Book Quotes Information Search System

ADRIANO SOARES, FILIPE CAMPOS, and FRANCISCO CERQUEIRA

In this report, we address the subjects of Data Preparation, Information Retrieval and the construction of a Search System. Foremost, during the Data Preparation step, we obtained information about a large number of books through the Goodreads popular website and assembled our initial data set. We prepared our data set by converting, adding and removing attributes and by removing books that did not have quotes associated, as they are one of our primary objectives. The resultant data set was then statistically and descriptively analysed to better understand what we are working on to create queries search tasks that are far more interesting and ultimately build an information search system based on them. After establishing an information search system using Solr, we indexed our documents through the use of a schema with multiple field types, with their own tokenizers and filters, that better suited our document structure. We proceeded by demonstrating the effectiveness of the system through the analysis of the relevance of the documents retrieved by the informational needs by us defined. We then established a System A with a set of boosts that were applied universally across all queries created and enhanced for each search task and compared it against the System B. This new System B was set up with a data set with an additional Wikipedia book description, with its description and quotes fields clustered by their languages so that we could apply stemmers specifically tailored to them and with field boosts fine tuned using a grid search algorithm. The results obtained were not as satisfactory as we had hoped initially, showing only a slight increase in their mean Average Precision score from 0.96675 to 0.96800.

ACM Reference Format:
Adriano Soares, Filipe Campos, and Francisco Cerqueira. 2023. Book Quotes Information Search System. 1, 1 (August 2023), 15 pages.

1 INTRODUCTION

The goal of this project is the implementation of an information search system. To carry out that goal, the project is subdivided into three milestones with distinct purposes: we will first tackle the data collection and preparation process, then we will proceed by designing queries to retrieve useful information from the data collected and finally we will build a system capable of recovering data with a user-friendly interface.

Our group's chosen theme was quotes from the books at the Goodreads [1] website. This way we ensure not only a large amount of data to be treated but also both textual and numerical data to work on.

This paper starts by presenting the steps of data collection, preparation and characterization. Then, the process of Information Retrieval is described, along with enhancements to the collection structure and an overview of the performance of the search engine over different types of search tasks. Finally, changes to the dataset, schema and query system are described, aiming at the improvement of the search engine.

Authors' address: Adriano Soares, up201904873@edu.fe.up.pt; Filipe Campos, up201905609@edu.fe.up.pt; Francisco Cerqueira, up201905337@edu.fe.up.pt.

2 DATA PREPARATION

2.1 Data Collection

To obtain the initial data, we created a **webcrawler** using the Scrapy [2] framework, written in Python, that assembles the data into a **JSON** structure.

We read through a large public book list[1] that contains 100 pages with at most 100 books each. For each book, up to 50 pages of book quotes were visited, each page with at most 30 quotes, totalling a maximum of 1,500 quotes per book. By the end of the crawler execution, we summed up 854,896 quotes and 9,934 instances of books.

To better understand the next section of the report we will very succinctly explain the information that we gathered: the book title, book description, ISBN (ISBN-10 and ISBN-13 format), genres, number of pages and likes, authors and quotes (each with an author and like count associated).

It should be noted that, at the time of writing this report, Goodreads website is testing a new, completely distinct, book page template. This new page template sporadically materializes in place of the old page template. As such, we require a pragmatic way to distinguish between the two-page templates.

2.2 Data Preparation

With the data obtained, we prepared and cleaned our data set to better analyse it during our next step, data characterization. To perform that task, our pipeline executes multiple Python scripts capable of reading an instance of a book from the standard input, processing it and returning the resultant processed instance through the standard output. This allows us to use the Unix built-in pipeline method to chain multiple scripts and load balance the preparation process through concurrent processes.

2.2.1 Data Preparation Pipeline. Our data preparation pipeline consists of the execution of multiple python micro-scripts that accept a document instance as an input and return the document with minor modifications. Firstly, we convert ISBN-10 standards to ISBN-13 as both were being utilised, then we format the "likes" and "pageCount" attributes from the strings "X likes" and "X pages", respectively, to an integer X. After removing quote-less books, we remove unnecessary white spaces from quotes and identify their language using NLP (through the spaCy[2] python library). To conclude, we add the fields that the crawler was unable to obtain with NULL value to standardize all documents. To better understand these concepts we represented the pipeline in figure 1.

2.3 Conceptual Data Model

The conceptual data model consists of six entities, as seen in Figure 2, being **Quote** the main asset to be analysed. A **Quote** is assigned

[1] https://www.goodreads.com/list/show/1.Best_Books_Ever
[2] https://spacy.io/

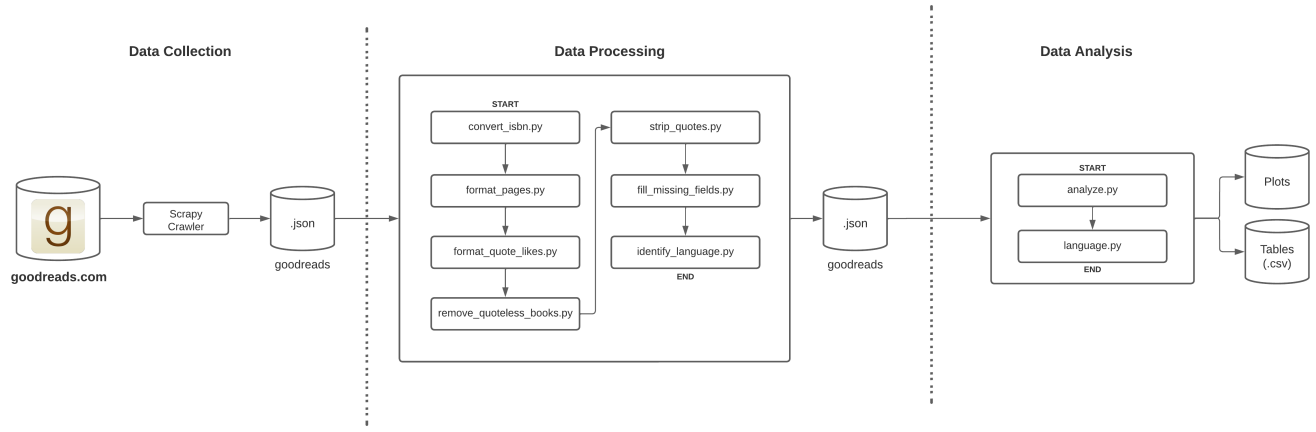

Fig. 1. Pipeline Diagram.

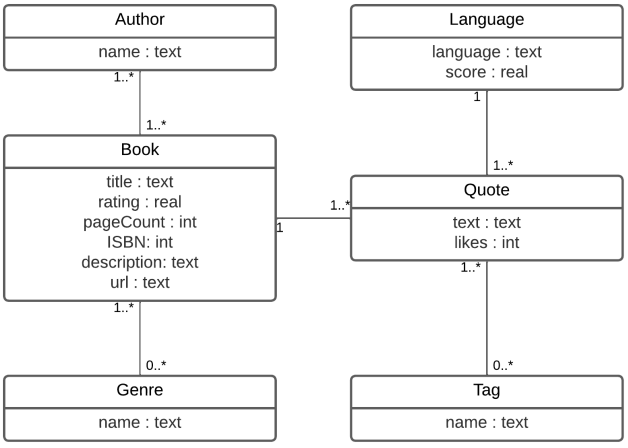

Fig. 2. Domain Model Diagram.

to a single book and language and can have multiple tags associated. It has the following attributes:

- **Text**: String with the text content of the quote;
- **Likes**: Number of likes that the quote has.

