, the processing pipeline is documented, as well as all the tools used, showing all the steps that led to the creation of a complete and coherent data set about books and their reviews. Some graphics and statistics are also present to better visualise the data. All of the goals for this phase were accomplished given that there is a better understanding of the chosen domain.

The creation of the information retrieval system involved selecting the Apache Solr search platform, which provides robust full-text search capabilities. The system's efficacy was assessed using two configurations: the Base System and the Enhanced System, each emphasising unique data aspects. Evaluation metrics including average precision, precision at 10 and precision-recall curve were used to provide a comprehensive assessment.

We later tried to optimise systems with features like Edge NGram and Porter Stem filters, synonym expansion, and semantic search. These tackled challenges related to user queries and content representation. The filters accommodated partial matching and linguistic variations. Synonym expansion revealed to have broadened search scope too much resulting in lower than expected performance. Keyword extraction and semantic search also score lower than expected results. An interface, built with Vue.js and FastAPI, was created for a better interaction with Solr. The overall results, mainly the Mean Average Precision metric placed the Enhanced System as the best among the rest of the tested system.

As the project comes to the end, we can certainly conclude than not always adding more features to a search system proves to be of value. However, a possible future work, we would possible enhance the UI with some Solr features like More Like This Suggestion or query Auto-Complete text suggestions.

REFERENCES

[1] [n.d.] *Datasets.* URL: https://www.kaggle.com/datasets.
[2] Mohamed Bekheet. *Amazon Books Reviews.* URL: https://www.kaggle.com/datasets/mohamedbakhet/amazon-books-reviews. (accessed: 29.09.2023).
[3] Datamuse. *Datamuse API.* URL: https://www.datamuse.com/api/. (last accessed: 13.12.2023).
[4] Apache Software Foundation. *Analyzers.* URL: https://solr.apache.org/guide/solr/9_3/indexing-guide/analyzers.html. (last accessed: 11.11.2023).
[5] Apache Software Foundation. *Apache Solr Reference Guide.* URL: https://solr.apache.org/guide/solr/9_3/index.html. (last accessed: 15.11.2023).
[6] Apache Software Foundation. *Extended DisMax (eDisMax) Query Parser.* URL: https://solr.apache.org/guide/solr/9_3/query-guide/edismax-query-parser.html. (last accessed: 11.11.2023).
[7] Apache Software Foundation. *Filters.* URL: https://solr.apache.org/guide/solr/9_3/indexing-guide/filters.html. (last accessed: 13.12.2023).
[8] Apache Software Foundation. *Tokenizers.* URL: https://solr.apache.org/guide/solr/9_3/indexing-guide/tokenizers.html. (last accessed: 11.11.2023).
[9] Maarten Grootendorst. *KeyBERT.* URL: https://github.com/MaartenGr/keyBERT?tab=readme-ov-file. (last accessed: 13.12.2023).
[10] NLTK Project. *Wordnet: Sample usage for wordnet.* URL: https://www.nltk.org/howto/wordnet.html. (last accessed: 13.12.2023).
[11] Nils Reimers. *Datamuse API.* URL: https://www.sbert.net/. (last accessed: 13.12.2023).
[12] FastAPI Development Team. *FastAPI.* URL: https://fastapi.tiangolo.com/. (last accessed: 14.12.2023).
[13] Vue.js Development Team. *Vue.* URL: https://vuejs.org/. (last accessed: 14.12.2023).

