PRI: Building an Information Retrieval System using University Data

Ilina Kirovska
up202301450@fe.up.pt

Gonçalo Almeida
up202308629@fe.up.pt

Žan Žlender
up202302230@fe.up.pt

Abstract

The primary objective of the project is to develop an information retrieval system, prioritizing comprehensive exploration of European higher education. In *Milestone 1* a collection of documents was initially obtained from Kaggle which went through different data collection and data cleaning processes using scripts written in Python. It was enriched with data from the Wikipedia and Wikidata APIs for each respective university and city, resulting in a collection of 529 University documents. Furthermore, to better understand the quality and context of the data, a more in-depth analysis with visual representations was performed. The final result was a deeper understanding of the distribution of universities through geolocation, attribute comparison as well as text analysis, which gave the best results given the nature of the data is mostly plain text. *Milestone 2* focuses on using the aforementioned documents in the open-source search engine, Solr[7]. From a set of 5 defined information needs Solr queries were created to extract those documents. Those same queries were evaluated, improved and evaluated again to see the differences in results. Concluding with a deeper understanding of the current state of the university search engine which proved to be quite well off, with an approximate 70% precision.

CCS Concepts: • **Information systems → Information retrieval**.

Keywords: data processing, datasets, data preparation, full-text search, conceptual data model, information retrieval system

1 Introduction

University rankings offer extensive assessment of higher education institutions. They are powerful tools, providing insightful analysis of the quality, performance and general credibility of the universities worldwide. Furthermore, they help students in finding educational institutions that align with their academic interests.

The Quacquarelli Symonds (QS) World University Rankings [3], debuted in 2004 and has since grown to become the foremost source of comparative data on university performance. As a result, we chose the 'QS World University Rankings 2024' Kaggle dataset [2] to be the base of our data. The dataset was then enriched using Wikipedia [5] and Wikidata [4] APIs, through which we extracted data about the universities and the cities they are located in.

The goal of the project is to build an information retrieval system by working with the extensive data made accessible through these APIs and the dataset. This document focuses on the initial stage of the project, the data preparation. In the following sections, we are going to thoroughly describe how we collected and cleaned the data, organised it into documents and explored it using various types of analysis.

This document is organized as follows. Firstly it is separated into sections of "Milestones". Milestone 1 focuses on preparing the needed documents, while Mileston 2 focuses on implementing and evaluating queries on those documents. In the section Data Collection and Preparation the processes done to get the final set of documents, including detailed a description of the encountered problems and solutions are described. The Conceptual Data Model section describes the different entities which have been defined for the domain of Universities, while the Data Characterization section provides insight into the methods used to better understand the domain and the gathered data. In the section Prospective search tasks we define the search queries which this search engine could give results for. In the following section, Document indexing, it is described in what manner a schema was created for the documents in the Solr search platform and how each individual attribute was index, using different methods. In the Retrieval section different Solr queries are defined using which we are able to extract the needed documents for the information needs. Next, in the Evaluation section different processes of evaluation are defined, both manual and automatic with which we were able to determine the efectiveness of the Solr queries. Finally, the Conclusion section summarizes all the previous sections and gives the final insights of this paper.

2 Milestone 1

2.1 Data Collection and Preparation

The scope of this project includes all European universities. As such, the first step was to filter out the original University ranking dataset to get the European ones. For that purpose, we used the Python library Pandas to load and filter the universities by their country of origin ISO code, by comparing them to the European countries' .csv file[1]. After filtering, this dataset included 529 universities located throughout Europe, including ranking information on their campuses and faculties.

At this point, we encountered a problem, which was that some university names were written in their local language

instead of English. Since the idea was to use the English Wikipedia API, which searches by the name of the university, the second step was to manually translate all the incorrectly written names into English. Using a simple Python script we identified the problematic names, then manually searched for them on Google. After finding the English version and the respective Wikipedia page, we then used those found names and wrote them into a new column in the dataset.

The following step included fetching the WikiData information for that university. With this information, we could identify any problematic entities which cannot be found, as well as get the relevant information about them. The most important is the WikiData identification number for that university. Only one university was not found because it had the symbol & in its name, which was quickly solved by exchanging the symbol with its UTF-8 representation %26.

