Joy Albertini

Project Nr 3 Book-Search

Project

My **project consists** of a **crawl** data from book repositories and create a web search specifical for books, the projects specifies that the book should also searched by title, but my search engine not only consider the title but also the context of a book, retrieving **more matching results.**

Crawling

Crawling domains

For my project I **used mainly three domains:**

1) https://directory.doabooks.org/recent-submissions?offset=0

Reason for choosing this:

- **completeness of the data:** in specific I needed the fields: title, **year of publication**, **language**, **authors**, **subjects**, **description**. This is the **best source** of data that I found; all the data is very complete in particular **subjects.**
- **Consistency of the data:** most book data are consistent, especially in English, but in other languages are missing translated data such as **subjects** and sometimes even the **description**.
- **Open-source data**: the site even gives you all the book data displayed in excel format, so crawling data shouldn't be a problem
- **Quantity of data:** the site should contain 46641 books
- **Easy access on newest book** because of the recent collection, useful in my case because I must limit my search on book from 2017 to 2021.
- **Access to a continuous stream of books:** with one link I can get up to 46641 books
- **Can index pages simply by increasing an URL value**

Downside is that most Volumes are in English, giving not much option in term of Language.

2) https://www.feedbooks.com/recent

Reason for choosing this:

- **completeness of the data** (as above), unfortunately **subjects** are quite simple on this site and not as specific on the above one.
- **Consistency of the data:** (as above) most of the data is consistent, there are a few mistakes in languages that are not **English**.
- **Quantity of data**: the site should contain **359,569** books only in English.
- **Easy access on newest book** (as above)
- **Access to a continuous stream of books:** (as above)
- **Quantity of different languages:** that's the main reason because I chose this source, so that I expand the set of languages present in my collection, the site has multiple languages with book titles description etc in the selected language. I crawled multiple links from the source with different language option.
- **Can index pages simply by increasing an URL value**

3) https://www.bookdepository.com/bestsellers?searchLang=123&page=0

Reason for choosing this:

- **completeness of the data:** (as first), **subjects still** not as specific

- **Quantity of data**: the site should contain **97295** books only in English
- **Easy access on newest book** (as above)
- **Access to a continuous stream of books:** (as above)
- **Can index pages simply by increasing an URL value.**

Unfortunately, this site is the worst one I picked, a lot of non-English book data is **inconsistent**, in particular subjects are only in English no matter which language you **enforce on the site.**

Problem data inconsistency

As mentioned above a major problem in crawling the data from sites was the **inconsistency of the data**.

Crawled Data

Full data in `project3book/project3Book/data/bookData3.json,` I crawled 16838 books

Crawling implementation

Code: `project3Book/project3book/spiders/bookSpider.py`

The crawling is quite simple, it uses the same **logic for all 3 sources:**

Urls

- Each **URL** specified in the `start_url` are **URLs** that point to the most recent submission (books) or **bestsellers** such that is easier to crawl the books released in the last 5 years.
- **Feedbooks URLs** also specifies a language meaning that all book retrieved will have such language, I design it like to be sure to have a bunch of different languages present in my collection.
- Each **URLs** can define the offset of a page by an numeric value, making it easy to index in the next page by increasing the offset in the page, I will call them `offset_url`

Generating offset URLs function

Function `offsetLinkGenerator` takes an `offset_url` and generates all the **URLs** to follow in a **while loop** stopped by parameter `pageMaxOffset`, such parameter is used to control the amount of data the current source will retrieve as un **upper bound**, the link generated will be passed to the **parse links function**.

Parse Links functions

Functions: `parseDoaBooksLinks`, `parseFeedBooksLinks`, `parseBookRepositoryLinks` one for each source, this function will get all the **links** for each book in the list of book present in each `offset_url`, the links of each book is sent to the **parse data functions.**

Parse Data functions

Functions: `parseBookRepositoryData`, `parseFeedBookData`, `parseDoaBookdata` one for each source, each one of those **function** will get all the **data** needed for each **book** scraping the link with full information about a single **book**. Each of those function will create an object `bookObject`, setting all data in it.

Book object

Book objects contains the data scraped in the **parse function**:

Function: `checkYearOfPublication` : will output to file the book only if it's year of publication is between 2017 – 2021.

