AcademicQuest
Information Processing and Retrieval

Ana Beatriz Fontão
up202003574@up.pt
FEUP
Porto, Portugal

Daniel José Mendes Rodrigues
up202006562@up.pt
FEUP
Porto, Portugal

José Luís Rodrigues
up202008462@up.pt
FEUP
Porto, Portugal

Martim Raúl da Rocha Henriques
up202004421@up.pt
FEUP
Porto, Portugal

ABSTRACT

One of the major challenges students face is deciding which degree to pursue in higher education. The vast amount of options can seem daunting. To make matters worse, it doesn't help that this information is scattered on the web. Each university presents its information in a different fashion, making it hard even to find out which courses make up each degree. AcademicQuest makes it easy to access this information, as it provides an easy-to-use search system that can answer questions such as "Where can I learn marketing?" and "Which degree will make me a journalist?". This is done with the aid of web-scraping on the university website and posterior organization of the information.

ACM Reference Format:

Ana Beatriz Fontão, Daniel José Mendes Rodrigues, José Luís Rodrigues, and Martim Raúl da Rocha Henriques. 2023. AcademicQuest Information Processing and Retrieval . In *Proceedings of (Group 81).* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In the domain of higher education, it is no wonder that the vast array of options can be overwhelming for students when it comes to choosing the right degree program. This critical decision-making process necessitates a reliable and efficient means of accessing information on the courses and degrees offered by various institutions. This is where AcademicQuest comes into play.

The system seeks to alleviate the challenging task of degree selection by providing an accessible and user-friendly solution. Through a thorough web scraping process of university and faculty websites, we gathered all the main information of degrees, such as objectives and curricular plan, course units, like program and pre-requirements and professors, names and contacts, among many other relevant topics. Regarding our most recent work, we established a consistent pipeline for data processing to ease the subsequent phases of the project.

The successful completion of this milestone will result in a well-prepared, comprehensive, and structured dataset that will serve as the bedrock for our project.

2 DATA SOURCES

One can imagine there is no central database with academic information. To get the necessary information for the project, we could only rely on scraping university websites. Because of this project's scope, we limited this task to the University Of Porto for this milestone. Retrieving data for different universities can be challenging as there is no standard display of information.

The data sources we selected and prepared are the foundation of our project. This milestone revolves around the crucial task of preparing and characterizing these datasets. As of now, information comes from two main sources:

- https://www.up.pt
- https://www.sigarra.up.pt

The first website allowed us to get all the reference links of the degrees from the University of Porto, which from this point forward were used for the degree description, course units, and professors, specifically of the Faculty of Engineering.

The information about each topic usually resides in Sigarra [2] on specific pages. The base URL for the Sigarra website is `https://sigarra .up.pt/faculty/pt/`, where `faculty` is the degree's specific faculty (e.g. FEUP, FBAUP). Then, following this link, these are the completing segments to reach our desired pages:

- Degree: `/cur_geral.cur_view?pv_curso_id=degree`, where `degree` is replaced by the ID of the degree
- Course Unit: `/ucurr_geral.ficha_uc_view?pv_ocorrencia_id=unit`, where `unit` is replaced by the ID of the course unit
- Professor: `/func_geral.formview?p_codigo=professor`, where `professor` is replaced by the ID of the professor

3 COLLECTION AND PREPARATION

As mentioned above, the data needed for this project will require information about degrees, comprising course units and professors. In the case of the University of Porto, this information resides on faculty websites. This means that our dataset was obtained

Group 81,
,

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

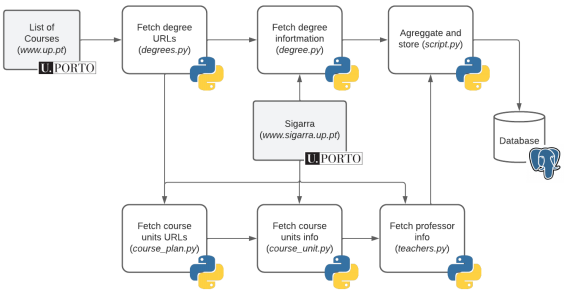


Figure 1: Data collection pipeline diagram

by scraping these pages. This section outlines the data collection process and how we structured its pipeline.

3.1 Pipeline

The collection process begins at the UP website, where the list of all degrees is fetched. Then, the collected URLs are used to scrape information about each degree on its official page, as well as retrieve its curricular plan or course list. Course-specific information can then be parsed from each course page. Finally, professor pages extracted from the course are also parsed. The web-scraping is done using Python [14] and BeautifulSoup [12] and the data is finally stored in a PostgreSQL database. Please refer to Figure 1 for an illustration of this process.

3.2 Conceptual Data Model

To better organize our data, we decided to use a relational system. This made sense as our entities are closely related to each other and avoiding SQL would result in a lot of unnecessary redundancy.

The conceptual model is depicted as a UML diagram in Figure 2. The key relations in usage are Degree, Course Unit, and Professor. University relation is not of much relevance as all the data comes from the same college for now. The attributes for each were derived from the information available online.

4 CHARACTERIZATION

In this section, we will describe the data collected and its characteristics. There will also be displayed some statistics and discussed the quality of the data acquired.

4.1 Collection characterization

As previously mentioned, we collected data about degrees, course units and professors from the Sigarra website.

The data we gathered is only about Porto University since we found that the amount of data would be enough for the dimension of this project. The ones explored in more detail (course units and professors) were the ones corresponding to the engineering faculty (FEUP).

This data also only refers to degrees and course units that are currently in operation. On top of that, the professors we collected information from are the ones connected to the current course units and degrees.