After that, we could safely start getting the actual information we wanted, which was the plain text of the universities' Wikipedia pages. Using the Wikipedia API and the university names, as well as a Wikitext parser library, we quickly managed to get the text we wanted.

The next step was to include the cities' information where the universities are located. Since the WikiData was fetched in the third step, we can easily query all of that university's information stored in WikiData.

Since all the data is structured it includes many useful properties and relations, with the relevant ones being: located in the administrative territorial entity (P131) and location (P276).

Using those two properties we could then fetch that city's WikiData information, which would return us the city's name, which could finally be used to get that city's Wikipedia page plain text as well.

Finally, during the analysis, we determined that there were a few columns which were irrelevant or had too many incorrect or missing values, so we removed them.

After the final conversion, the dataset included 529 universities with a file size of around 33MB, as a JSON file.

The process defined in this section is expressed as a flowchart in Figure 1.

The final dataset is a JSON file containing an array of universities, where each university object consists of the following attributes: 2024_rank, 2023_rank, institution_name, country_code, country, size, focus, age, status, academic_reputation_score, academic_reputation_rank, employer_reputation_score, employer_reputation_rank, faculty_student_score, faculty_student_rank, citations_per_faculty_score, citations_per_faculty_rank, international_students_score, international_students_rank, international_research_network_score, foundation_year international_research_network_rank, employment_outcomes_score, employment_outcomes_rank, overall_score, institution_name__wrong, wikidata, wikipedia_text, city_name, city_wikipedia_text, coordinates.

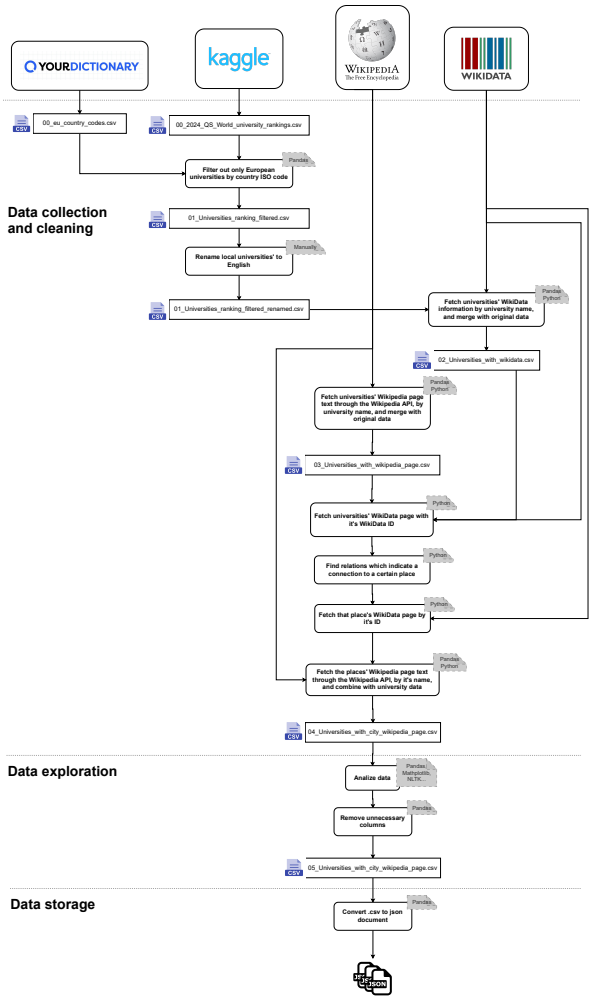

Figure 1. Data collection and preparation pipeline

2.2 Conceptual Data Model

The conceptual data model, as seen in Figure 2, represents the relationship between each entity in the domain. In this case, the model is straightforward since most of the information is centred around the university. The university information includes its rank, as well as past year's ranking. Additionally, there are many metrics which determine the rank, like Employer Reputation Score, Employer Reputation Rank, Faculty Student Score, Faculty Student Rank etc. One of the most important metrics is the **Overall SCORE**. Although we had extracted different information about the university, for example, the Wikipedia text, it's still connected to a specific university. Finally, each university is located in a city, which has its name as well as the City Wikipedia text, extracted from its Wikipedia web page.