A ANNEXES

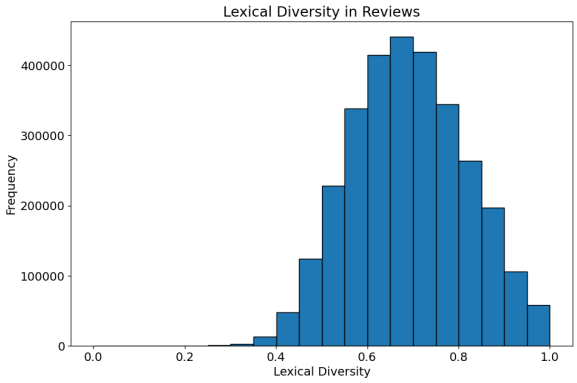

Fig. 11. Reviews' Lexical Diversity Distribution

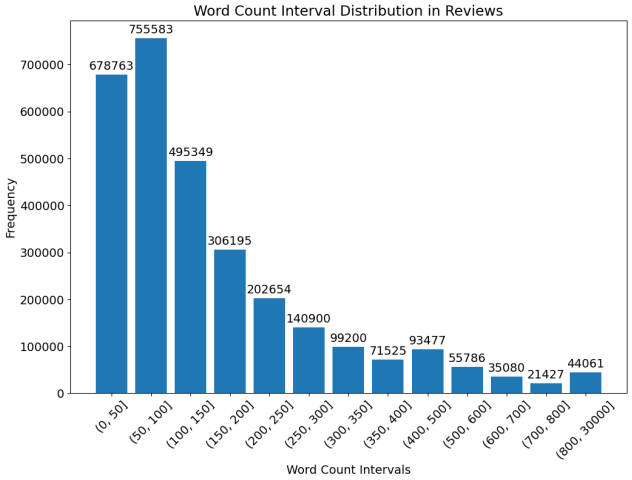

Fig. 13. Word Count Interval Distribution in Reviews

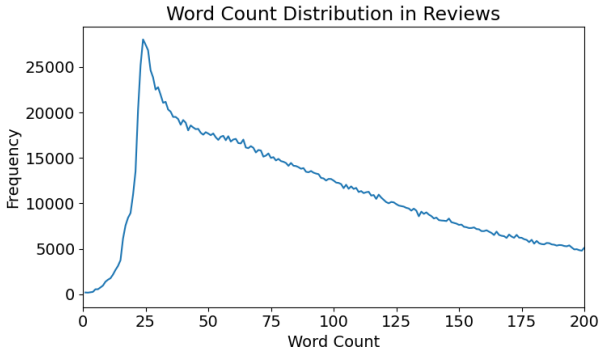

Fig. 14. Word Count Distribution in Reviews

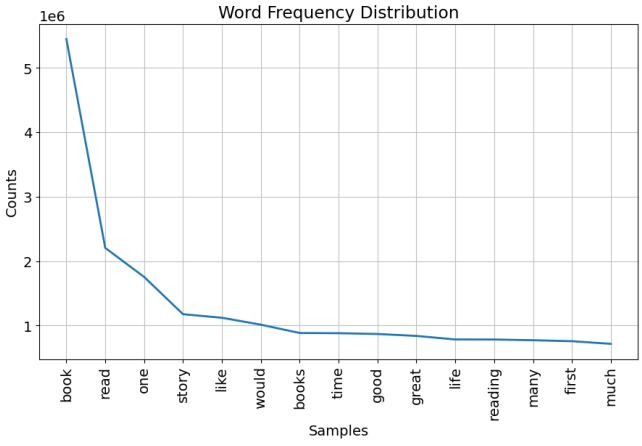

Fig. 12. Most Common Words

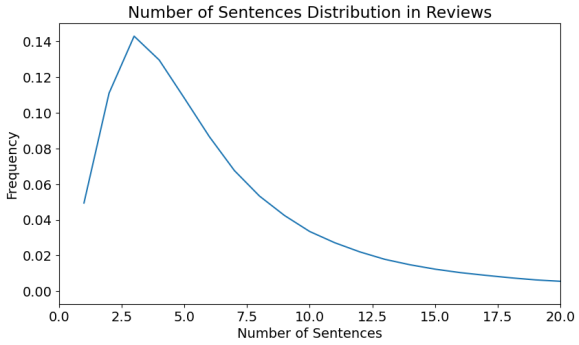

Fig. 15. Number of Sentences Distribution in Reviews

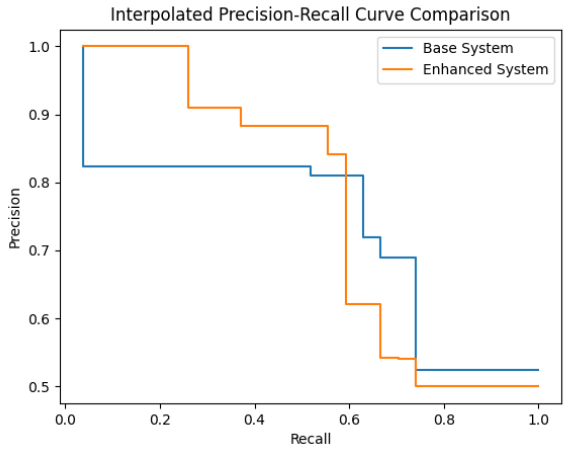

Fig. 16. Precision-Recall Curve for Books about Artificial Intelligence

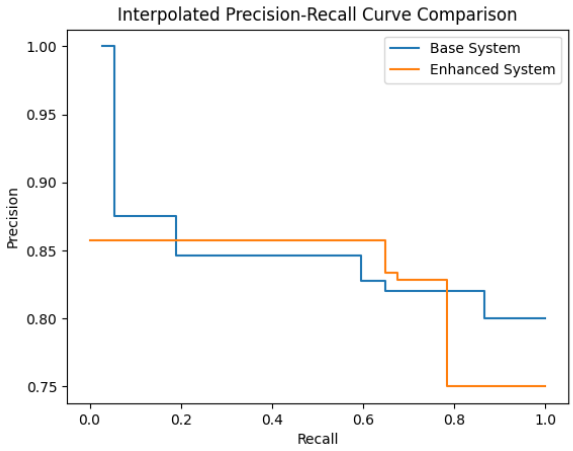

Fig. 18. Precision-Recall Curve for Novels in Post-Apocalyptic Worlds

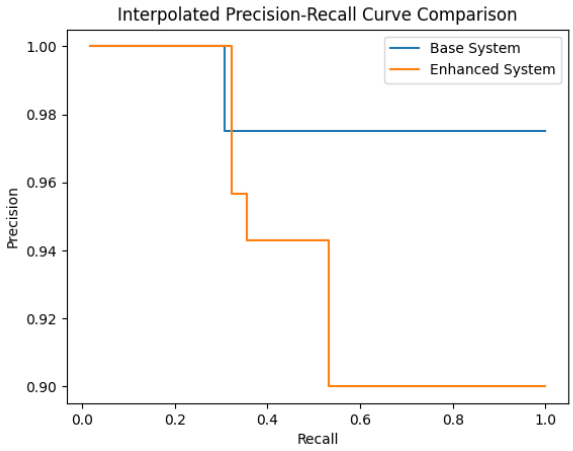