A **Book** can have multiple authors, genres and quotes associated. It should be noted that some books may not be labelled with any genre. It has the following attributes:

- **Title**: Name of the book;
- **Rating**: Classification of the book;
- **Page Count**: Number of pages the book has;
- **ISBN**: Numeric commercial book identifier;
- **Description**: Succinct description of the book's content;
- **URL**: Reference to the book's web resource.

Attribute	Value
Raw data size	353 MB
Processed data size	355 MB
Total number of books	9,154 books
Total number of quotes	854,896 quotes
Average number of quotes per book	93.39 quotes
Median number of quotes per book	37 quotes
Number of authors	4,028 authors
Number of genres	693 unique genres
Number of tags	74,114 unique tags

Table 1. Data volume.

2.4 Data Characterization

Our data set after processing is sized at 355 MB, containing a total of 9,154 books and 854,896 quotes, as seen in Table 1. The data set size remains almost identical after being processed as the removal of books without quotes and some unnecessary characters present in the likes and pages fields were evened out by the addition of the language attribute and previously unset attributes.

This amount of data is sufficient to implement our Information Retrieval system, given the large number of quotes available.

Most books have high ratings, around 4 stars. This distribution, which can be seen in Figure 3, is expected since the list we used as the basis for our crawling system is sorted based on popularity.

Most authors present in the list tend to write only 1 or 2 books, as seen in Figure 4, but there are some very well-known authors that are outliers, as they have written an exceedingly large amount of books. The most prominent example is Stephen King with 63 books, as seen in Table 2. The average rating of an author tends to increase with the number of books written, as seen in Figure 5.

Figures 6 and 7 show the most common genres and tags, respectively. This distribution was expected since the top tags and genres

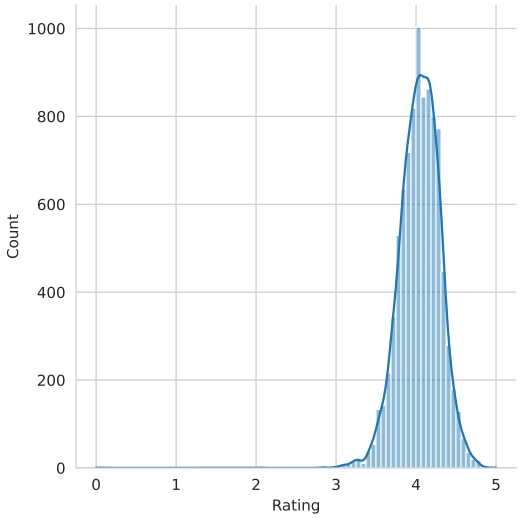

Fig. 3. Rating distribution.

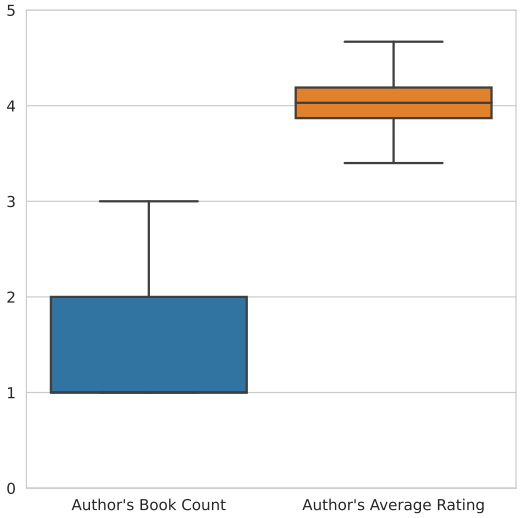

Fig. 4. Book and rating per author distribution.

Author	Number of Books	Average Rating
Stephen King	63	4.02
Terry Pratchett	40	4.16
James Patterson	36	3.95
William Shakespeare	36	3.77
Rick Riordan	34	4.31

Table 2. Authors with most books.

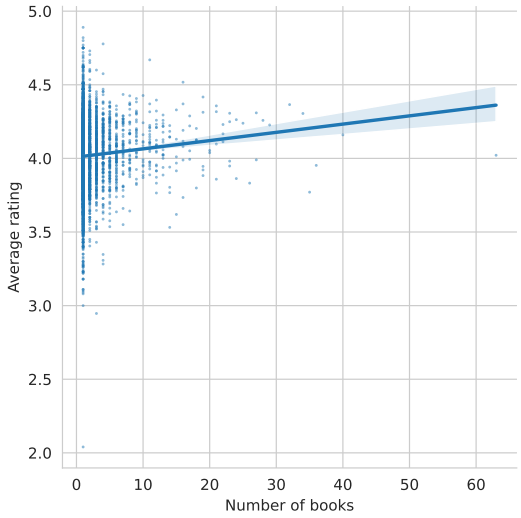

Fig. 5. Average Rating over Number of books written by author.

cover broad topics that can be applied to a plethora of situations. For example the “Fiction” and “Fantasy” genres and the “love” and “humor” tags.

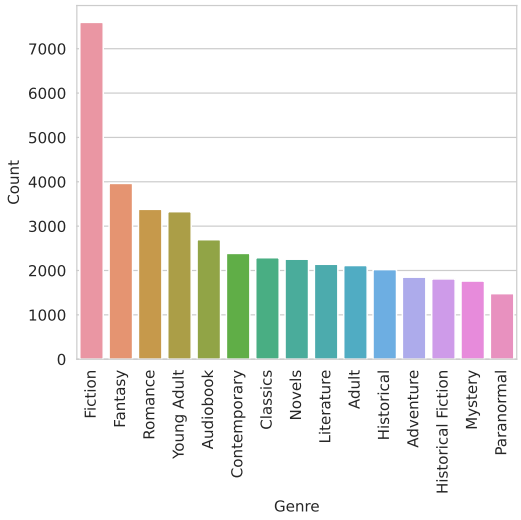

Fig. 6. Most popular book genres.

Based on Figure 8 we concluded that the average number of likes per quote does not correlate with a book’s rating.

Using the data previously obtained using natural language processing, we analysed the language used in each quote, the overwhelming majority were written in English which is represented in Figure 9 that has a logarithmic scale. This was expected since the target audience of our data source, Goodreads, is mostly English speaking.