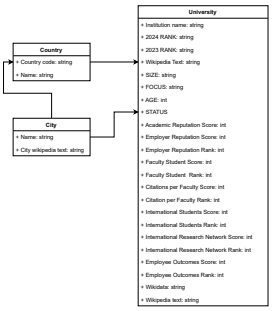

Figure 2. Conceptual data model

2.3 Data Characterization

Understanding the collected data and identifying its key features is of great importance when building an information retrieval system. This was accomplished by doing additional data analysis, through which we gained more insights about the structure and context of the data. The results, along with proper visualisation, are explained in the following subsections.

2.3.1 Initial analysis of the acquired dataset. In the beginning stages of the process, a basic statistical analysis of the Kaggle dataset [2] was done. The aim was to identify how many of the universities fit our criteria of being a European university. As seen in Figure 3, 35.34% of all universities were in Europe and hence, kept in the dataset.

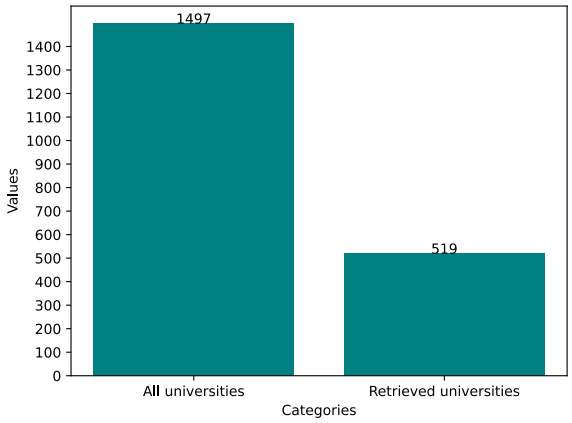

Figure 3. Number of all universities versus the number of the universities that were kept

2.3.2 Location analysis. As our focus is on European universities, their distribution over Europe is vital information. Therefore, it is illustrated using a variety of visual representations.
Figure 4 shows that the countries with the most universities are the United Kingdom (90) and Germany (49).

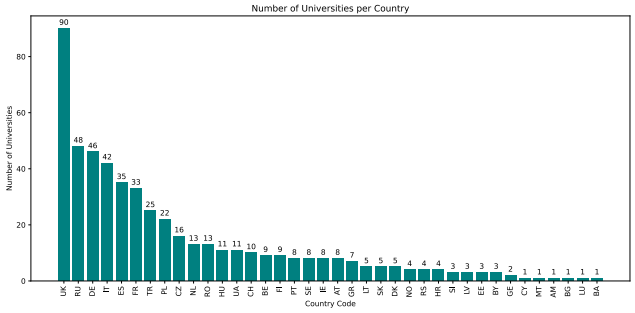

Figure 4. Number of universities grouped by country

Moving on to geographic visualization, Figure 5 is a map that displays the cities where universities are located. Each city is pinpointed on the map, providing a clear spatial perspective of the university distribution.

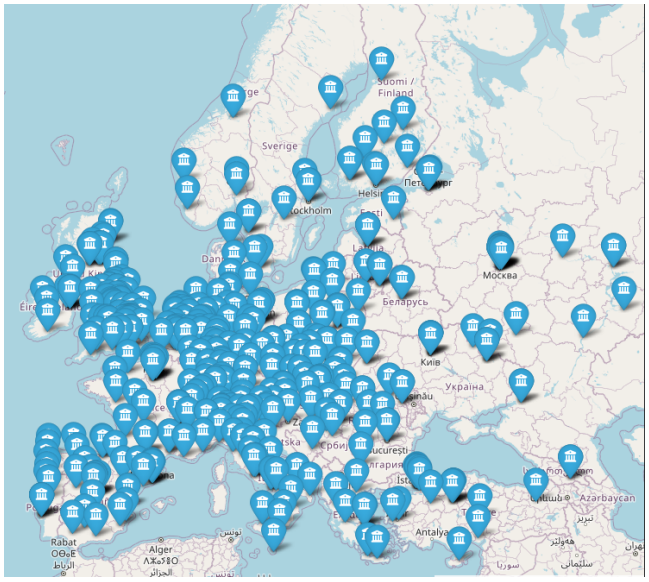

Figure 5. Map of the universities analysed

Figure 6 is a heat map that visualizes the distribution of universities across Europe, and it helps to identify clusters of universities in specific regions.