Fig. 19. Precision-Recall Curve for Books about the American Civil War

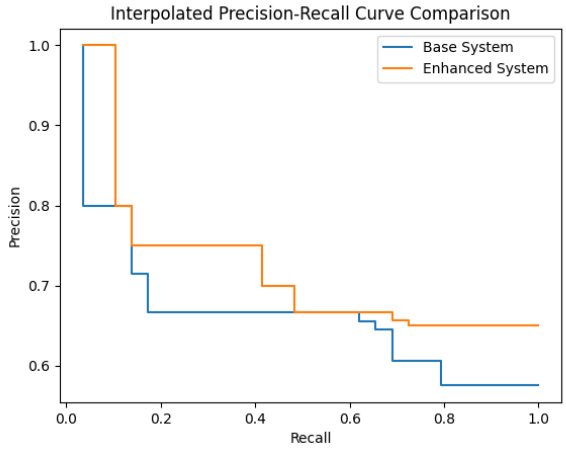

Fig. 17. Precision-Recall Curve for Books that support fights against racism

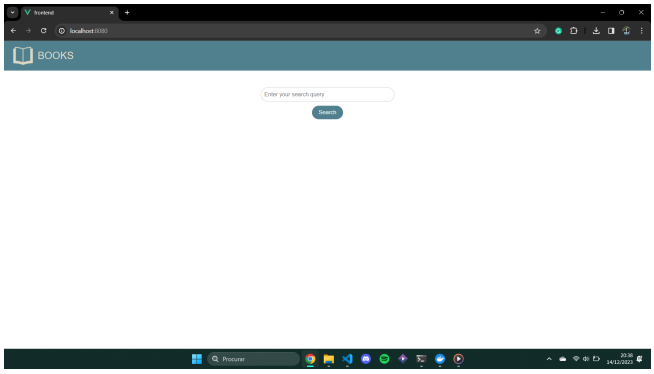

Fig. 20. Screenshot of User Interface Home Page

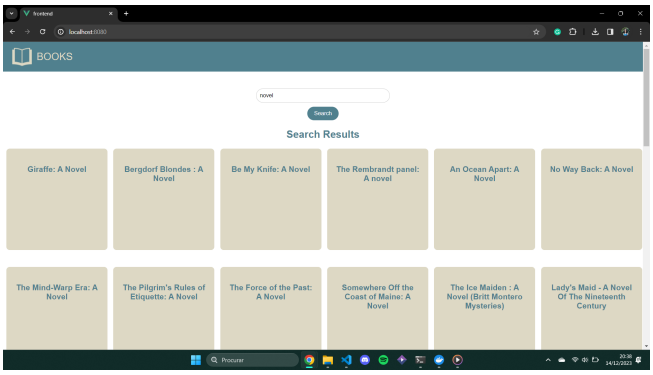

Fig. 21. Screenshot of User Interface Requested Books

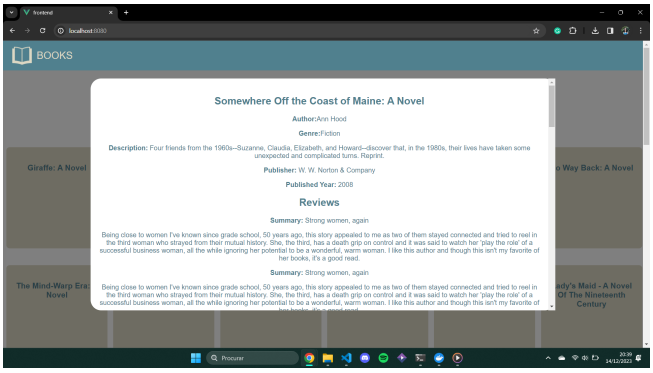

Fig. 22. Screenshot of User Interface Book Page

Rank	Base System	Enhanced System	New Schema System	Synonyms System	Keywords System	Embeddings System	Combined System
1	R	R	R	NR	R	R	NR
2	NR	R	NR	NR	NR	NR	NR
3	R	R	R	NR	NR	R	NR
4	R	R	R	NR	NR	R	NR
5	R	R	NR	NR	NR	R	NR
6	NR	R	R	NR	R	NR	NR
7	R	R	R	NR	R	R	NR