Function: `computeSuffix` : the book object is that will dynamically modify the name of the fields so that `solr` will parse appropriately **based on the language**; this will be done for fields : `title, description subjects.` For example, if the language of the book is **english** the fields become `title_txt_en`, `description_txt_en` and `subjects_txt_en` , for Italian `title_txt_in`, `description_txt_it` .

Joy Albertini

Solr

CORS

I develop only the front-end without a server, so a query directly `solr` to retrieve data, but in order to make the request to `solr` I must **deactivate CORS protection**, I done it by adding lines `24-39` to file `solr-8.11.0/server/solr-webapp/webapp/WEB-INF/web.xml`

Override max URI size

As you will see from my report my query becomes very long, so long that the if you would add more than 4 words `solr` will not accept the query with error **Uri too long**, to fix it I had to modify the max `requestHeaderSize` from 8192 to 65536, in file `solr-8.11.0/server/etc/jetty.xml`

Solr implementation

I divide two concept **base-query** and **operators**

Base-query: I defined **base-query** the query that user writes in the main search box:

Query solr: base-query

Code for defining the query in `bookSearchUI/quasar-demo/src/store/solr.js`

Ored-words in fields

To get most results I or-ed all the words split by space in the **query**, so for example “Children of Dune” in the **query** become:

```
title:(Children || of || Dune)
```

This will allow to get all books with title:

1. with precisely this sentence (`children of dune`) because: Precisely word of query → **higher score**.
2. With all matched word with **words** not **precisely** in the **order** of the **query** (Dune : third chapter of children) → **lower score than 1)**.
3. With not all **matched words** (Dune: the first novel) → **lower score than 2)**.

Using **AND** between word, the query become **Over-constrained**, because:

- will only get results if the user knows exactly what he/she want to **retrieve (precise title)**,
- will not find book **with similar context.**
- doesn’t account for user **mistake**, example if the user writes “children dune” will not get any result.

OR the words, the query can be **Under-constrained**, but I find the retrieving more result is better than less, because both **ad-hoc need** and **users that want to get as many relevant docs as possible** are satisfied, because: the first result are the more relevant, so the ad-hoc need is satisfied, and there are many results.

Multiple fields

There are multiple fields that are important for retrieving book not only with matching title but also context, I defined separately:

- `title`
- `description`
- `subjects`

- `allOtherData`: containing field `yearOfPublication`, `language`, `authors` and all other data present in the page that doesn't have a specific field in JSON.

So, the full query becomes:

```
(title:(Children || of || Dune) || description:(Children || of || Dune) ||
subjects:(Children || of || Dune) || allOtherData:(Children || of || Dune))
```

Ored fields for the same reason as above **AND** will be **Over-constraining**.

`allOtherData`: field is contemplated since if the user query for example "`children of dune 2021`" or `children of dune Frank Herbert` an entry with such with field `author` or `year` matching should have a higher score. This should be specified using **operators** (**later**), but I considered the case and user makes a **mistake** or doesn't want to define an operator for a **quick search**; also contains other data without a specific field.

Field boost

Some fields are more important than others:

- `subjects` most important field, because it specifies at best the context of book, even better than the title. (boost ^2.1)
- `title` most important field when a **query** is exactly a title of a book and not the context such **title**. (boost^2.1)
- `Description` still important, since it also specifies the context of a book, but less precise than `subjects.` (boost ^1.7)
- `allOtherData` less important field, should not be **contemplated**. (no boost)

So the full query becomes:

```
(title:(Children || of || Dune))^2.1 || (description:(Children || of || Dune))^2.1 ||
(subjects:(Children || of || Dune))^1.7 || (allOtherData:(Children || of || Dune))
```

Multiple Language

My project needs to support multiple languages, multiple languages mean multiple different parsing of the data (stop-word removal, stemming of words ecc…), to compute it for each book I need to set the language per book for parser/indexing. This is done in the scraping part `title`, `description` and `subjects` is assigned a suffix based on the language (_txt_en, _txt_it).

Having such field with such **suffix**, will use `solr` dynamic fields and will parse/index the data in a language specific way as defined in `solr-8.11.0/server/solr/configsets/_deafult/conf/managed-schema`

Example _txt_en : indexing/parsing JSON field and query:

text: "So, not all apples are Apple's computers."