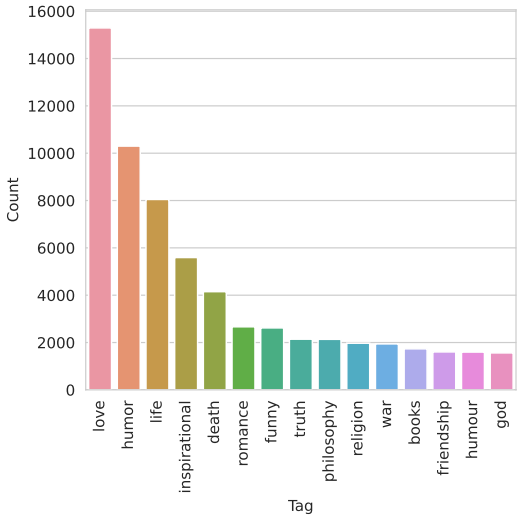

Fig. 7. Most used quote tags.

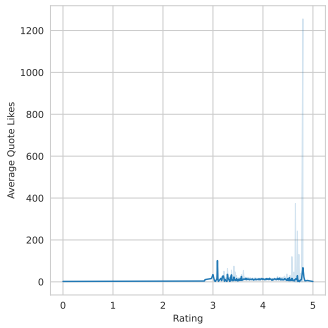

Fig. 8. Average quote likes over book rating.

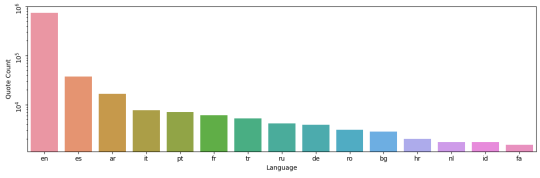

Fig. 9. Most popular quote languages.

2.5 Search Tasks

Different types of queries can be made after obtaining the processed data set, for instance, some interesting retrieval tasks are:

- What are the most popular books about philosophy?
- What are the books whose genre is not philosophy with quotes whose topic is related to philosophy?
- What books written by the author Stephen King are about "epic fantasy"?
- What are the non-fiction books about space with one author and more than 200 pages?

3 INFORMATION RETRIEVAL

The goal of our Information Retrieval system is to find documents within our collection that are relevant, that is, that satisfy our informational needs. In the following subsections, we will focus on the retrieval of information through the use of queries.

To achieve this, we implemented our system using Solr [3], an open-source search platform built on Apache Lucene [4]. Other tools were considered, such as ElasticSearch [5], but they weren't ideal for our purpose.

The set of informational needs previously stated will be translated to queries and evaluated based on their Precision at 5 (P@5), Precision at 10 (P@10) and Average Precision (AP) values [6].

3.1 Collection and Indexing

To further enhance our Information Retrieval capabilities some changes to our pipeline have been made: three scripts have been appended to the processing step that adds the *authorsCount*, *quotesCount* and *genre[1..3]* (top 3 genres) fields.

Type	Field	Indexed
textField	title	true
	authors	true
	ISBN	false
	description	true
	genres	true
	quotes.text	true
	quotes.tags	true
	genre1	true
	genre2	true
	genre3	true
	quotes.language.language	false
integerField	authorsCount	true
	rating	false
	pageCount	true
	quotesCount	true
	quotes.likes	false
	quotes.language.score	false
urlField	url	false

Table 3. Schema fields.

We choose to index our files based on three distinct field types by us defined in our schema.json file: *textField*, *integerField* and *urlField*.

The *textField* extends the Solr TextField class and has a tokenizer and two filters in place, the StandardTokenizerFactory tokenizer, the ASCIIFoldingFilterFactory filter and the LowerCaseFilterFactory filter, respectively. The StandardTokenizerFactory tokenizer splits the text field into tokens, treating whitespace and punctuation as delimiters (does not include delimiters). The ASCIIFoldingFilterFactory filter attempts to convert alphabetic, numeric, and symbolic Unicode characters which are not in the first 127 ASCII characters to their ASCII equivalents. The latter has the argument preserveOriginal set

to True in order to preserve the original tokens (for instance, "olá" ⇒ ["ola", "olá"] and not "olá" ⇒ ["ola"]). The LowerCaseFilterFactory filter converts every uppercase letter to its equivalent lowercase form. We considered using Beider-Morse Phonetic Matching [7] on the "authors" field to make the system more tolerant of input errors but dismissed the idea since the Beider-Morse Filter introduced unnecessary noise to our queries. Additionally, we tested stemming the *quotes.tags* fields but doing so didn't improve our search results.

The *integerField* and *urlField* field types extend the Solr IntPointField and TextField classes, respectively, without any tokenizers or filters coalesced since we did not find any advantage in doing so to fulfil our informational needs and it would only hinder the query processing time.

It's important to note that the *authors*, *genres*, *quotes.text*, *quotes.tags*, *quotes.likes*, *quotes.language.language* and *quotes.language.score* are established as *multiValued* fields given the nature of their values.

3.2 Retrieval Process and Evaluation

To evaluate our retrieval system we analysed the top 10 retrieved documents for each informational need, stating their relevance or non-relevance, allowing us to compute the Precision at 5 (P@5), Precision at 10 (P@10) and Average Precision (AP) metrics [6].

For each search task we attempted to enhance the results after a Baseline query has been settled, either by searching for a given term or phrase in a given field that wasn't being searched on previously; boosting the score if a given term or phrase appears in a relevant field; excluding documents that contain a given term in a given field; applying fuzziness to a given term. Furthermore, we experimented with both the Standard and Extended DisMax query parsers, from now on referenced as Lucene, DisMax and eDisMax query parsers, respectively, each with their own set of features that can be used.

To compile a list of relevant documents (qrels) we retrieved the top twenty documents from each query and manually analysed each respective Goodreads web page to ensure that the book is pertinent. We chose to analyse twenty documents, even though we only calculate P@5 and P@10, so that we could have a data set with more relevant documents that could be retrieved if a new enhanced query is instantiated.

A. What are the most popular books about philosophy?

For this informational need we commenced by searching for the term philosophy in the title, genres, quotes.text and quotes.tags fields, as shown in Table 4. We considered the fact that a book might be philosophical but not have the genre "Philosophy" strictly declared in the genres field, hence the search for the term in the remaining relevant fields.

Parameter	Value
deftype	edismax
q	philosophy
qf	title
	genres
	quotes.text
	quotes.tags

Table 4. Search Task A: Baseline query.

To enhance the results obtained from the previous query we used fuzzy search on the quotes.tags fields with an edit distance of 2, as shown in Table 5. This allows us to find similar tags that could potentially have the same meaning as "Philosophy".

Parameter	Value
deftype	edismax
q	philosophy
	quotes.tags:philosophy~2
qf	title
	description
	genres
	quotes.text

Table 5. Search Task A: Enhanced query.

This change yielded some good results, since we managed to increase the Average Precision value, as demonstrated in Table 6 and Table 7.

Model	AP	P@5	P@10
Baseline	0.895465	0.800000	0.700000
Enhanced	0.908730	0.800000	0.700000

Table 6. Search Task A Metrics.

Rank	1	2	3	4	5	6	7	8	9	10
Baseline	R	R	R	N	R	R	R	N	R	N
Enhanced	R	R	R	R	N	R	N	R	R	N

Table 7. Search Task A Top Documents.

B. What are the books whose genre is not philosophy with quotes whose topic is related to philosophy?

For this search task, we started by searching for the term philosophy in quotes.tags field, but not in the genres field, as shown in Table 8.