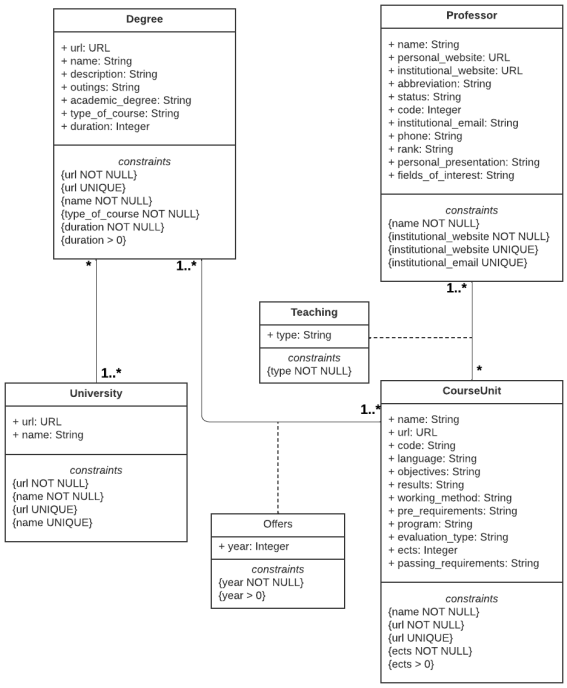


Figure 2: Conceptual Data Model UML Diagram

Regarding the amount of data collected, here are the quantities of data scraped:

- The number of degrees is 224
- The number of course units is 5096
- The number of professors is 3057

When it came to the quality of the data, a few problems emerged.

During the scraping process of an individual course unit page, we noticed that the HTML was very disorganized and lacking consistency. For example, most sections aren't identified by a class or id, meaning that we almost always had to use the method 'find' method with the title (in Portuguese) of the section to extract their information. Due to this irregularity, some fields that, even though had text in them, weren't correctly extracted since the code developed wasn't able to process the varied structures that section may have (e.g. section 'Programa' from the course unit's page). This may bring some issues later on since it relies heavily on hard-coded strings. A similar situation happened when extracting the course unit URLs.

4.2 Document Presentation

We gathered, from each page, the information we found more relevant to the purpose of our platform.

As referred to earlier, the data is stored in a PostgreSQL database [4]. In figure 2, the classes and their attributes can be seen. Some of these attributes contain textual data and others simple values.

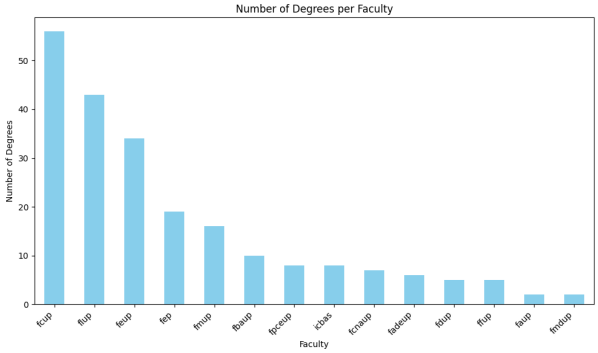


Figure 3: Number of degrees per faculty

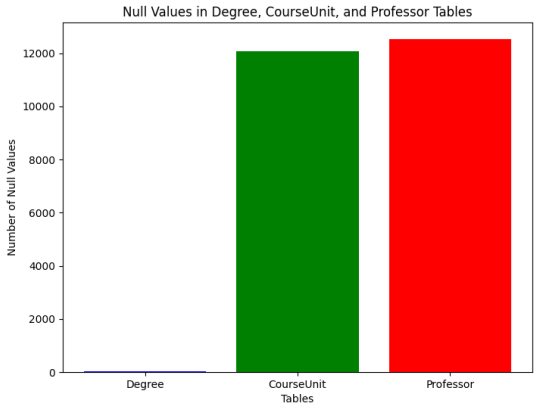


Figure 4: Total number of null values in each table

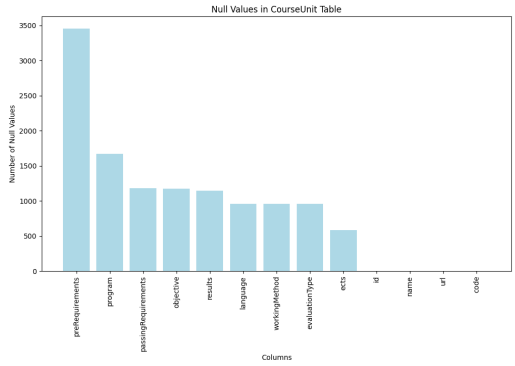


Figure 5: Distribution of null values in course unit table

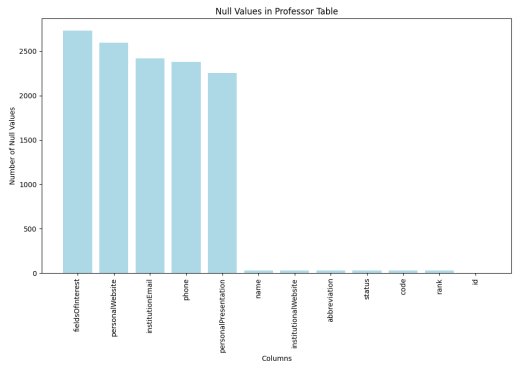


Figure 6: Distribution of null values in professor table

4.3 Descriptive and exploratory statistics

To better understand the information collected for this project, we assembled a few plots and visual representations of the data.

Figure 3 shows the amount of degrees per faculty. This analysis allows us to better understand the distribution of degrees over the various faculties.