ST	So	not	all	apples	are	Apple's	computer
SF	So		all	apples		Apple's	computer
LCF	so		all	apples		apple's	computer
EPF	so		all	apples		apple	computer
SKMF	so		all	apples		apple	computer
PSF	so		all	appl		appl	comput

- **ST standard-tokenizer**: this tokenizer splits the text field into tokens, treating whitespace and punctuation as delimiters, with some exception:
    - **Periods** (**dots**) followed by **whitespace** are kept as **part of the token.**
- **SF stop filter:** removes stop-word token, uses `lang/stopwords_en.txt` list.
- **LCF low case filter:** makes all word characters lower case.
- **EPF English possessive filter**: This filter removes **singular possessives** (trailing 's) from words.
- **SKMF set key word marker filter:** set keywords not to parse by the PSF
- **PSF Porter Stem filter:** stemmer appropriate only for English, **very fast.**

Each language specific `solr` dynamic field is implemented differently, and should be without modification very performant.

Supported languages: Arabic, English, German, Spanish, Finnish, French, Italian, Dutch, Norwegian, Portuguese

When a user performs a query in the search-box I don't know the language of the query, so I must evaluate the query for all languages **supported**, this is done by Oring all fields of different languages.

So, the **query** becomes **quite large:**

```
( (title_txt_ar:(Children || of || dune))^2.1 ||
(subjects_txt_ar:(Children || of || dune))^2.1 ||
(description_txt_ar:(Children || of || dune))^1.7 ||

(title_txt_en:(Children || of || dune))^2.1  ||
(subjects_txt_en:(Children || of || dune))^2.1 ||
(description_txt_en:(Children || of || dune))^1.7

… all other languages …

allOtherData:(Children || of || Dune))
```

Problem with stop-words and multiple languages

Because data of book is not as consistence: some books are listed as being in German but the **description** or **subjects** are in **English**, or sometimes a **non-English** books contain some bit of **English**, this problem also is present in **non-English** language to **non-English** books. This creates a small problem if a user uses the system with a query containing **stop words non-English** book or **book** with **inconsistent data,** since **those** will have **a higher** score than actual English one; due to stop words not being removed by solr and we know that Stop words also are a lot even in a small text.

Boost on proximity

A query also defines an order of the word, document close the order and phrase should be rewarded more, in my testing this helps to distinguish when a user search for novels genre of book like "science fiction" and the book with scientific content. In fact, without the two get mixed together. To augment the score we sum the score of the **proximity operator** to the **query** (we need to set as base **query operation** to OR `(q.op=OR)`, so **query** becomes:

```
(… (title_txt_en:(children || of || dune)
      + title_txt_en:\"(children) (of) (dune)\"~3))^2.1
|| ((subjects_txt_en:(children || of || dune)
      + subjects_txt_en:\"(children) (of) (dune)\"~3))^2.1

… allOtherData:(Children || of || Dune))
```

The number of position possible offset is **3**, which is the number of word in the query.

Joy Albertini

Small problem with `solr` proximity operator, it seems to give some value to stop-words, in fact if write the in search box "**the**" and then add operator AND language:english you will get 16 results.

Boost on **containing all words**

An entry book that **contains** all **words** of the **query** in a **field** should get a **boost** independently if those words are near **each other** (so not considering proximity):

So**, the query becomes**:

```
(… (title_txt_en:(children || of || dune)
+ title_txt_en:\"(children) (of) (dune)\"~3)
+ title_txt_en:(children && of && dune))^2.1
…
allOtherData:(Children || of || Dune))
```

Query strategy

The query composition is structured to achieve both **high recall** and **precision**, the first document will be precise to the query and also system will retrieve a lot document with similar context to the query so that the user can browse the collection, and the fact that the document are sorted based on similarity with the query helps the user digest the data. There are 3 main components to achieve this in each **field**:

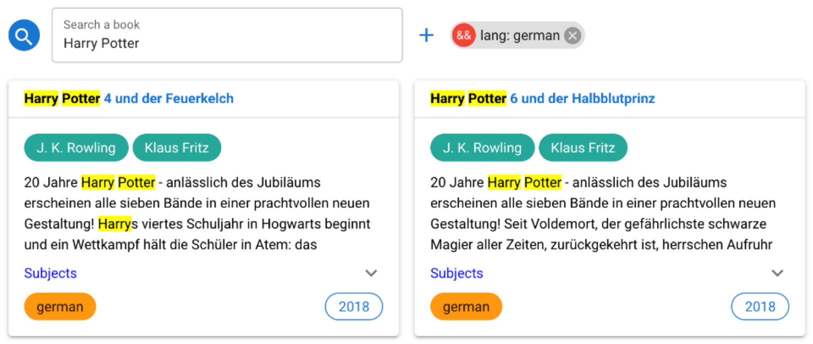
```
1)(title_txt_en:(children || of || dune)
2)+ title_txt_en:\"(children) (of) (dune)\"~3)
3)+ title_txt_en:(children && of && dune))^2.1
```

1. retrieve all books with one of those words, retrieving all books with similar context (define Recall)
2. **increase score** of books with same words of query in sparse order very near to each other (define precision)
3. **increase score** of books that contain all words in the query (define precision)

Query-solr: **operators**

I define as operators as additional query parameter to the query with Boolean logic (basically the feature of the project). You can add an operator to the **base-query**, to constraint further the data retrieved you can use **operators**

Example: Retrieve all book with title Harry Potter and language German


Search a book
Harry Potter
+
&& lang: german

Harry Potter 4 und der Feuerkelch
J. K. Rowling
Klaus Fritz
20 Jahre Harry Potter - anlässlich des Jubiläums erscheinen alle sieben Bände in einer prachtvollen neuen Gestaltung! Harrys viertes Schuljahr in Hogwarts beginnt und ein Wettkampf hält die Schüler in Atem: das
Subjects
german
2018

Harry Potter 6 und der Halbblutprinz
J. K. Rowling
Klaus Fritz
20 Jahre Harry Potter - anlässlich des Jubiläums erscheinen alle sieben Bände in einer prachtvollen neuen Gestaltung! Seit Voldemort, der gefährlichste schwarze Magier aller Zeiten, zurückgekehrt ist, herrschen Aufruhr
Subjects
german
2018

The operators available (to add one click on +)

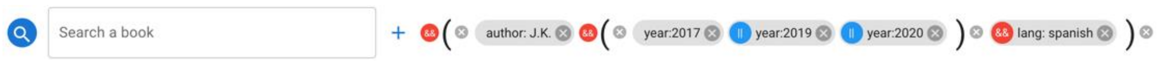
- **Author**: constraint on author
- **Year**: constraint define a range of publication
- **Language**: constraint language
- **(**: Open parentheses for Boolean logic
- **)**: Closed parentheses for Boolean logic

➢ You can add operators as a stack

Joy Albertini

I designed the application such that you can define any possible Boolean query using **AND**, **OR (**, and **(**, **giving most of the power of** `solr` **through a UI,** is basically a reskin of `solr` **query system.**

Example: you can define complex queries such as

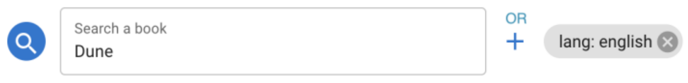


This query will retrieve all books by author **J.K. Rowling** published in 2017 and 2019-2020 in language **Spanish**.

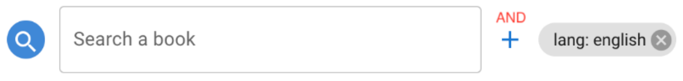
Note: that parentheses are important for the execution of the query correctly.

Operation between **search-Box**

Between an **non-empty** search box and the **first operator** if you don't specify a **Boolean operator**, there will be an **OR** because of base query operator`(q.op=OR`), an **AND** will be more appropriate for user mistake but I needed an OR for **base-query.**



Between an **empty** search box and the **first operator** if you don't specify **a Boolean operator**, will be an **AND**, because will be the only field present in the query, **which is user-friendly.**



Empty Query

An **empty query** will result in **empty results**

Bad Query

With operators a user can define query that are not parsable by `solr`, even though this is not user friendly it offers max possibility to query the system.