Parameter	Value
deftype	lucene
q	-genres:philosophy
	quotes.tags:philosophy

Table 8. Search Task B: Baseline query.

To improve the results achieved by the previous query we attempted the use of the Porter Stem Filter in the textField type so that both "Philosophical" and "Philosophy" quote tags could be found when only searching for the latter. This held no enhancement, as the resultant stemmed version of both terms were not the same, "Philosophical" ⇒ "Philosophic" versus "Philosophy" ⇒ "Philosoph". A wildcard was also experimented using "philosoph*" term in quotes.tags field, but it led to lower performance, according to our metrics. Boosts were also tested, but, as the query only uses one field to perform the search, they had no effect. Therefore, with no successful improvements, no additional enhanced version of this query has been considered. The Precision at 5, Precision at 10 and Average Precision metrics of this search task can be analysed in Table 9 and Table 10.

Model	AP	P@5	P@10
Baseline	0.975000	1.000000	0.800000

Table 9. Search Task B Metrics.

Rank	1	2	3	4	5	6	7	8	9	10
Baseline	R	R	R	R	R	R	R	N	N	R

Table 10. Search Task B Top Documents.

C. What books written by the author Stephen King are about "epic fantasy"?

We consider relevant all the 12 books from Stephen King's website [8], that are described as "epic fantasy" books. This information need could be fulfilled by a simple query that uses filters instead of a relevancy-based approach. We decided to take this route since, if the books that match the criteria do not exist, the system can still suggest books that might be relevant even if it doesn't fulfil the information need in its entirety. As a baseline query (Table 11) we searched "epic fantasy" in the fields description, genres and quotes.tags since they contain the most information about a book's content, alongside a phrase search for the author Stephen King.

Parameter	Value
deftype	edismax
q	epic fantasy authors:"Stephen King"
qf	description genres quotes.tags

Table 11. Search Task C: Baseline query.

The query shown in Table 12 improves upon the baseline, in the sense that it increased the Precision at 10 value even though it reduced the Average Precision, by boosting the *authors* and *genres* fields, which are the most pertinent fields for this search task.

Parameter	Value
deftype	edismax
q	epic fantasy authors:"Stephen King"^20 genres:(epic fantasy)^10
qf	description quotes.tags

Table 12. Search Task C: Enhanced A query.

We observed that the genres field wasn't being used to its maximum potential, since it gave equal weight to all genres in the multi-valued list, even though they are ordered. So we conducted a small test (Table 13) in which we used only the *authors* field and the top genre on the list, which should be the most important.

Parameter	Value
deftype	edismax
q	epic fantasy authors:"Stephen King"^15 genres1:fantasy^30

Table 13. Search Task C: Enhanced B query.

Our final query iteration, described in Table 14, reintroduces the *description*, *quotes.tag* and *genres* fields, and additionally gives different boosts for the second and third genres. The results from this iteration were remarkable, as shown in the precision-recall curve in Figure 20. The precision stays constant until all the relevant books in the collection are obtained, after which point, it starts obtaining similar non-relevant books.

Parameter	Value
deftype	edismax
q	epic fantasy authors:"Stephen King"^15 genre1:fantasy^30 genre2:fantasy^10 genre3:fantasy^2
qf	description quotes.tags genres

Table 14. Search Task C: Enhanced C query.

The key takeaway from Table 15 and Table 16 is that we can achieve near-perfect precision through the use of boosts. Yet, this isn't surprising as the query is also achievable through the use of filters.

Model	AP	P@5	P@10
Baseline	0.805556	0.600000	0.300000
Enhanced A	0.644444	0.400000	0.500000
Enhanced B	0.946781	1.000000	0.900000
Enhanced C	1.000000	1.000000	1.000000

Table 15. Search Task C Metrics.

Rank	1	2	3	4	5	6	7	8	9	10
Baseline	R	N	R	R	N	N	N	N	N	N
Enhanced A	R	N	R	N	N	R	N	R	R	N
Enhanced B	R	R	R	R	R	N	R	R	R	R
Enhanced C	R	R	R	R	R	R	R	R	R	R

Table 16. Search Task C Top Documents.

D. What are the non-fiction books about space with one author and more than 200 pages?

For this informational need we started by searching for documents with "space" and "nonfiction" genres, one author and more than 200 pages, as shown in Table 17. For the same reasons described in search task C, we choose to use a relevance-based approach instead of a filter approach.

Parameter	Value
deftype	lucene
q	genres:space,nonfiction authorsCount:1 pageCount:[200 TO *]

Table 17. Search Task D: Baseline query.

We noticed that, despite the attempt to remove fiction books through the search of the "nonfiction" genre, many science fiction books were still being retrieved. In an attempt to improve upon the baseline query we attempted to fetch books that did not contain the "fiction" genre, or a variation of it, for instance "science-fiction" (Table 18).

Parameter	Value
deftype	lucene
q	genres:space -genres:fiction authorsCount:1 pageCount:[200 TO *]

Table 18. Search Task D: Enhanced A query.

As an experiment, we attempted to search for "space" in the quotes text field (Table 19) and concluded that there's a higher chance of the book being relevant the more the word "space" is referenced.

Parameter	Value
deftype	lucene
q	genres:space,nonfiction -genres:fiction authorsCount:1 pageCount:[200 TO *] quotes.text:space

Table 19. Search Task D: Enhanced B query.

So as to include quotes with "spacial" or similar terms related to space, we applied fuzziness with an edit distance of 2 to the "space" term when searching in the quotes text field, as shown in Table 20.

Parameter	Value
deftype	lucene
q	genres:space,nonfiction -genres:fiction authorsCount:1 pageCount:[200 TO *] quotes.text:space~2

Table 20. Search Task D: Enhanced C query.

These changes yielded some good results, since we managed to increase the Precision at 10 value, as demonstrated in Table 21 and Table 22.

Model	AP	P@5	P@10
Baseline	0.976543	1.000000	0.900000
Enhanced A	0.976543	1.000000	0.900000
Enhanced B	1.000000	1.000000	0.900000
Enhanced C	1.000000	1.000000	1.000000

Table 21. Search Task D Metrics.

Rank	1	2	3	4	5	6	7	8	9	10
Baseline	R	R	R	R	R	R	R	N	R	R
Enhanced A	R	R	R	R	R	R	R	N	R	R
Enhanced B	R	R	R	R	R	R	R	R	R	N
Enhanced C	R	R	R	R	R	R	R	R	R	R

Table 22. Search Task D Top Documents.

3.3 Single System

As to proceed to the next stage we first established a system whose boosts are universally set to be used across all our search tasks. This way we can easily compare it to any future system developed. We expected a small decrease in performance, when compared to the previous evaluations because in this system the boosts cannot be fine tuned to each individual search task.

We left the query parameters for each informational need as close as the originals and added fields to the query fields with different boost values that we thought pertinent: *title* field with a boost of 5; *authors* field with a boost of 3; *description* field with a boost of 4; *genre1* field with a boost of 3; *genre2* field with a boost of 2; *genre3* field with a boost of 1.5; and, finally, *quotes.text* and *quotes.tags* fields without an associated boost.

We'll proceed by demonstrating the results obtained solely using the query versions that achieved the best results.