8	R	NR	R	NR	R	R	NR
9	R	R	R	NR	NR	NR	NR
10	NR	R	R	NR	NR	R	NR
11	R	R	R	NR	NR	R	NR
12	R	NR	R	NR	NR	R	NR
13	R	R	NR	NR	NR	R	NR
14	R	R	R	NR	NR	R	NR
15	R	R	R	NR	NR	R	NR
16	R	R	R	NR	NR	R	NR
17	R	R	R	NR	NR	R	NR
18	NR	NR	R	NR	NR	NR	NR
19	R	R	R	NR	NR	R	NR
20	R	NR	NR	NR	NR	R	NR
21	R	NR	R	NR	NR	R	NR
22	NR	NR	NR	NR	NR	R	NR
23	NR	NR	NR	NR	NR	R	NR
24	NR	NR	R	NR	NR	R	NR
25	R	NR	R	NR	NR	NR	NR
26	NR	NR	R	NR	NR	R	NR
27	NR	NR	NR	NR	NR	R	NR
28	R	R	NR	NR	NR	NR	NR
29	R	R	NR	NR	NR	NR	NR
30	NR	NR	R	NR	NR	NR	NR
31	NR	NR	NR	NR	NR	NR	NR
32	R	NR	NR	NR	NR	R	NR
33	NR	NR	NR	NR	NR	R	NR
34	NR	NR	R	NR	NR	R	NR
35	NR	R	NR	NR	NR	NR	NR
36	NR	NR	NR	NR	NR	R	NR
37	R	R	NR	NR	NR	R	NR
38	NR	NR	NR	NR	NR	R	NR
39	NR	NR	NR	NR	NR	R	NR
40	R	NR	NR	NR	NR	NR	NR