In figure 4, we can see the number of null values in the three main tables of the database: degree, course unit and professor. As expected, the degree table is the one with the lowest number of null values. This is because the degree pages contained less information than other pages, which resulted in fewer columns in the table. On top of that, and as seen in section 4.1, the amount of data gathered about degrees is much smaller than the other two tables.

We found that the most relevant tables to study in terms of null values were the course unit and the professor.

In the next plot (figure 5), we can see that many more columns contain null values. The only ones that always have a value are the ID, name, code and URL. The remaining columns all contain null values. These columns mainly correspond to textual sections of an individual course unit's page. The large amount of missing values in these fields is due to the absence of said section in the pages. Additionally, as mentioned in section 4.1, many times these pages have very different layouts, so it's possible that the code was unable to adapt to certain pages that had a more irregular structure.

The following plot, displayed in figure 6, shows the number of null values over the columns of the professor's table. The attributes with the greatest amount of null values are `fields_of_interest`, `personal_website`, `institutional_email`, `personal_presentation` and phone. Since professors likely have the option to decide what information is displayed on their pages, these components are frequently missing from professor pages. Nonetheless, we found these attributes important when they are present, so the table includes them.

When analyzing the quality of the data gathered, we found that:

- There are some courses with the same name. These names are very generic (e.g. 'Matemática II' or 'Geografia'), so it may be correct information and not a bug.
- The same thing happened with the names of professors. Although all the `instituinal_website` values in the database are unique, there are some abbreviations and names that are not. This can be related to a professor having multiple personal pages in Sigarra.

Now, examining the size of relevant degree information, such as its description and the possible outings. For this analysis, we created

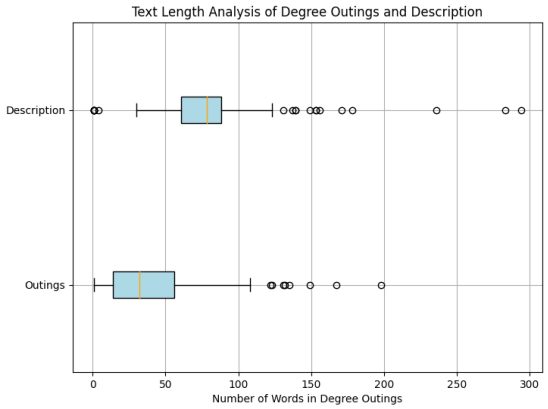


Figure 7: Text length of degrees outings and description

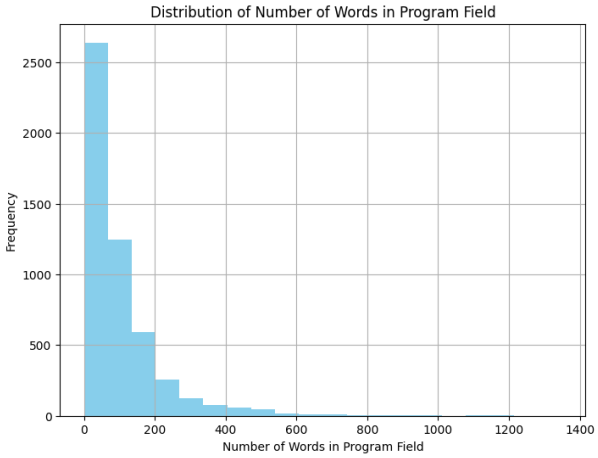


Figure 8: Text length of course units' programs

two box plot plots that display the number of words that appear in each of this section. The text length of degrees' descriptions and outings are presented in figure 7. The average amount of words in these sections is, approximately, 90 and 40, respectively. This large amount of information may become confusing for future college students, adding another reason with this project is so useful.

In figure 8, the number of words per course unit program is shown. It is obvious, by observing the graph, that the majority of course units have a relatively short program (between 0 and 200 words). These numbers still represent multiple phrases that a possible student would have to read and reflect on. Given the large number of degrees and course units, reading all these programs would still be a tedious task.

4.4 Text Analysis

We created a word cloud that shows the most common words found in the possible outings section of the degrees' pages. It is displayed

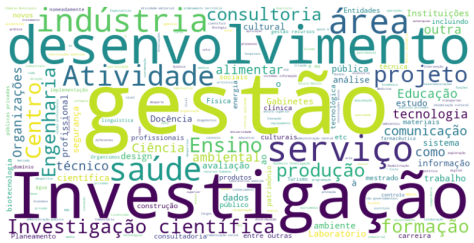


Figure 9: Word Cloud for outings of all courses

in figure 9, and we can see that the most prominent words are investigation, management, development and service.

5 PROSPECTIVE SEARCH TASKS

As previously stated, the main problem our project set out to solve is to help students with hard decisions regarding their academic paths. Thus, the interface and prompt structure should be oriented with a student's perspective in mind. Besides, the system should make sense for different levels of education. Maybe a high schooler is looking for a bachelor's or a graduate student is picking a master's. It makes sense that the system can cater to both in distinct ways.

This sort of decision requires different perspectives, i.e. there are a lot of factors to consider [1]. We will now outline a set of different information needs and scenarios that can illustrate the use of the system.

Job Opportunities Many students will grow up with a list of professions they would want to become as they grow older. Maybe by the time they get to college, some have outlined a single or a couple they are more certain about. Regardless, the path to this dream job may not seem clearly cut and it will then be useful to use such a system that can outline academic paths according to foreseen profession. Luckily, many universities include professional outings on their website. AcademicQuest uses this information and brings forward a type of prompt specific to this problem, allowing for career-oriented searching.