A. What are the most popular books about philosophy?

Using the query defined in Table 23 we obtained a slight decrease in the average precision of the query when compared to the one described in Table 5, but the P@5 and P@10 values remained the same (Table 24 and 25).

Parameter	Value
deftype	edismax
q	philosophy
	quotes.tags:philosophy~2
qf	title^5
	author^3
	description^4
	genre1^3
	genre2^2
	genre3^1.5
	quotes.text^1
	quotes.tags^1

Table 23. Search Task A: Enhanced query with boosts.

Model	AP	P@5	P@10
Enhanced with Boosts	0.891723	0.800000	0.700000

Table 24. Search Task A: Enhanced query with boosts metrics.

Rank	1	2	3	4	5	6	7	8	9	10
Enhanced with Boosts	R	R	R	R	N	N	R	R	R	N

Table 25. Search Task A: Enhanced query with boosts top documents.

B. What are the books whose genre is not philosophy with quotes whose topic is related to philosophy?

The results obtained from the query described in Table 26, shown in Tables 27 and 28, remained exactly the same as they were before.

Parameter	Value
deftype	edismax
q	-genres:philosophy
	quotes.tags:philosophy
qf	title^5
	author^3
	description^4
	genre1^3
	genre2^2
	genre3^1.5
	quotes.text^1
	quotes.tags^1

Table 26. Search Task B: Baseline query with boosts.

Model	AP	P@5	P@10
Baseline with Boosts	0.975000	1.000000	0.800000

Table 27. Search Task B: Baseline query with boosts metrics.

Rank	1	2	3	4	5	6	7	8	9	10
Baseline with Boosts	R	R	R	R	R	R	R	N	N	R

Table 28. Search Task B: Baseline query with boosts top documents.

C. What books written by the author Stephen King are about "epic fantasy"?

The query shown in Table 29 presents the same results as its predecessor, as seen in Tables 30 and 31.

Parameter	Value
deftype	edismax
q	epic fantasy
	authors:"Stephen King"^15
	genre1:fantasy^30
	genre2:fantasy^10
	genre3:fantasy^2
qf	title^5
	author^3
	description^4
	genre1^3
	genre2^2
	genre3^1.5
	quotes.text^1
	quotes.tags^1

Table 29. Search Task C: Enhanced C query with boosts.

Model	AP	P@5	P@10
Enhanced C with Boosts	1.000000	1.000000	1.000000

Table 30. Search Task C: Enhanced C query with boosts metrics.

Rank	1	2	3	4	5	6	7	8	9	10
Enhanced C with Boosts	R	R	R	R	R	R	R	R	R	R

Table 31. Search Task C: Enhanced C query with boosts top documents.

D. What are the non-fiction books about space with one author and more than 200 pages?

Similarly to the previous search tasks, the query described in Table 32 shows no changes to its results (Tables 33 and 34).

Parameter	Value
deftype	edismax
q	genres:space,nonfiction -genres:fiction authorsCount:1 pageCount:[200 TO *] quotes.text:space~2
qf	title^5 author^3 description^4 genre1^3 genre2^2 genre3^1.5 quotes.text^1 quotes.tags^1

Table 32. Search Task D: Enhanced C query with boosts.

Model	AP	P@5	P@10
Enhanced C with Boosts	1.000000	1.000000	1.000000

Table 33. Search Task D: Enhanced C query with boosts metrics.

Rank	1	2	3	4	5	6	7	8	9	10
Enhanced C with Boosts	R	R	R	R	R	R	R	R	R	R

Table 34. Search Task D: Enhanced C query with boosts top documents.

From this point onward this system will be referenced as **System A**, and each query will be referenced as "Query for Search Task <search_task> from System A".

4 SYSTEM IMPROVEMENTS

We developed an improved system, henceforth referred to as **System B**, through the use of 3 different techniques which will be detailed in the following subsections.

4.1 Multi-Language Stemming

Stemming is a technique that reduces a word into its root form that, when used both at indexing and query time processes, should improve the system's relevant document-finding capabilities since we don't longer need an exact word match.

One of the issues we faced while developing the schema, was how to properly apply stemming to textual fields that include many different languages. Most of the stemmer filters available assume that either the text is English or that the language is known beforehand, which is not the case in this project.

Our solution to this issue was to apply natural language processing (NLP) techniques to identify the language used in the *description* and *quotes.text* fields, cluster them into new fields distinguished by language (e.g. *description_en* and *quotes_en.text*) and apply the correct stemmer for each of them by creating different field types with different stemmer configurations.

We identified languages in a new data-processing task, leveraging the capabilities of the spaCy [9] library to perform the necessary analysis. It managed to identify 28 and 52 different languages for descriptions and quotes, respectively. The schema was therefore extended with the creation of several *description_<language_iso_code>*, *quotes_<language_iso_code>.text*, *quotes_<language_iso_code>.tags* and *quotes_<language_iso_code>.likes* fields.

Many of the languages identified are not handled by our stemmer of choice, the Snowball Porter Filter Factory. For these cases, we choose their field types based on the default schema established by Solr.

An example of one of the resultant field types can be seen in Listing 1.

```
{
    "name": "textField_pt",
    "class": "solr.TextField",
    "analyzer":{
        "tokenizer":{
            "class": "solr.
                StandardTokenizerFactory"
        },
        "filters":[
            {"class": "solr.
                ASCIIFoldingFilterFactory", "
                preserveOriginal": "true"},
            {"class": "solr.LowerCaseFilterFactory
                "},
            {"class": "solr.
                SnowballPorterFilterFactory", "
                language": "Portuguese"}
        ]
    }
}
```

Listing 1. Schema text field example

4.2 Wikipedia Book Description Data Acquiral

Most of the textual fields present in the original dataset present a subjective view of a book's content since they were written by the author itself (e.g description and quotes). This creates a bias that leads to a view of a book that might not be entirely correct. To combat this problem we introduce external data from each book's Wikipedia [10] page which should be more descriptive and objective – more specifically, we obtained the description of each book and saved it in the newly created *wikipedia_description* field. This should improve the performance of our system, in particular, in search tasks that probe for a specific topic, since this description includes more detailed information about a book's content.

To obtain these descriptions we used the MediaWiki web service API [11]. We acquired descriptions for 7533 out of 9167 books because the remaining books either did not have a Wikipedia page or that page did not contain a description.

4.3 Grid Search

The manual process of choosing the best boosts to apply to each field would be both time-consuming and inaccurate since there are too many possible combinations to experiment with, leading to non-optimal choices.

To solve this issue we decided to apply grid search [12], a technique commonly used in the field of Machine Learning, to discover the optimal boost combination for our search tasks. This served as a solution for both our initial problems.

This algorithm tweaks the fields present in the query field attributes, these being the *title*, *authors*, *genre1*, *genre2*, *genre3* and *wikipedia_description* by running the aforementioned search tasks with all combinations of boosts in the integer range $[1, 10]$ and evaluating them based on the mean Precision at 10 of the system. In some cases, some systems achieved the same mean Precision at 10 values, to break the tie we choose the one with a higher mean Average Precision score.