Table 17. Information Need 1 Relevance Table

Rank	Base System	Enhanced System	New Schema System	Synonyms System	Keywords System	Embeddings System	Combined System
1	R	R	R	NR	R	R	NR
2	NR	R	NR	NR	NR	NR	NR
3	R	R	R	NR	NR	R	NR
4	R	R	R	NR	NR	R	NR
5	R	R	NR	NR	NR	R	NR
6	NR	R	R	NR	R	NR	NR
7	R	R	R	NR	R	R	NR
8	R	NR	R	NR	R	R	NR
9	R	R	R	NR	NR	NR	NR
10	NR	R	R	NR	NR	R	NR
11	R	R	R	NR	NR	R	NR
12	R	NR	R	NR	NR	R	NR
13	R	R	NR	NR	NR	R	NR
14	R	R	R	NR	NR	R	NR
15	R	R	R	NR	NR	R	NR
16	R	R	R	NR	NR	R	NR
17	R	R	R	NR	NR	R	NR
18	NR	NR	R	NR	NR	NR	NR
19	R	R	R	NR	NR	R	NR
20	R	NR	NR	NR	NR	R	NR
21	R	NR	R	NR	NR	R	NR
22	NR	NR	NR	NR	NR	R	NR
23	NR	NR	NR	NR	NR	R	NR
24	NR	NR	R	NR	NR	R	NR
25	R	NR	R	NR	NR	NR	NR
26	NR	NR	R	NR	NR	R	NR
27	NR	NR	NR	NR	NR	R	NR
28	R	R	NR	NR	NR	NR	NR
29	R	R	NR	NR	NR	NR	NR
30	NR	NR	R	NR	NR	NR	NR
31	NR	NR	NR	NR	NR	NR	NR
32	R	NR	NR	NR	NR	R	NR
33	NR	NR	NR	NR	NR	R	NR
34	NR	NR	R	NR	NR	R	NR
35	NR	R	NR	NR	NR	NR	NR
36	NR	NR	NR	NR	NR	R	NR
37	R	R	NR	NR	NR	R	NR
38	NR	NR	NR	NR	NR	R	NR
39	NR	NR	NR	NR	NR	R	NR
40	R	NR	NR	NR	NR	NR	NR