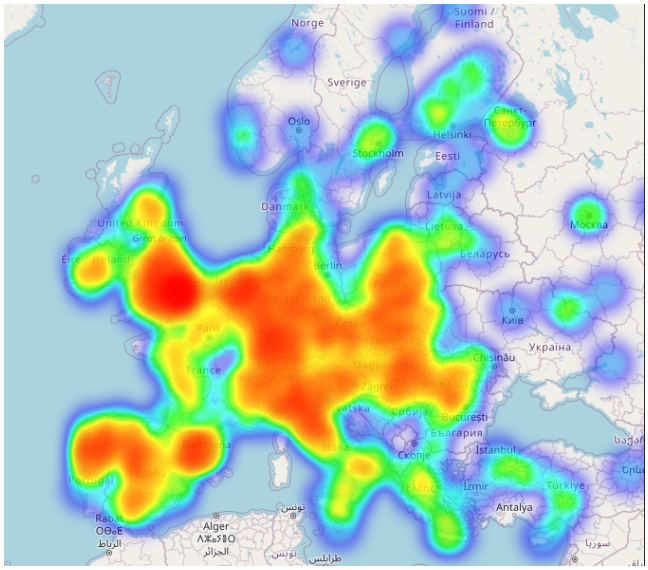

Figure 6. Heat map of the universities analysed

2.3.3 Ranking analysis. Figure 7 shows the changes in university rankings over time, by comparing the university ranks in 2024 (2023/2024) and 2023 (2022/2023). This dynamic visualization highlights universities that have experienced rank changes, such as improvements or declines, between the two years.

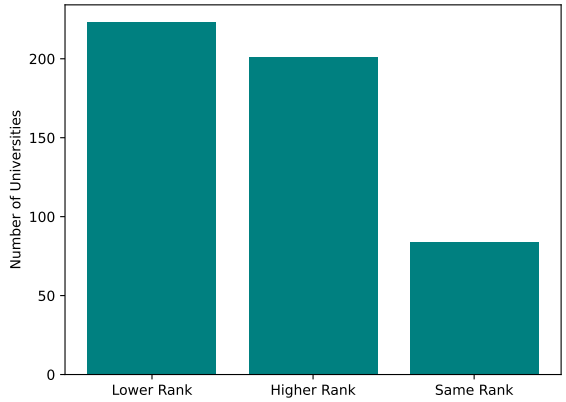

Figure 7. Distribution of universities grouped by rank

2.3.4 Size and Age Analysis. Figure 8 illustrates the number of universities falling into different size categories (XL, L, M, S) while considering their age, which ranges from new (Category 1) to older (Category 5). The teal-coloured bars represent the count of universities in each size category, with age categories distinguished by different shades within each bar. This visualization helps us understand how university sizes vary across different age categories.

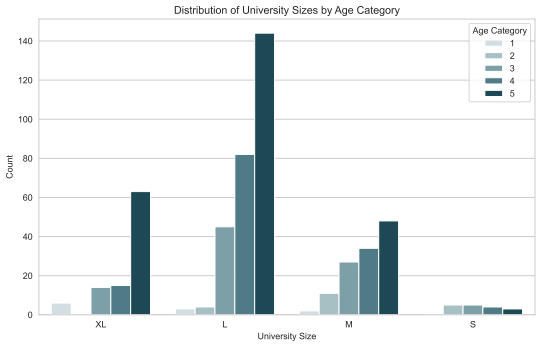

Figure 8. Distribution of university sizes by age category

2.3.5 Text analysis. The majority of the data in our system is stored as plain text, hence one of the most efficient ways to analyse it is to identify the most common words. Through them, we can easily find out the tone and content of the text.

We used the NLTK library [6] which offers NLP techniques. After the text was preprocessed using NLTK, we calculated the word frequencies and constructed the word clouds. As shown in the first photo in Figure 9 some of the most common unigrams in the universities' text data were university (appears 31.918 times), student (9.584 times), faculty (8.344 times) and science (7.946 times). The other 3 photos show the most common bigrams, trigrams and fourgrams. When we calculated these wordclouds we specifically removed the n-grams which consisted of only stopwords, this mostly affected the bigrams.