To conclude, the boost combination which maximizes these values is: *title* field without any boost; *authors* field without any boost; *genre1* field with a boost of 2; *genre2* field with a boost of 2; *genre3* field without any boost; and *wikipedia_description* field without any boost.

4.4 Redefined Queries

Since the previously mentioned improvements led to changes to our schema, we had to review the queries used in each of the search tasks.

To transform the previous queries into the ones shown in Tables 35, 36, 37 and 38 we had to move both the quotes fields and the description to the queries themselves since the language used is query dependent.

A. What are the most popular books about philosophy?

Parameter	Value
deftype	edismax
q	philosophy
	quotes_en.text:philosophy
	quotes_en.tags:philosophy~2
	description_en:philosophy
qf	title
	authors
	genre1^2
	genre2^2
	genre3^1
	wikipedia_description

Table 35. Search Task A: Redefined query for System B.

B. What are the books whose genre is not philosophy with quotes whose topic is related to philosophy?

Parameter	Value
deftype	edismax
q	-genres:philosophy
	quotes_en.tags:philosophy
qf	title
	authors
	genre1^2
	genre2^2
	genre3^1
	wikipedia_description

Table 36. Search Task B: Redefined query for System B.

C. What books written by the author Stephen King are about "epic fantasy"?

Parameter	Value
deftype	edismax
q	epic fantasy
	description_en:(epic fantasy)
	quotes_en.tags:(epic fantasy)
	authors:"Stephen King"^15
	genre1:fantasy^30
	genre2:fantasy^10
	genre3:fantasy^2
qf	title
	authors
	genre1^2
	genre2^2
	genre3^1
	wikipedia_description

Table 37. Search Task C: Redefined query for System B.

D. What are the non-fiction books about space with one author and more than 200 pages?

Parameter	Value
deftype	edismax
q	space nonfiction
	-genres:fiction^3
	genres:space^5
	genres:nonfiction^2
	authorsCount:1
	pageCount:[200 TO *]
	quotes_en.text:space^5
qf	title
	authors
	genre1^2
	genre2^2
	genre3^1
	wikipedia_description

Table 38. Search Task C: Redefined query for System B.

4.5 Performance Improvement

To validate our improvements, we compared the new system with System A, described in section 3.3.

Model	AP	P@5	P@10
System A	0.891723	0.800000	0.700000
System B	0.962654	1.000000	0.900000

Table 39. Search Task A: System comparison of performance.

Rank	1	2	3	4	5	6	7	8	9	10
System A	R	R	R	R	N	N	R	R	R	N
System B	R	R	R	R	R	R	N	R	R	R

Table 40. Search Task A: System comparison of top documents.

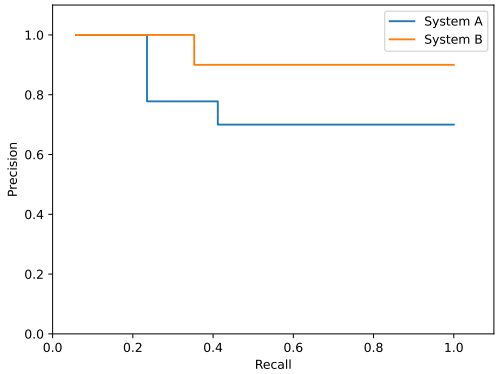

Fig. 10. Search Task A: Precision-Recall Curve comparison between systems

As we can see in Figure 10 and on Tables 39 and 40, the improved system outperforms the previous one, having consistently better precision.

Model	AP	P@5	P@10
System A	0.975000	1.000000	0.800000
System B	0.908532	0.800000	0.800000

Table 41. Search Task B: System comparison of performance.

Rank	1	2	3	4	5	6	7	8	9	10
System A	R	R	R	R	R	R	R	N	N	R
System B	R	R	R	R	N	R	R	N	R	R

Table 42. Search Task B: System comparison of top documents.

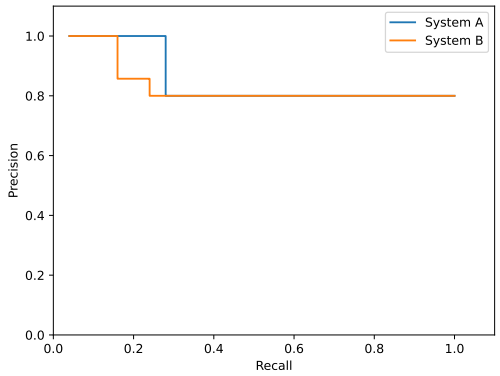

Fig. 11. Search Task B: Precision-Recall Curve comparison between systems

As a small trade-off for the aforementioned improvement, in search task B the results worsened, when compared to the previous system as seen in Figure 11. In the results (Tables 41 and 42) this led to an lower AP and P@5 value, but the P@10 stayed the same. This is not ideal since, for a user, the most important documents tend to be the top few results, making a lower P@5 undesirable.

Model	AP	P@5	P@10
System A	1.000000	1.000000	1.000000
System B	1.000000	1.000000	1.000000

Table 43. Search Task C: System comparison of performance.

Rank	1	2	3	4	5	6	7	8	9	10
System A	R	R	R	R	R	R	R	R	R	R
System B	R	R	R	R	R	R	R	R	R	R

Table 44. Search Task C: System comparison of top documents.

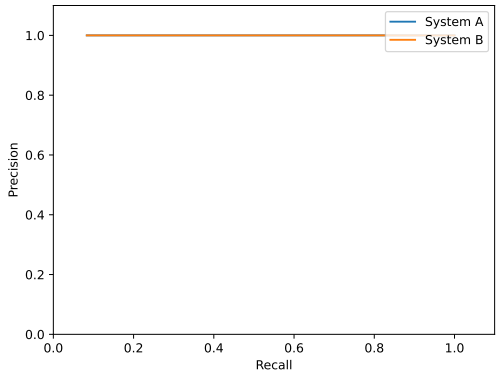

Fig. 12. Search Task C: Precision-Recall Curve comparison between systems

Model	AP	P@5	P@10
System A	1.000000	1.000000	1.000000
System B	1.000000	1.000000	1.000000

Table 45. Search Task D: System comparison of performance.

Rank	1	2	3	4	5	6	7	8	9	10
System A	R	R	R	R	R	R	R	R	R	R
System B	R	R	R	R	R	R	R	R	R	R

Table 46. Search Task D: System comparison of top documents.

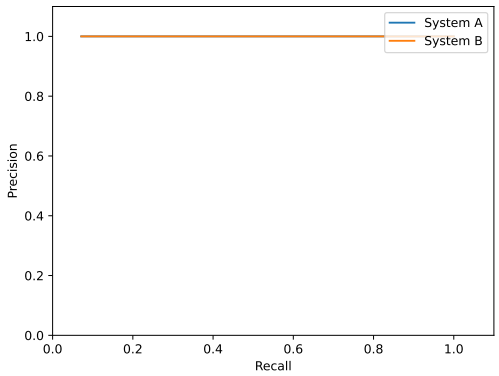

Fig. 13. Search Task D: Precision-Recall Curve comparison between systems

Both Figure 12 and Figure 13 demonstrate that the System B did not worsen the excellent results obtained previously for search tasks C and D. This conclusion is further validated by Tables 43 and 44 for search task C, and Tables 45 and 46 for search task D.