Subjects Another popular strategy for choosing a degree is to look for areas the students have shown interest and aptitude for. Consequently, it makes sense that the system should enable subject-oriented exploration. This can be applied to fields addressed in the curricular plan as well as specific admission requisites (e.g. required exams). In further detail, there is also value in searching for more specific topics, such as a student looking to find out which technologies they can expect to use and learn during a degree.

Duration As some people can be more eager to get into the job market, filtering with duration in mind will help in exploring varied options.

Cost Tuition cost may be a decision factor between alternatives and should thus be taken into account.

Language It is the case that some degrees may be more oriented to international students, while at the same time ostracizing others not so comfortable with foreign languages. Thus, it could be valuable to bring forth this sort of information.

6 INDEXING

With all the necessary data gathered, the remainder of this document will focus on the information retrieval stage. The tool chosen for this task was **Solr** [7], as it suited our information needs while being part of the curriculum. This section describes the indexing of documents.

6.1 Documents

Our adoption of a relation model in the previous stage required an extra transformation step before being able to work with the chosen tool. From the initial domain's relations, the University can be discarded at this stage as all of our data is coming from the same institution. This leaves us with Degrees, Courses, and Professors. All three are deeply relational, i.e. courses belong inside degrees and professors belong inside courses. If we take into consideration that all of these will be queried separately, organizing them in nested documents would be a bad choice, as searches could become convoluted.

Hence, three Solr cores were created, to achieve separation of concerns and thus improve overall query performance.

6.2 Field type

Four types of fields were created, each tailored to manage specific types of data:

- **stringText**: Used to handle regular text, with a few filters added (more on those later) to enhance token matching and user experience. Tokenization is done using the "standardTokenizesFactory", which tokenizes the text using white spaces and punctuation. A "TextField" Solr default field type is utilized.
- **stringCode**: The goal of this field type is to handle codes or acronyms. Using the "NGramTokenizerFactory," tokenization allows users to look for partial continuous character sequences in their queries. A "TextField" Solr default field type is utilized.
- **int**: This field type acts as syntactic sugar, storing simple integer values just by using the Solr default type "IntPointField".
- **urlEmail**: This field type is intended for handling emails or URLs. It employs the "UAX29URLEmailTokenizerFactory" for tokenization, ensuring that email addresses or URLs remain intact and aren't split into separate tokens. Similar to stringText and stringCode, it utilizes the Solr field type "TextField".
- courseVector: This field type manages the embeddings (dense vectors) employed in the semantic search. It utilizes the Solr field type "DenseVectorField" and defines the values for the parameters "vectorDimension," "similarityFunction," and "knnAlgorithm" as 384, "cosine," and "hnsw," respectively.

Additionally, as shown in Table 1, the following filters were used for each type of field :

- **ASCII Folding:** Converts Unicode characters outside the Basic Latin Unicode block to their corresponding ASCII counterparts;
- **Lower Case:** Converts uppercase tokens to lowercase equivalents.
- **Suggest Stop:** Discards tokens found in the provided stop words list generated in Portuguese using Natural Language Toolkit (NLTK) [9] Python library.
- **Synonym Graph:** Enables users to search without requiring specific words from the documents by implementing synonym mapping. This feature is made possible by Portuguese synonym lists generated using NLTK.
- **Flatten Graph:** Essential in index time for the Synonym Graph Filter.
- **Remove Duplicates:** Eliminates duplicate tokens within the stream, crucial for handling potential duplicates created by the synonym filter.
- **Snowball Porter Stemmer:** Recognizes and returns the "stem" words.
- **Hyphenated Words:** Reconstructs hyphenated words tokenized as two tokens due to line breaks or white space, joining tokens ending with a hyphen and discarding the hyphen itself.

Filter	stringText	stringCode	int	urlEmail
ASCIIFolding	Yes	Yes	No	Yes
LowerCase	Yes	Yes	No	Yes
SuggestStop	Yes	No	No	No
SynonymGraph	Yes	No	No	No
RemoveDuplicates	Yes	No	No	No
SnowballPorter	Yes	No	No	No
HyphenatedWords	Yes	No	No	No
FlattenGraph	Yes	No	No	No
courseVector	No	No	No	No

Table 1: Applied filters across each field types

6.3 Schema Details

The field names, types, and indexing status for the cores Degree, Professor, and Course Unit are shown in Tables 2, 3, and 4. The decision to index these fields relied on the team's analysis of queries and potential search tasks, specifically focusing on whether user queries would involve searching within these fields.

Name	Type	Indexed
id	stringText	True
url	urlEmail	False
name	stringText	True
description	stringText	True
outings	stringText	True
typeOfCourse	stringText	True
duration	stringText	True
vector	courseVector	True

Table 2: Schema Fields Overview: Degree Core

Name	Type	Indexed
id	stringText	True
name	stringText	True
personalWebsite	urlEmail	False
institutionalWebsite	urlEmail	True
abbreviation	stringCode	False
status	stringText	False
code	int	False
institutionalEmail	urlEmail	False
phone	stringText	False
rank	stringText	True
personalPresentation	stringText	True
fieldsOfInterest	stringText	True
vector	courseVector	True

Table 3: Schema Fields Overview: Professor Core

Name	Type	Indexed
id	stringText	True
name	stringText	True
url	urlEmail	False
code	stringCode	False
language	stringText	True
ects	int	True
objectives	stringText	True
results	stringText	True
workingMethod	stringText	True
preRequirements	stringText	False
program	stringText	True
evaluationType	stringText	True
passingRequirements	stringText	False
vector	courseVector	True

Table 4: Schema Fields Overview: Course Unit Core

7 RETRIEVAL

Following the indexing of the documents, this section presents the queries created to suit the information needs in section 5. These were created using Solr's Standard Query Parser and made use of functionalities such as boosts, fuzziness, and proximity search. We found that the usage of these modifiers was slightly less relevant than the Stemmer filter described in the last section. Nevertheless, both features worked better together to deliver a more comprehensive set of results.