Design of the UI:

To implement the UI, I used mainly **VUE** with **VUEX** and **Quasar**, the code can be a bit difficult to understand if you **don't know both systems**, there are a multitude of file in quasar project, I will point out the one that should be your interest:

Design of the UI, and some logic function in Js for the UI:

- `bookSearchUI/quasar-demo/src/components/addModifier.vue:` pop-up menu for adding operators.
- `bookSearchUI/quasar-demo/src/components/overlayPage.vue:` black screen below the pop up add modifier
- `bookSearchUI/quasar-demo/src/components/Books.vue`: main page for book visualization and operator visualization and code to call the store `solr` to query `solr` system with all needed parameters.

On start

When you start the application it will display some random results, I implemented it so that the user can already **see the structure of book** entity and will **be easier to query the system**.

Posted on Github pages

https://joyalbertini.github.io/bookSearchUI/#/

Joy Albertini

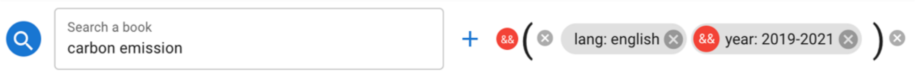
Posted on there so is easier for you to evaluate it, you can simply use that UI on that page by starting your solr server at `localhost:8983`

User evaluation

To perform user evaluation I used google docs, 4 people did the tasks: Joao, Carlo, Ian and Arianna.

I designed **6 different tasks** aim to test different **aspect to user-interface usability**, incorporating all **feature off the application.**

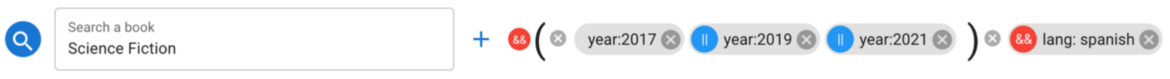
1. **Task 1**: find books about **reduction of carbon emission** in **English** published between 2019-2021, solution:

   Test user ability to constraint the query with **language** and **year,** with **AND** operation**.**

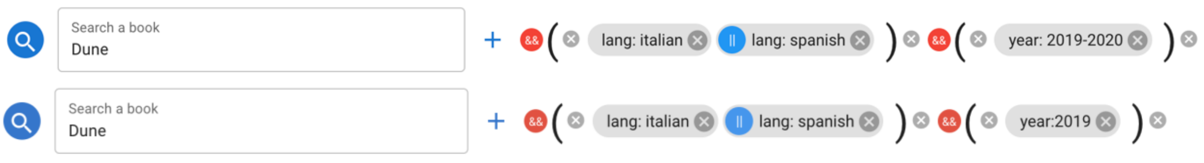
2. **Task2**: Try to retrieve the max number of book regarding calculus in general, you can try 3 different queries, one possible solution: "Calculus Math Algebra"
   Test user ability to find as many document as possible, check if he will exploit or between words.

3. **Task3**: Search for novel **genre** that you like, published in (2017 or 2019 or 2021) and constraint it with a language, solution:

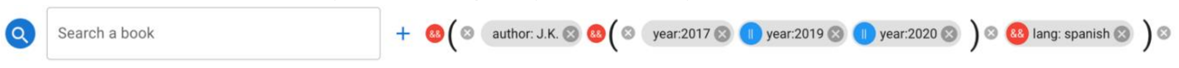
   Test user ability to constraint the query with **language** and AND-OR operation between **years**

4. **Task4**: Query with keyword Dune, with language Italian or Spanish, and with year 2019 or 2020; after that modify the query or add constraint such that you will find a single result, solution:

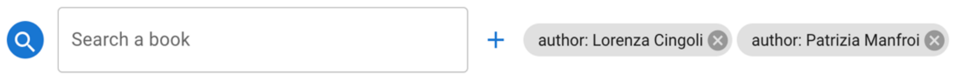
   Test user ability to constraint the query with **language** and AND-OR operation between **years**

5. **Task5**: Find books written by J.K Rowling in Spanish from years 2017 or 2019 or 2020.