In conclusion, our improvements had a positive impact on one of the search tasks, and a smaller negative impact on another. This change in performance seems like an overall improvement, unfortunately, we could not take any meaningful conclusions from the results of the two remaining search tasks, therefore making it hard to ascertain whether or not there was a significant improvement over System A, even though the difference in the mean average precision was very small, as demonstrated in Table 47.

System	Mean Average Precision
System A	0.96675
System B	0.96800

Table 47. System MAP comparison

As future work, we propose validating these improvements by closely monitoring the search tasks created by real users on both of the systems which would provide us with more data points from which to draw conclusions.

5 CONCLUSIONS

In this paper, we presented the process used to develop a book Information Retrieval system that incorporates book quotes data. During the Data Preparation stage, we demonstrated how we collect the original Goodreads data using a web crawler and prepare it in order to build our own dataset. We defined a set of search tasks that would define the information needs that must be fulfilled by our system and used them to guide its development. Using our previously prepared data set, we moved on to the Information Retrieval process. We chose to use the tool Solr to index and retrieve our documents since it was the most adequate for our needs. To demonstrate the effectiveness of our system we attempted to retrieve relevant documents from our dataset using the previously established informational needs and evaluated the results. By adjusting the query parameters we achieved a system with a satisfactory Mean Average Precision, compared to a simpler baseline implementation. Over the previously mentioned system we made additional improvements: expanded the dataset using Wikipedia data, applied multi-language stemming and used grid-search to fine tune our boosts which led to a small but inconclusive improvement to the mean average precision of the system. This process lead to a precise and useful Information Retrieval tool.

REFERENCES

[1] "Goodreads | meet your next favorite book," accessed: 2022-11-17. [Online]. Available: https://www.goodreads.com/
[2] "A fast and powerful scraping and web crawling framework," accessed: 2022-10-10. [Online]. Available: https://scrapy.org/
[3] "Welcome to apache solr," accessed: 2022-11-16. [Online]. Available: https://solr.apache.org/
[4] "Apache lucene - the apache software foundation!" accessed: 2022-11-16. [Online]. Available: https://lucene.apache.org
[5] "Elasticsearch: The official distributed search analytics engine," accessed: 2022-11-16. [Online]. Available: https://www.elastic.co/elasticsearch/
[6] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval: The Concepts and Technology behind Search*, 2nd ed. USA: Addison-Wesley Publishing Company, 2011.
[7] "Beider-morse phonetic matching," accessed: 2022-11-16. [Online]. Available: https://stevemorse.org/phoneticinfo.htm
[8] "Stephen king | the official website," accessed: 2022-11-17. [Online]. Available: https://stephenking.com/
[9] "Spacy · industrial-strength natural language processing in python," accessed: 2022-10-12. [Online]. Available: https://spacy.io/
[10] "Wikipedia," accessed: 2022-12-15. [Online]. Available: https://www.wikipedia.org/
[11] "Api:main page - mediawiki," accessed: 2022-12-15. [Online]. Available: https://www.mediawiki.org/wiki/API:Main_page
[12] P. Liashchynskyi and P. Liashchynskyi, "Grid search, random search, genetic algorithm: A big comparison for nas," 2019. [Online]. Available: https://arxiv.org/abs/1912.06059

APPENDIX

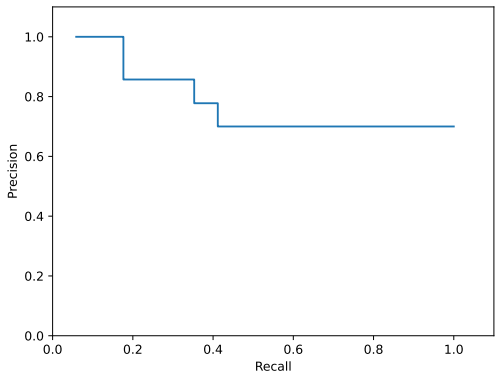

Fig. 14. Search Task A: Baseline query precision-recall curve.

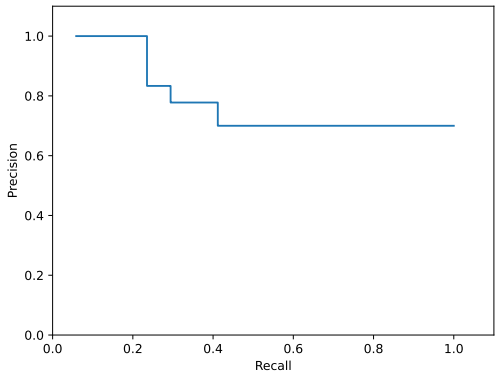

Fig. 15. Search Task A: Enhanced query precision-recall curve.

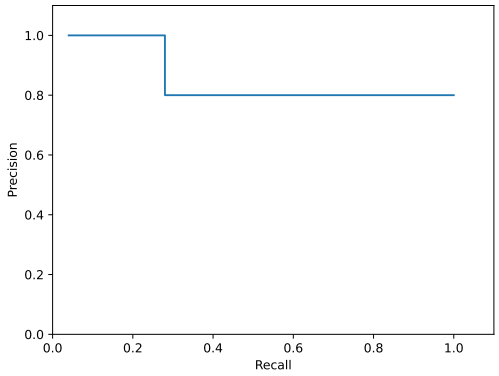

Fig. 16. Search Task B: Baseline query precision-recall curve.

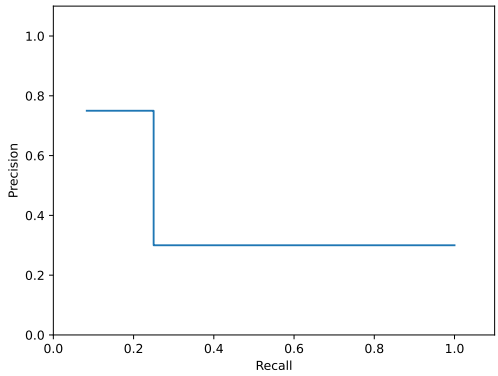

Fig. 17. Search Task C: Baseline query precision-recall curve.

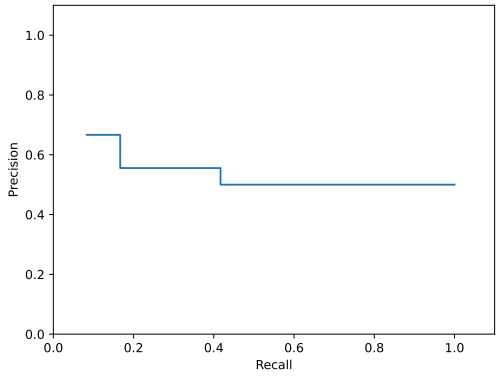

Fig. 18. Search Task C: Enhanced A query precision-recall curve.