7.1 Biology-related degrees that will allow me to work in research

parameter	value
q	(name:biologia~)^3 OR (description:biologia~)
fq	outings:investigação
sort	
fl	

To broaden the result space, both `name` and `description` were queried. In addition, a boost was used to highlight the more relevant term name.

The use of fuzziness (tilde in Solr) made sure we included results in the biology domain defined with similar words (e.g. Biológico).

7.2 Master degrees related to high-school teaching sorted by ascending duration

parameter	value
q	(name:"ensino secundario"~5)^3 OR (outings:"ensino secundario"~5)^2 OR (description:"ensino secundario"~5)
fq	typeOfCourse:Mestrado + typeOfCourse:"Mestrado Integrado"
sort	duration asc
fl	

This query makes use of proximity search as it would not retrieve relevant information such as "Ensino Básico e Secundário" otherwise.

7.3 Mathematical Logic-related Course Units taught in Portuguese

parameter	value
q	(name:logica OR name:matematica) AND (program:matematica OR program:logica)
fq	language:portugues
sort	
fl	

To obtain the desired result, `name` and `program` were queried to check whether `logica` or `matematica`. The data is filtered by the ones that have `Portuguese` in `language`.

parameter	value
q	(name:metodos AND name:estatistica)^5 OR (program:metodos AND program:estatistica)
fq	
sort	ects desc
fl	

7.4 Curricular Units that address statistical methods sorted by descending ects

A simpler query that searches for references of `Métodos` and `Estatítica` in either `name` and `program` and sorts the results in descending order of ects.

7.5 Active Professors whose field of interest is engineering and environment

parameter	value
q	(fieldsOfInterest:engenharia OR (fieldsOfInterest:ambiente)^5) AND status:Ativo
fq	
sort	
fl	

To increase value to the word 'ambiente' we added weight to it. This will prioritize this independent term over the remaining ones.

7.6 Auxiliary Professors who teach at FEUP

parameter	value
q	(institutionalWebsite:*feup* AND rank:"* Auxiliar")
fq	
sort	
fl	

To obtain the professors from FEUP (or any other faculty), the institutional website field was checked. This is the only field that contains faculty information. On top of that, there are two ranks of auxiliary professors: auxiliary professors and auxiliary investigators. This query includes both of these cases.

8 EVALUATION

Information needs to be defined in section 5 are an indication of the expected behaviour of the system. To assess the quality of the setup, we will try to understand how the queries deliver the relevant required information.

To accomplish the latter, two different systems will be tested: a **Basic** design and its **Enhanced** version. The latter matches the schemas described in section 6 and the query parameters in section 7. To create the basic version we removed boosts, fuzziness, and proximity search from queries. Likewise, stemming was removed from the schema for all the queried fields. This configuration will allow the assessment of the relevance of the mentioned modifiers.

The evaluation process is done in a semi-automatic manner. Humans have to manually identify relevant results for each query so a Python script can compute its performance. Naturally, the detection of such results in a system with thousands of entries is not an easy task. This was solved by using a sample of 200 random entries for each core being tested. Then, a `qrels` file would be filled with relevant results. As a rule of thumb, queries should yield at least 10 results, to suit the measurements defined below.

Lastly, performance was assessed using the following metrics:

Average Precision - Average Precision is a comprehensive measure that considers the precision at each relevant document rank, providing an overall assessment of the system's effectiveness in retrieving relevant information.

P@10 Precision at 10 focuses on the relevancy of the top 10 retrieved documents, offering insights into the system's ability to deliver pertinent results within the initial set of items.

R@10 - Recall at 10 evaluates the system's capacity to retrieve a significant proportion of relevant documents within the top 10 results, highlighting its recall performance in a more concise set.

F1@10 - The F1 score at 10 harmonizes precision and recall, providing a balanced evaluation of the system's performance, especially in scenarios where achieving both high precision and recall is crucial, such as information retrieval.

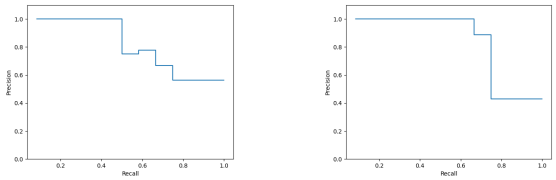
These metrics collectively offer an idea of the effectiveness of the system. Furthermore, the precision-recall curve, which measures the trade-off between the two, is also plotted.

To evaluate these systems, a subset of two hundred random entries of each component was generated. Manually, we extracted some of the most relevant results and registered their IDs. This information is stored in the q_rels.txt file for each query. We opted to create a smaller subset of data since it was much simpler to locate pertinent data manually in this manner.

The remainder of this section describes the results obtained for each of the queries defined in section 7.

8.1 Query results

8.1.1 Biology-related degrees that will allow me to work in research.



(a) Basic system **(b) Enhanced system**

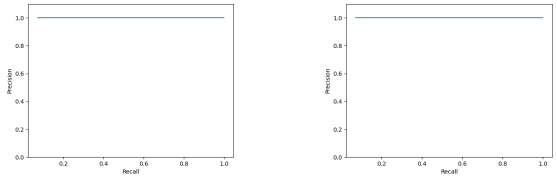
Figure 10: Query 1 Precision-Recall curves for both setups

Table 5: Query 1 metrics compared for both systems

Metric	Basic	Enhanced
Average Precision	0.918898	0.988889
Precision at 10 (P@10)	0.800000	0.900000
Recall at 10 (R@10)	0.666667	0.750000
F1 at 10 (F@10)	0.727273	0.818182

It was no wonder to observe that the enhanced version achieved greater results than the basic one. We can conclude based on these outcomes that features such as boosts, fuzziness, stemming and proximity search have a positive impact in the product of the query.