Figure 9. Wordclouds for the universities using unigrams, bigrams, trigrams and fourgrams respectively (starting from the top left corner).

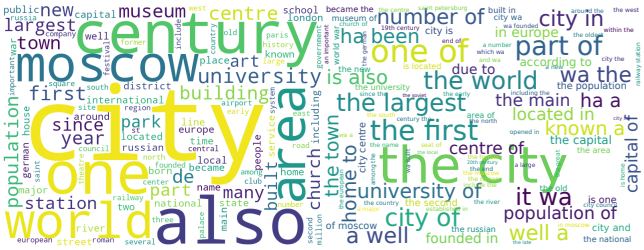

Figure 10. Wordclouds for the cities using unigrams and bigrams.

The left photo from Figure 10 shows that in the cities' data, the most common unigrams were city (41.723 times), also (10.365 times), one (8.811 times) and century (8.583 times), and on the right we can see the most common bigrams.

2.4 Prospective search tasks

The focus of this search engine is to provide information about the available universities. Taking that into consideration, these are some of the possible search tasks:

1. Looking for universities that are top-ranked in computer science and are located in cities that have rich cultural heritage
2. Looking for universities located in the United Kingdom that have courses in biology and are ranked in the top 150
3. Looking for universities in Germany that have a dental medicine faculty/dentistry and a large number of students
4. Looking for universities that have a faculty of engineering or a faculty of science and are located in a city with a Mediterranean climate
5. Looking for top-ranked universities in the north of Europe with a focus on the Computer Science field

3 Milestone 2

In the following sections, we will focus on the Information Retrieval part of the project. To be able to carry out this task we chose to use Solr - a powerful open-source search platform based on Apache Lucene. Solr allows document indexing and retrieval as well as efficient query processing and document ranking. Using Solr queries, for each of the information needs defined in Section 2.4 we retrieved the first 30 results and consequently evaluated our search engine.

3.1 Document indexing

As mentioned in section 2.1, the result of Milestone 1 was a collection of 529 documents stored in one JSON file. To be able to retrieve the wanted information from these documents using Solr, we first need to index them using a schema. Schemas define various characteristics of the document fields, such as their type and how they should be processed during indexing and querying.

The indexing process was completely automatised by using a shell script. To upload the schemas and index the documents we used the Solr API and the commands shown on Listing 1.

Listing 1. Solr commands for uploading the schema and documents

```
curl -X POST -H 'Content-type:application/json' \
--data-binary "@new_schema.json" \
http://localhost:8983/solr/universities/schema

curl -X POST -H 'Content-type:application/json' \
--data-binary "@datasets/06_University_documents.json" \
http://localhost:8983/solr/universities/update?commit=true
```

3.1.1 Schema design.

In order to optimize and improve the query results for a more efficient search experience, we designed two schemas — one with a basic structure (Version 1 - V1) and another enhanced version incorporating various filters (Version 2 - V2). Both schemas have the same fields, as shown in Table 1, but with different field types. We didn't index the fields 'wikidata' and 'institution_name_wrong' as they do not provide essential information that is helpful during the search process.

When we designed the two schemas the goal was to have one schema with little to no optimization and another one utilizing multiple Solr features. As seen in Table 1 for the first schema we used the same field type for all fields, except for the coordinates, where we used the LatLonPointSpatialField class. Moreover, no filters were used during indexing or querying apart from two very basic ones - ASCII Folding Filter and Lower Case Filter.

The second schema uses a lot more of the features Solr offers. In terms of field types, we used basic types such as int and float for fields containing standard data, whereas, for the fields 'wikipedia_text' and 'city_wikipedia_text', which contain the most complex and important data, we defined a custom type called wikipediaText. The tokenizer used for it is the Classic Tokenizer. For both, indexing and querying, we applied several different filters in the following order:

- ASCII Folding Filter - converts alphabetic, numeric, and symbolic Unicode characters which are not in the Basic Latin Unicode block (the first 127 ASCII characters) to their ASCII equivalents, if one exists
- Lower Case Filter - converts all characters to lowercase
- Classic Filter - strips periods from acronyms and "'s" from possessives
- English Minimal Stem Filter - stems plural English words to their singular form
- Porter Stem Filter - applies the Porter Stemming Algorithm

By using these filters we make sure to have some generality and reduce the ambiguity while querying. After this pipeline and underlying all of the filter transformations tokens have a more general form which is a crucial part of query matching.