Table 18. Information Need 2 Relevance Table

Rank	Base System	Enhanced System	New Schema System	Synonyms System	Keywords System	Embeddings System	Combined System
1	R	R	R	NR	R	R	NR
2	NR	R	NR	NR	NR	NR	NR
3	R	R	R	NR	NR	R	NR
4	R	R	R	NR	NR	R	NR
5	R	R	NR	NR	NR	R	NR
6	NR	R	R	NR	R	NR	NR
7	R	R	R	NR	R	R	NR
8	R	NR	R	NR	R	R	NR
9	R	R	R	NR	NR	NR	NR
10	NR	R	R	NR	NR	R	NR
11	R	R	R	NR	NR	R	NR
12	R	NR	R	NR	NR	R	NR
13	R	R	NR	NR	NR	R	NR
14	R	R	R	NR	NR	R	NR
15	R	R	R	NR	NR	R	NR
16	R	R	R	NR	NR	R	NR
17	R	R	R	NR	NR	R	NR
18	NR	NR	R	NR	NR	NR	NR
19	R	R	R	NR	NR	R	NR
20	R	NR	NR	NR	NR	R	NR
21	R	NR	R	NR	NR	R	NR
22	NR	NR	NR	NR	NR	R	NR
23	NR	NR	NR	NR	NR	R	NR
24	NR	NR	R	NR	NR	R	NR
25	R	NR	R	NR	NR	NR	NR
26	NR	NR	R	NR	NR	R	NR
27	NR	NR	NR	NR	NR	R	NR
28	R	R	NR	NR	NR	NR	NR
29	R	R	NR	NR	NR	NR	NR
30	NR	NR	R	NR	NR	NR	NR
31	NR	NR	NR	NR	NR	NR	NR
32	R	NR	NR	NR	NR	R	NR
33	NR	NR	NR	NR	NR	R	NR
34	NR	NR	R	NR	NR	R	NR
35	NR	R	NR	NR	NR	NR	NR
36	NR	NR	NR	NR	NR	R	NR
37	R	R	NR	NR	NR	R	NR
38	NR	NR	NR	NR	NR	R	NR
39	NR	NR	NR	NR	NR	R	NR
40	R	NR	NR	NR	NR	NR	NR

Table 19. Information Need 3 Relevance Table

Rank	Base System	Enhanced System	New Schema System	Synonyms System	Keywords System	Embeddings System	Combined System
1	R	R	R	NR	R	R	NR
2	NR	R	NR	NR	NR	NR	NR
3	R	R	R	NR	NR	R	NR
4	R	R	R	NR	NR	R	NR
5	R	R	NR	NR	NR	R	NR
6	NR	R	R	NR	R	NR	NR
7	R	R	R	NR	R	R	NR
8	R	NR	R	NR	R	R	NR
9	R	R	R	NR	NR	NR	NR
10	NR	R	R	NR	NR	R	NR
11	R	R	R	NR	NR	R	NR
12	R	NR	R	NR	NR	R	NR
13	R	R	NR	NR	NR	R	NR
14	R	R	R	NR	NR	R	NR
15	R	R	R	NR	NR	R	NR
16	R	R	R	NR	NR	R	NR
17	R	R	R	NR	NR	R	NR
18	NR	NR	R	NR	NR	NR	NR
19	R	R	R	NR	NR	R	NR
20	R	NR	NR	NR	NR	R	NR
21	R	NR	R	NR	NR	R	NR
22	NR	NR	NR	NR	NR	R	NR
23	NR	NR	NR	NR	NR	R	NR
24	NR	NR	R	NR	NR	R	NR
25	R	NR	R	NR	NR	NR	NR
26	NR	NR	R	NR	NR	R	NR
27	NR	NR	NR	NR	NR	R	NR
28	R	R	NR	NR	NR	NR	NR
29	R	R	NR	NR	NR	NR	NR
30	NR	NR	R	NR	NR	NR	NR
31	NR	NR	NR	NR	NR	NR	NR
32	R	NR	NR	NR	NR	R	NR
33	NR	NR	NR	NR	NR	R	NR
34	NR	NR	R	NR	NR	R	NR
35	NR	R	NR	NR	NR	NR	NR
36	NR	NR	NR	NR	NR	R	NR
37	R	R	NR	NR	NR	R	NR
38	NR	NR	NR	NR	NR	R	NR
39	NR	NR	NR	NR	NR	R	NR
40	R	NR	NR	NR	NR	NR	NR

Table 20. Information Need 4 Relevance Table