8.1.2 Master degrees related to high-school teaching sorted by ascending duration.



(a) Basic system **(b) Enhanced system**

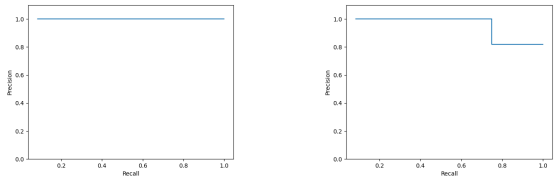
Figure 11: Query 2 Precision-Recall curves for both setups

Table 6: Query 2 metrics compared for both systems

Metric	Basic	Enhanced
Average Precision	1.000000	1.000000
Precision at 10 (P@10)	1.000000	1.000000
Recall at 10 (R@10)	0.714286	0.714286
F1 at 10 (F@10)	0.833333	0.833333

However, some weird results may happen, as we would not expect to see the same result in both systems, even more so that they reveal 1.000000 Average Precision and Precision at 10 (P@10). Specifically for this case, both systems obtain all the relevant results that we highlighted in the subset for this query.

8.1.3 Mathematical Logic-related Course Units taught in Portuguese.



(a) Basic system **(b) Enhanced system**

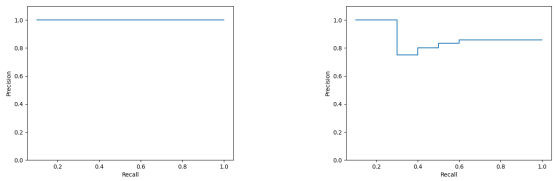
Figure 12: Query 3 Precision-Recall curves for both setups

Table 7: Query 3 metrics compared for both systems

Metric	Basic	Enhanced
Average Precision	1.000000	1.000000
Precision at 10 (P@10)	0.700000	0.900000
Recall at 10 (R@10)	0.583333	0.750000
F1 at 10 (F@10)	0.636364	0.818182

Similarly to the last query, both systems obtained very high results. And as seen in the table of the results, the enhanced version managed to achieve better outcomes.

8.1.4 Curricular Units that address statistical methods sorted by descending ects.



(a) Basic system **(b) Enhanced system**

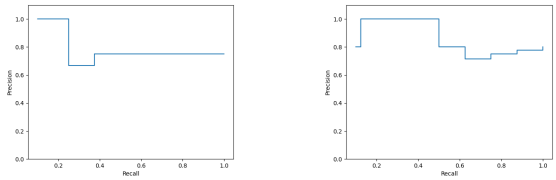
Figure 13: Query 4 Precision-Recall curves for both setups

Table 8: Query 4 Metrics compared for both systems

Metric	Basic	Enhanced
Average Precision	1.000000	0.915079
Precision at 10 (P@10)	0.400000	0.600000
Recall at 10 (R@10)	0.400000	0.600000
F1 at 10 (F@10)	0.400000	0.600000

Once again, the results of the systems are very similar, even a bit unexpected. Although the enhanced system got a lower result on the precision metric, its overall score was higher than the simple system. These results might suggest that the basic system may excel in retrieving relevant information across the dataset, but might not be as effective in presenting the most relevant results within the top 10. Or, the enhanced system, by sacrificing a bit of overall precision, does a better job of identifying highly relevant results within the top 10. The difference is very insignificant in this case, so we consider the enhanced system a better option.

8.1.5 Active Professors whose field of interest is engineering and environment.



(a) Basic system (b) Enhanced system

Figure 14: Query 5 Precision-Recall curves for both setups

Table 9: Query 5 Metrics compared for both systems

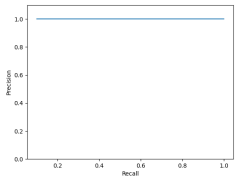
Metric	Basic	Enhanced
Average Precision	0.916667	0.895139
Precision at 10 (P@10)	0.300000	0.800000
Recall at 10 (R@10)	0.375000	1.000000
F1 at 10 (F@10)	0.333333	0.947368

For this query, the results obtained (table 9 and figure 14), were expected. Overall, most results obtained by the enhanced system outperform the basic system.

To, hopefully, improve the user experience and results, we reformulated this query to include a phrase match element. The new query can be found in table **??**.

parameter	**value**
q	(fieldsOfInterest:"Tecnologia ambiental" OR (fieldsOfInterest:"ciências do ambiente")^5) AND status:Ativo
fq	
sort	
fl	

For this query, we integrated phrase match (by using quotes on specific parts) and independent boost (on the second field of interest). In table **??** and image 15 we can see the result on the chosen metrics.



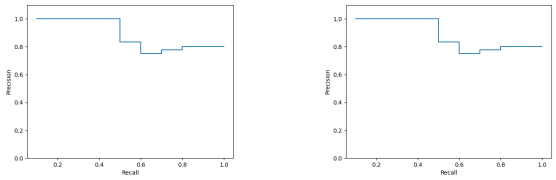
(a) Enhanced system

Figure 15: Query 5 Precision-Recall curves for new setup

Table 10: Query 6 Metrics for the new query

Metric	Enhanced
Average Precision	1.000000
Precision at 10 (P@10)	0.700000
Recall at 10 (R@10)	0.600000
F1 at 10 (F@10)	0.444444

The new results obtained were lower than the original ones, but also satisfactory. This might be since the query isn't the same one, just similar.