Field	Indexed	Field type	
		Version 1	Version 2
2024_rank	true	text	int
2023_rank	true	text	float
institution_name_-_wrong	false	text	text
institution_name	true	text	text
country_code	true	text	text
country	true	text	text
size	true	text	text
focus	true	text	text
age	true	text	text
status	true	text	text
academic_reputation_score	true	text	float
academic_reputation_rank	true	text	text
employer_reputation_score	true	text	float
employer_reputation_rank	true	text	text
faculty_student_score	true	text	float
faculty_student_rank	true	text	text
citations_per_faculty_score	true	text	float
citations_per_faculty_rank	true	text	text
international_students_score	true	text	float
international_students_rank	true	text	text
international_research _network_score	true	text	float
international_research _network_rank	true	text	text
employment_outcomes_score	true	text	float
employment_outcomes_rank	true	text	text
overall_score	true	text	float
foundation_date	true	text	tdate
wikidata	false	text	text
wikipedia_text	true	text	wikipediaText
city_wikipedia_text	true	text	wikipediaText
coordinates	true	coordinates	coordinates

Table 1. Schema fields and their types.

3.2 Retrieval

To be able to evaluate our search engine, we need to first query our information needs and retrieve the results. Solr offers numerous retrieval features that allow the user to tailor the search queries in order to get the best possible results. While querying we used Solr's Standard Query Parser. As can be seen in Section 3.2.1 we experimented with various boosts that are available and in our case, the most useful boosts were the Term Boost and Wildcards.

3.2.1 Queries. To meet the Information Needs, presented in section 3.2, we initially used a combination of our basic schema (Version 1) with the following queries (default operator OR):

1. wikipedia_text:"computer science" OR
   city_wikipedia_text:heritage OR
   2024_rank:[1 TO 100]
2. country: "United Kingdom"
   country_code: UK
   wikipedia_text: biology
   2024_rank:[1 TO 150]
3. country: Germany
   country_code: DE
   size: large
   wikipedia_text: "dental medicine"
4. wikipedia_text: "faculty of science"
   wikipedia_text: "faculty of engineering"
   city_wikipedia_text: "Mediterranean climate"
5. wikipedia_test: "Computer Science"
   2024_rank:[1 TO 150]
   city_wikipedia_text: "north europe"

Then, we used the following boosted queries on our improved schema (Version 2):

1. wikipedia_text:"computer science"^3 AND
   city_wikipedia_text:heritage^2 AND
   2024_rank:[* TO 200]^4
2. country: "United Kingdom"^2
   country_code: UK
   wikipedia_text:biology^2
   2024_rank:[1 TO 150]^3
3. country: Germany^2
   country_code: DE
   size: large
   wikipedia_text: dent*^2
4. wikipedia_text: "faculty of science"^2
   wikipedia_text: "faculty of engineering"^2
   city_wikipedia_text: "Mediterranean climate"
5. wikipedia_test: "Comput* Science"^4
   2024_rank:[1 TO 150]
   fq: !geofilt sfield=coordinates
   pt=61.069625,4.867638 d=1430868.94
   (sort: 2024_rank asc was tested, but it wasn't a good approach)

3.3 Evaluation

3.3.1 Manual evaluation process. Each of our information needs was queried using two queries on two different schemas, one query per schema. For each query, we recovered the first 30 results, which were then deemed relevant or not relevant, based on the fact if they satisfied the query requirements. Because each of the information needs we have, has multiple requirements that must be met, we defined the

following user priorities in case a result does not meet all of them, so we can sort it into one of the two groups (the number of the information need corresponds to its number in Section 2.4):