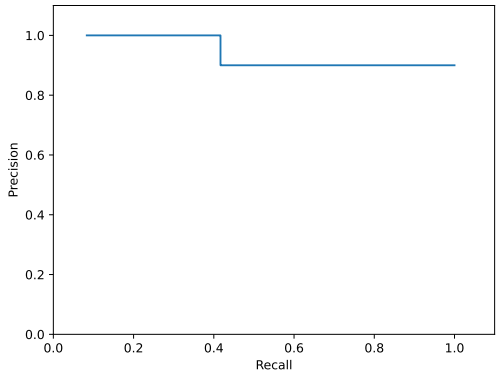

Fig. 19. Search Task C: Enhanced B query precision-recall curve.

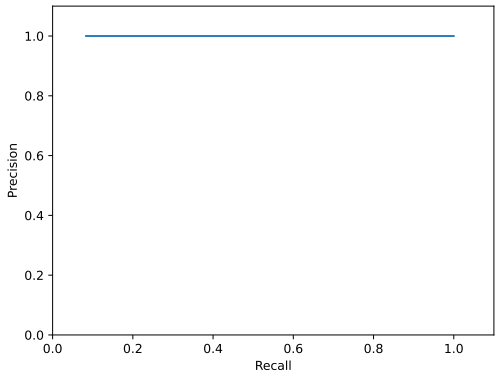

Fig. 20. Search Task C: Enhanced C query precision-recall curve.

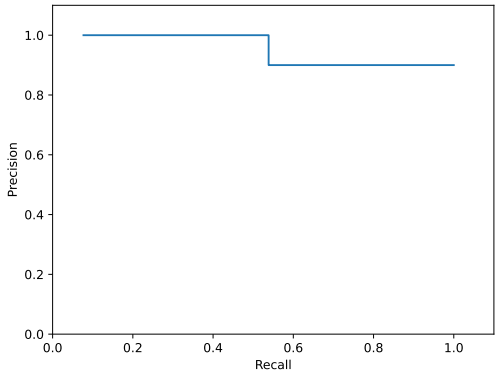

Fig. 21. Search Task D: Baseline query precision-recall curve.

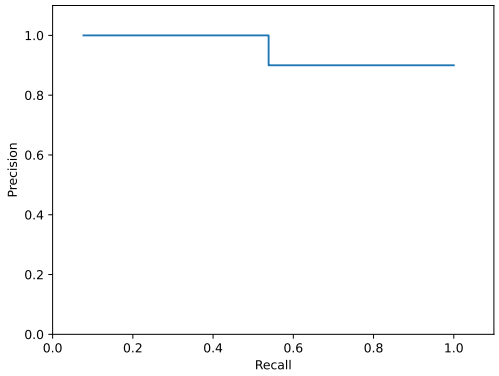

Fig. 22. Search Task D: Enhanced A query precision-recall curve.

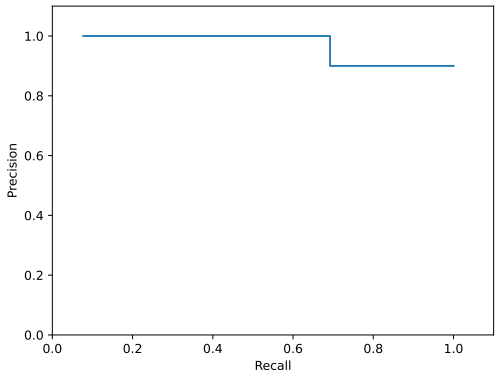

Fig. 23. Search Task D: Enhanced B query precision-recall curve.

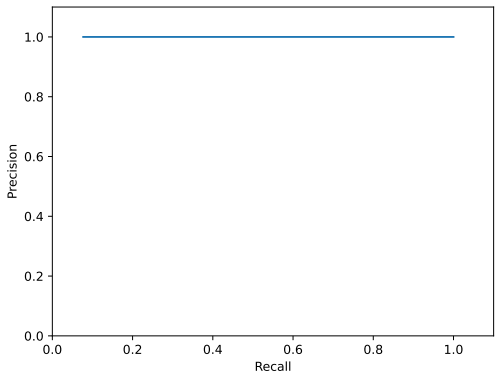

Fig. 24. Search Task D: Enhanced C query precision-recall curve.

Example Document


```
{
    "title": "The Amulet of Samarkand",
    "authors": ["Jonathan Stroud"],
    "rating": "4.02",
    "pageCount": 462,
    "ISBN": "9780786818594",
    "genres": ["Fantasy", "Young Adult", "Fiction", "Magic", "Childrens", "Middle Grade", "Adventure", "
        Urban Fantasy", "Young Adult Fantasy", "Audiobook"],
    "url": "https://www.goodreads.com/book/show/334123.The_Amulet_of_Samarkand",
    "quotes_en": [
        {
            "text": "\u201cOne magician demanded I show him an image of the love of his life. I rustled up
                 a mirror.\u201d \u2015",
            "tags": ["amulet-of-samarkand", "bartimaeus-trilogy", "jonathan-stroud"],
            "likes": 381
        }
    ],
    "quotes_de": [
        {
            "text": "\u201cDazu bedarf es [...] vor allem des richtigen Namens. Ich meine, es ist ja nicht
                 so, als bestellte man ein Taxi\u2014bei einer Beschw\u00f6rung kommt nicht einfach !\
                u201d \u2015",
            "tags": ["humor", "namen"],
            "likes": 5
        }
    ],
    "description_en": "Nathaniel is a boy magician-in-training, sold to the government by his birth
        parents at the age of five and sent to live as an apprentice to a master. Powerful magicians rule
        Britain, and its empire, and Nathaniel is told his is the \"ultimate sacrifice\" for a \"noble
        destiny.\" \n\nIf leaving his parents and erasing his past life isn't tough enough, Nathaniel's
        master, Arthur Underwood, is a cold, condescending, and cruel middle-ranking magician in the
        Ministry of Internal Affairs. The boy's only saving grace is the master's wife, Martha Underwood,
        who shows him genuine affection that he rewards with fierce devotion. Nathaniel gets along
        tolerably well over the years in the Underwood household until the summer before his eleventh
        birthday. Everything changes when he is publicly humiliated by the ruthless magician Simon
        Lovelace and betrayed by his cowardly master who does not defend him.\n\nNathaniel vows revenge.
        In a Faustian fever, he devours magical texts and hones his magic skills, all the while trying to
        appear subservient to his master. When he musters the strength to summon the 5,000-year-old djinni
         Bartimaeus to avenge Lovelace by stealing the powerful Amulet of Samarkand, the boy magician
        plunges into a situation more dangerous and deadly than anything he could ever imagine.",
    "genre1": "Fantasy",
    "genre2": "Young Adult",
    "genre3": "Fiction",
    "wikipedia_description": "The Amulet of Samarkand is a children's novel of alternate history, fantasy
        and magic. It is the first book in the Bartimaeus trilogy written by English author Jonathan
        Stroud. The first edition (paperback) was published in September 2003 by Doubleday in the United
        Kingdom. The book and series are about power struggles in a magical dystopia centred in London,
        England, and features a mix of current and ancient, secular and mythological themes. The book is
        named after a magical artifact created in the ancient Asian city of Samarkand, around which the
        story revolves.",
    "authorsCount": 1
}
```