(a) Basic system (b) Enhanced system

Figure 16: Query 6 Precision-Recall curves for both setups

Table 11: Query 6 Metrics compared for both systems

Metric	Basic	Enhanced
Average Precision	0.929365	0.929365
Precision at 10 (P@10)	0.800000	0.800000
Recall at 10 (R@10)	0.800000	0.800000
F1 at 10 (F@10)	0.800000	0.800000

8.1.6 Auxiliary Professors who teach at FEUP. This query presents very similar results with both systems. These results can be found in table 11 and figure 16. It is a very direct query, and that might be the reason the values obtained are similar.

The responses to this query include two ranks of professors (auxiliary professors and auxiliary investigators), but, as one of these ranks has such low frequency in our data, this aspect did not have a notable effect on the results.

8.2 Discussion

If we take a look at the table 12, we can see that the results obtained for the mean average precision are better in the enhanced system than in the basic system. This was expected since the enhanced system has more features that provide more relevant and overall better results. This system broadens the results space, by accepting more distant words from the queries, while the basic system is very specific and straightforward. The downside is that more results that do not belong will also be added, thus diminishing precision. Although fewer results are related to higher precision, it doesn't mean that the search system will perform better, as can be observed by the higher recall for the Enhanced system.

For future work, a goal is to improve these results by refining the filters, such as increasing the tolerance to misspelt words or the use of synonyms. On top of that, study how to correctly weight fields to better match the user's inquiries.

Table 12: Mean Average Precision

Metric	Basic	Enhanced
Mean Average Precision	0.955	0.956
Mean Recall	0.586	0.770

9 IMPROVEMENTS

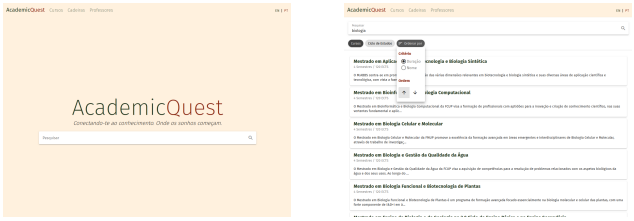
The last milestone included making improvements to the queries created for the second delivery. These changes are documented in Section 7.

The remainder of the improvements performed are described throughout this section.

9.1 Interface

The work carried out during this phase of the project included the development of a Graphical User Interface. This system is targeted to the end user and connects all the information gathered and retrieval methods created to a web application.

To create the program, Vue.js [11], along with Vuetify [15] and TailwindCSS [13], were used for the front end. Moreover, Django [10] was used to interact with both the Postgres database and Solr.



(a) Home Page **(b) Search Example**

Figure 17: Interface Example Figures

9.1.1 Features. The implemented features of the system are defined as follows:

- Perform specific searches for each core
- Filter results by specific parameters for each core
- Sort results by specific parameters for each core
- Related documents suggestion
- Entity linking

9.2 Semantic Search

Semantic search captures the semantic context of the query, allowing the user to formulate a question without sticking to a predetermined style. To accomplish this, the dense vectors, or embeddings, from the query text and the documents must be computed. For this, the 'all-MiniLM-L6-v2' [3] model was employed. Every field in the documents was joined together to create a single string, which will be supplied to the model. The nearest neighbors algorithm is then used to determine which dense vector is closest to the dense vector associated with the query.

For the GUI, the user uses both semantic search and the previously specified queries at the same time when searching in our system, giving priority to actual matches.

9.3 More Like This

The University of Porto offers over 200 courses, including bachelor's and master's degrees, across various fields, which sometimes leads to an overlap in study areas. To assist new students in their decision-making process, a Related section has been created, where one can find similar courses or paths for the next level of study (master's degree). Additionally, this section can also be found on the Professor and Course Unit pages.

For each core, a request handler, `/mlt`, has been configured, taking advantage of Solr's 'More Like This' class [8]. A subset of features was selected to calculate the similarity between two documents, and different weights were assigned based on their relevance:

- Degree
  - Features Used: name, description, outings
  - Respective Weights: 5, 1, 3
- Professor
  - Features Used: fieldsOfInterest, personalPresentation
  - Respective Weights: 5, 1
- Course Unit
  - Features Used: name, objectives, results, program
  - Respective Weights: 5, 1, 2, 4

Additionally, some parameters common to all were configured:

- "maxdf": 10.
  - Ignores terms that occur N times in documents. This allows ignoring frequent terms that might introduce noise when calculating documents. For example, 'ensino superior' often appears in professional outings but contributes little to document similarity.
- "mintf": 1.
  - Determines the threshold frequency below which terms will be disregarded in the source document. A value of 1 was used to ensure that more specific terms were captured and used when calculating similarity.
- "mindf": 2.
  - Determines the minimum frequency threshold for terms to be disregarded if they appear in fewer than this specified number of documents. This parameter follows the logic of the previously described point."

9.3.1 Evaluation. To evaluate MLT, 5 documents from the degree's core were chosen at random. The search was performed against these degrees, which yielded 10 results per document. It is important to note that this is a considerably small sample space, and as such, the results should be interpreted with caution. Each of these results was evaluated using Graded Relevance, according to the following scale:

1 Not related at all
2 May have some fields in common

3 Somewhat overlapping areas
4 Related field
5 Very related or same area

Figure 18 displays a chart on how the results for each of the five randomly chosen degrees vary in relevance.