1. *Information need*: the user prioritises high ranking universities in the field of computer science where the city the university is located in also has a rich heritage. Because of the aforementioned reasons the 1st query searches the 'wikipedia_text' attribute for the word "computer science", where the '2024_rank' is greater than 100. Given that we deemed the word "heritage" from the phrase "cities that have rich cultural heritage", from the information need, important we also query the 'city_wikipedia_text' for the word 'heritage'.
2. *Information need*: the user prioritises universities that have a rank of 150 or higher, giving those in the UK with courses in biology a lower priority
3. *Information need*: the user prioritises universities located in Germany offering courses in dental medicine, giving those having a large number of students a lower priority
4. *Information need*: the user prioritises universities offering courses in science and engineering, giving those in a city with a Mediterranean climate a lower priority than those with the wanted courses.
5. *Information need*: the user prioritises the top universities located in the north of Europe that have interest on the Computer Science field.

For each of the given queries, the process of evaluating if the result was relevant was a bit different but relative to the topic. For example, in the first query one of the search phrases was *"cities that have rich cultural heritage"*, since that phrase is subjective we had to examine each of the resulting documents, in this case the 'city_wikipedia_text' attribute, if it contained mentions that might suggest that city has a long standing history, cultural locations, or even just the word heritage. A similar reasoning was used while examining the documents for mentions of *"computer science"* and the '2024_rank' attribute.

These priorities can also be identified by looking at the term boosts each query uses.

3.3.2 Precision metrics. After manually determining the relevance of each document, for each query, we calculated the following performance measures:

- Precision at n (P@n): Measures the proportion of retrieved documents in the top n results that are relevant to the information need.
- Average Precision (AvP): Calculates the average precision across all relevant documents, providing a single-value summary of the precision-recall curve.
- Recall at n (R@n): Evaluates the ability of the system to retrieve relevant documents by measuring the proportion of relevant documents among the top n results.
- Mean Average Precision (MAP): Calculates the average precision across multiple queries, providing an overall measure of the system's performance. It considers both precision and recall, making it a comprehensive metric for our information retrieval system.
- F@n (F-measure at n): Combines precision and recall into a single metric, balancing both aspects of the system's performance, being useful when there is a need to consider both false positives and false negatives in the top n results.

Tables 2 and 3 show the P@n, AvP, R@n, and F@n for each of the five queries, non-boosted and boosted, respectively. We only display P@10, R@10, and F@10 because the first 10 search results are the ones that we deem to be the most relevant. For the calculation of AvP we used all 30 results.

Table 2. Information retrieval effectiveness measures for non-boosted queries.

Query	P@10	AvP	R@10	F@10
Query 1	0.4	0.42	0.29	0.42
Query 2	0.4	0.65	0.57	0.57
Query 3	0.7	0.8	0.58	0.64
Query 4	0.6	0.79	0.27	0.38
Query 5	0.3	0.37	0.5	0.38

Table 3. Information retrieval effectiveness measures for boosted queries.

Query	P@10	AvP	R@10	F@10
Query 1	1.0	1.0	0.34	0.67
Query 2	0.9	0.9	0.64	0.78
Query 3	1.0	1.0	0.56	0.71
Query 4	1.0	0.89	0.43	0.61
Query 5	1.0	1.0	0.33	0.5

Table 4. Mean Average Precision (MAP) for both search engines.

Metric	Version 1	Version 2
MAP	0.62	0.96

Taking the precision and recall query metrics results we can plot Precision Recall curves (P-R curves) for a more visual perspective on the trends. In each line chart the blue line, marked as "V1", represents the initial query while the red line, or "V2", represents the boosted query. The X-axis shows the recall values and the Y-axis shows the precision. Meaning the graph itself can be interpreted as a trend showing the number of relevant documents as more documents are retrieved.

As can be seen in Figure 11 all P-R curves follow a similar trend. In the first graph, for the first information need, we can clearly see the difference between the initial and boosted query. Although their trends are similar, where both maintain an almost straight level of precision across all retrieved documents, the boosted version has a perfect precision value while the initial version starts with a 50% precision which drops off at the end.

The queries for the second, third and forth information need start off similarly, with all queries having a perfect precision. However, as the recall goes up both the initial queries' precision falls off. In contrast, the boosted query of the third information need keeps it's precision, while the second and forth drops of slightly.