   Test user ability to constraint with complex constraint, and particular **author.**

6. **Task6:** Find books with author Lorenza Cingoli and Patrizia

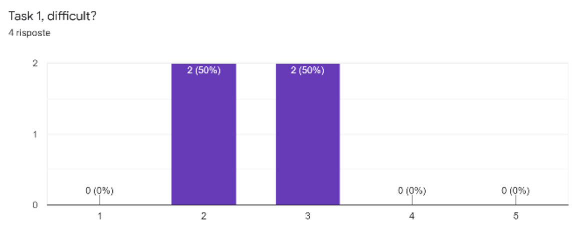
   

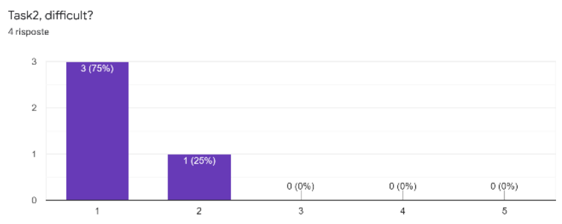
   Test user ability to constraint with multiple **authors.**

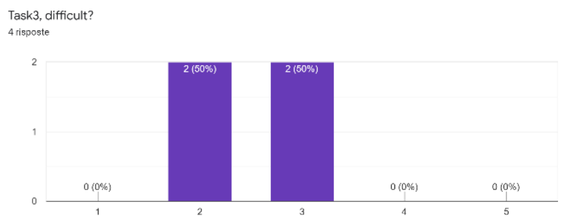
Each task I got feedback by the user in both **difficulties to perform the task** and **relevance** of then **results from the user, using a liker scale of 1: best result --** to **5 worst result**

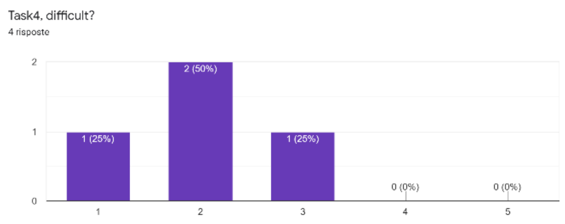
Difficulty in performing the task by user:

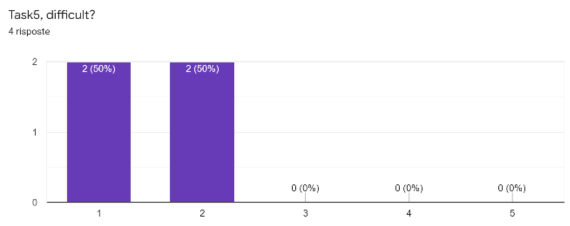
Joy Albertini

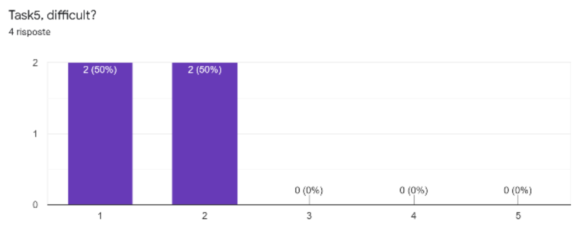




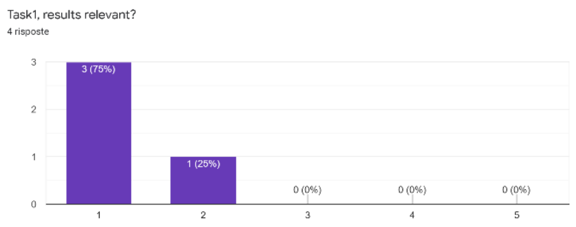


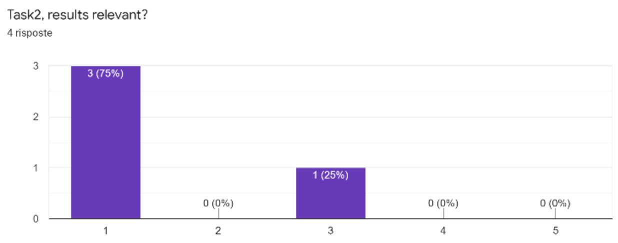


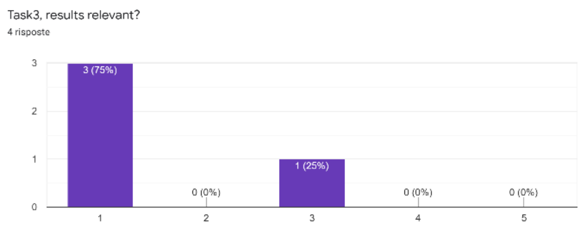


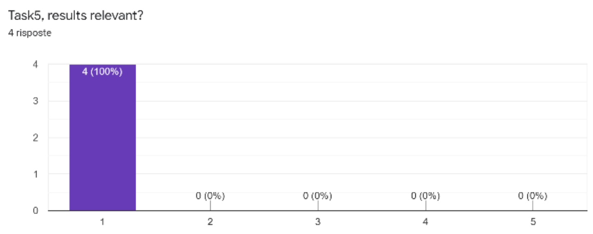


Relevance of the retrieved result by user:

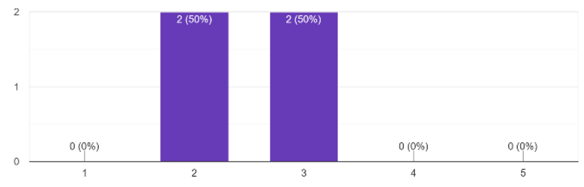




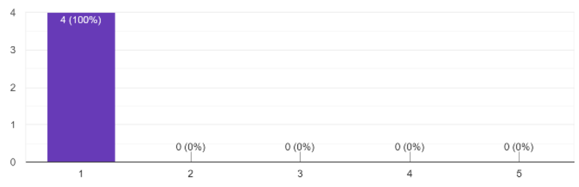




Usability of the UI

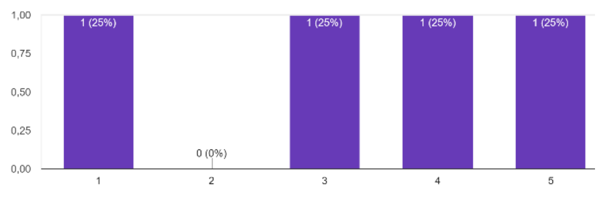


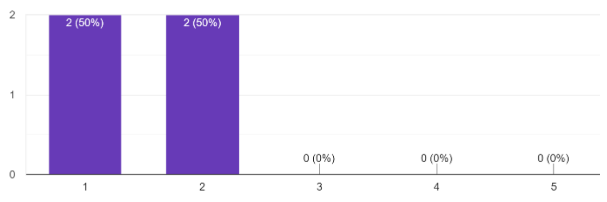
Design of the UI



Usability of boolean constraint (operators)

People you use my book search system





Observation from users:

- operators too complex (as you can see also from the charts) without explanation how to use it.
- Add an info section of how to use operators.
- Add drag and drop between operators and make them replaceable on click.
- Simplify the operators section, removing AND OR possibility
- Make clearer what happen between search-box and first operator.
- Add page number so that I can quickly index the page I want.
- Task proposed are too long.
- Books are mostly relevant to user, confirmed also by the charts about the relevance of the retrived data.

Observation of users:

I mainly see that people have difficulty understanding that search-box and operators are connect, and they need to specify a Boolean operator between them, also some difficulty come from Boolean operators to some they seem difficult to use. Another is that users tend to not use parentheses which are important. Users also struggles a bit in year range when they have to specify a single year instead of range.

One important observation is that I see that people struggles a lot in the first task but quickly learn how to query using operators, this is confirmed by charts about difficulty , the first ones have the lower score, instead the last two have an higher score, and task 5 is the most difficult one, so the learning curve is quite good.

Remark of user observation.

Some of the criticism I expected it, about the operator, I also considered to add drag and drop to operators but is not possible in quasar.

Further improve my system.

To be more user friendly:

- I could constraint the Boolean operators, for example removing the possibility to specify multiple years when staked together, because it doesn't make sense a book entity have only one year of publication.
- Add a How to use section.
- Add the possibility to replace and drag and drop between **operators.**

Improve operators possibility:

- We could add the **NOT** Boolean operator.
- Add operators for **ISBN** and **Editor** search.

Add language detection to the query such that you can exclude result in on other **language automatically**, in my system because they are **ored** sometimes some result in other language different from query pop-up mainly due to data inconsistency of sites or the problem with stop-words described previously. Also, this will **prevent the query form becoming too long**.

Implement the query using POST instead of GET, such that the problem of URI too long will never occur.