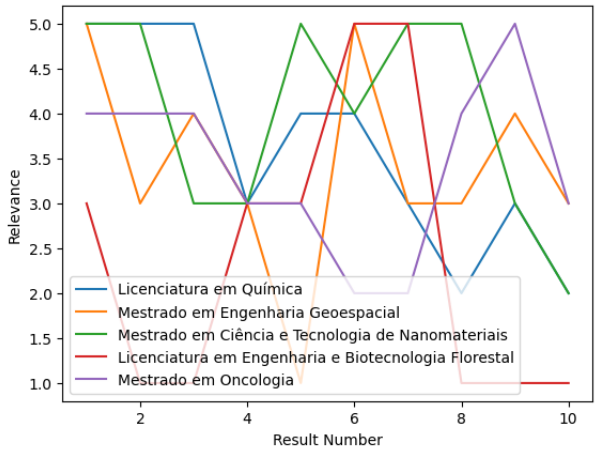


Figure 18: MLT relevance for first 10 results.

Moreover, normalized Discounted Cumulative Gain (nDCG) was calculated. This measure takes into account both the relevance and the position of each item in the list, to convey a comprehensive assessment of the system. Table 13 displays the values obtained for this evaluation. A mean nDCG of 0.84 suggests that, on average, the ranking system is achieving high-quality ranking orders across the instances.

Table 13: Normalized Discounted Cumulative Gain for MLT

Document	nDCG
Licenciatura em Química	0.99
Mestrado em Engenharia Geoespacial	0.89
Mestrado em Ciência e Tecnologia de Nanomateriais	0.94
Licenciatura em Engenharia e Biotecnologia Florestal	0.59
Mestrado em Oncologia	0.81
Average	0.84

9.4 Entities

Among subjects, locations, and faculty names, the knowledge base is filled with entities. These are words related to real-world objects or concepts. Entity linking was used to aid clarification of certain words in the database.

The setup of this system went as follows:

(1) Chose relevant fields from each core
(2) Prepared the text by doing some basic cleanup
(3) Used the Spacy Model [6] to identify keywords in the text
(4) Cross-referenced the model's results with the Wikipedia database [5], to cross out some irrelevant information

To sum up, entities were used in the system by highlighting relevant words and providing a link to Wikipedia.

10 FUTURE WORK

The project was successful in creating a search system suited to the defined information needs. However, as with any product, there is still room for improvement.

The following topics represent some ideas for future work:

Multilingual Support The usage of Portuguese severely limits the potential user base. Not only that but the increased support and development of tools for English text could severely empower retrieval.

Entity Search Entity Linking was used to provide further knowledge of certain topics. The entities are already identified and indexed in Solr. It could make sense not only to present these but to also allow an entity-oriented search in the system.

Suggester Solr provides functionalities for auto-suggesting queries. This could be used to complete user prompts and improve the UX.

11 CONCLUSION

The objective of this project was to tackle the common challenges faced by students when navigating higher education course options. We aimed to provide a user-friendly solution, developing a system that extracts and analyzes data from Sigarra, encompassing all degrees and course units offered by the University of Porto.

Upon locally gathering the scraped data, we devised various search scenarios along with corresponding queries to enhance the rudimentary search system. Our improvements included the creation of multiple indexes, the incorporation of diverse filters and tokenizers, and the implementation of advanced techniques like semantic search, stemming, boost utilization in queries, and entity exploration.

Thorough testing validated the efficacy of these enhancements, underscoring their significant contributions to the system's overall efficiency.

This project served as a valuable exploration of information retrieval systems, particularly Solr, providing hands-on experience with its features, indexing mechanisms, and advanced querying capabilities.

In summary, AcademicQuest's success can be attributed to meticulous design and proficient execution, establishing itself as a valuable tool poised to assist prospective university students in their educational journey.

REFERENCES

[1] [n. d.]. The Student's guide to choosing a major | BestColleges. https://www.bestcolleges.com/resources/choosing-a-major/
[2] [n. d.]. University of Porto. https://sigarra.up.pt/up/pt/web_base.gera_pagina?p_pagina=inicial. Accessed: December 10, 2023.
[3] 2023. all-MiniLM-L6-v2. https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2. Accessed: December 10, 2023.
[4] 2023. PostgreSQL. https://www.postgresql.org/docs/current/index.html. Accessed: December 10, 2023.
[5] 2023. Wikipedia. https://www.wikipedia.org/. Accessed: December 10, 2023.
[6] Explosion AI. 2023. *spaCy - Industrial-strength Natural Language Processing in Python.* https://spacy.io/ Accessed: December 10, 2023.
[7] Apache Software Foundation. 2023. Apache Solr. https://lucene.apache.org/solr/guide/. Accessed: December 10, 2023.

[8] Apache Software Foundation. 2023. *Apache Solr MoreLikeThis Feature*. https://solr.apache.org/guide/solr/latest/query-guide/morelikethis.html Accessed: December 10, 2023.
[9] Steven Bird, Edward Loper, and Ewan Klein. 2023. NLTK: The Natural Language Toolkit. https://www.nltk.org/.
[10] Django Software Foundation. 2023. Django. https://docs.djangoproject.com/en/stable/. Accessed: December 10, 2023.
[11] Evan You. 2023. Vue.js. https://vuejs.org/. Accessed: December 10, 2023.
[12] Leonard Richardson and Cris Ewing. 2023. Beautiful Soup. https://www.crummy.com/software/BeautifulSoup/bs4/doc/. Accessed: December 10, 2023.
[13] Tailwind Labs. 2023. Tailwind CSS. https://tailwindcss.com/. Accessed: December 10, 2023.
[14] Guido Van Rossum and Fred L. Drake. 2009. *Python 3 Reference misc.* CreateSpace, Scotts Valley, CA.
[15] Vuetify. 2023. Vuetify. https://vuetifyjs.com/. Accessed: December 10, 2023.