Finally, in the fifth information need's queries we can see the greatest differences. The initial query starts off with a lower precision than the boosted one, and starts decreasing in precision drastically finishing with less than 25% precision at the end. On the other hand, the boosted query maintains it's 100% precision.

In all cases at the end of the curve the initial version ended up having a lower precision than the boosted version of the query, from which we can conclude that boosting the queries was beneficial to the results of the search engine.

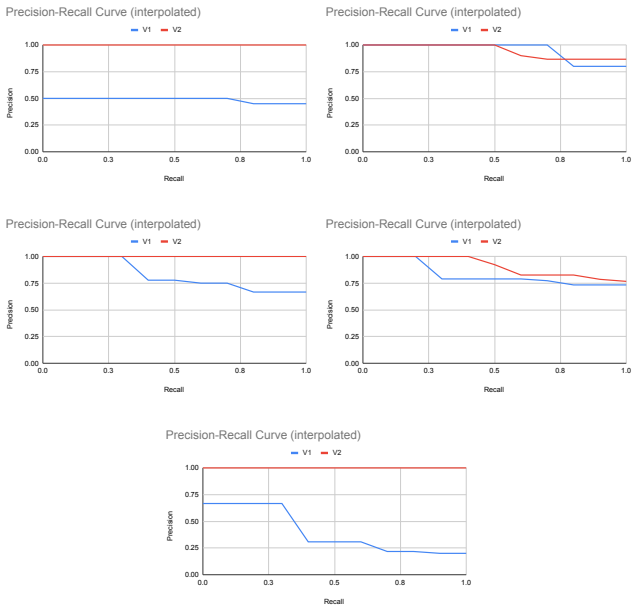

Figure 11. Interpolated Precision-Recall curves for Queries 1-5 (starting from the top left corner).

3.3.3 Discussion. Our evaluation results are documented in Table 2, Table 3 and Table 4. In Table 2 we can see the evaluation results obtained when using Version 1 of the schema (basic version) and the queries without any boosts or filters. Furthermore, Table 3 shows the evaluation results obtained when using Version 2 of the schema (optimized version) and the boosted queries.

For information retrieval systems the relevance of the top results is of primary importance. This is reflected by the P@10 metric. If for each query we compare the P@10 values in Table 2 and Table 3 we can see great improvement. By looking at the values in Table 3, which are almost all 1.0, we can conclude that by using the right combination of Solr Tokenizers, Filters and query boosts we managed to develop a system whose top 10 results are all relevant. This conclusion is also supported by the AvP values for Version 2.

3.4 Conclusion

This report details the first two milestones of the project, whose goal is to build an information retrieval system.

In the first milestone, we focused primarily on dataset selection, preparation and exploration. The dataset that was used includes all of the basic information about European universities, including rankings, locations, and institutional characteristics. Wikidata and Wikipedia API were used to streamline the data-collecting process and ensure data relevance and accuracy for the study. At the end of the milestone, we had a document collection with all the necessary university information.

In the second milestone, we successfully implemented Solr, indexed the document collection, and retrieved and evaluated the query results based on our information needs. Our efforts were concentrated on refining document indexing, improving the basic schema, crafting effective retrieval queries, and evaluating the results manually to assess Solr's performance. Also, for each query, we made a boosted version where we got significantly better results, as expected. The next steps for the project will be further improvement of the system as well as creating a user interface for it.

References

[1] [n. d.]. List of European countries. Retrieved October 10, 2023 from https://www.yourdictionary.com/articles/europe-country-codes

[2] [n. d.]. QS World University Rankings 2024 Dataset. Retrieved September 28, 2023 from https://www.kaggle.com/datasets/joebeachcapital/qs-world-university-rankings-2024/data

[3] [n. d.]. Quacquarelli Symonds (QS) World University Rankings. Retrieved October 10, 2023 from https://www.qs.com/

[4] [n. d.]. Wikidata API. Retrieved October 10, 2023 from https://www.wikidata.org/wiki/Wikidata:REST_API

[5] [n. d.]. Wikipedia API. Retrieved October 10, 2023 from https://www.mediawiki.org/wiki/API:Main_page#Endpoint

[6] Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.*

[7] lucene.apache.org. 2012. Solr. http://lucene.apache.org/solr/