Università della Svizzera italiana

Information Retrieval **2024**

Student: Costanza Rodiguez Gavazzi, Agnese Zamboni, Davide Frova

Due date: Wednesday, 27 November 2024, 11:59 PM

1. Overview

The Information Retrieval project is advancing according to plan. The responsibilities have been allocated among three team members, each concentrating on a separate part of the project.

2. Frontend Design and Implementation

Davide is responsible for the frontend design and implementation. The initial design research and UI mockup creation have been completed using Figma. The next steps involve implementing the design using Next.js. The design aims to be minimalistic and user-friendly. The current prototype contains simple search bar with the iconic search button, the length of the bar is not too short to incentivize the user to type longer queries. A logo representing the project will be placed above the search bar, this will aim to give the user a sense of the project's identity. Under the search bar we have the three pill shaped filter dropdowns, we identified as possible filters: the charity 'theme', the 'location' and the last filter is still to be decided. After entering the query the user will be presented with a list of cards cotaining various informations about the charities, such as the name, the logo and a "google snippet" like description with highlighted keywords. There are also present the 'thumb up' and 'thumb down' buttons to allow the user to give feedback on the results, this will re-run the query with the feedback in mind.
Here are some screenshots of the current design:

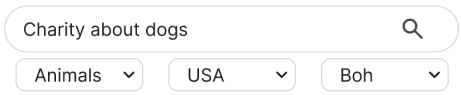

Figure 1: Mockup Search Page

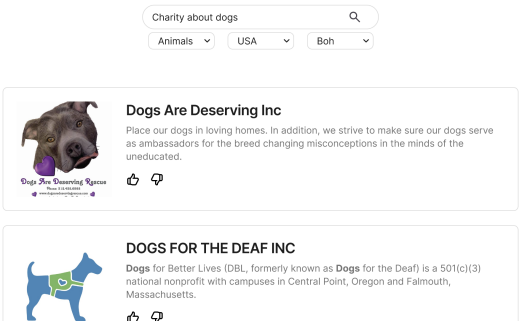

Figure 2: Mockup Results Browsing

3. Data Collection and Processing

We built data processing systems for two major charity platforms - Global Giving and Charity Navigator. Each required a different approach due to how their data is made available.

For Global Giving, we used their API to download the available data in XML format. Rather than scraping their website, we wrote a parser to extract information from this structured data file. The system processes several types of information about each organization:

- Basic details like their name and location
- Operational metrics (active and total projects)
- Mission statements and website URLs
- Causes they work on ("themes")
- Countries where they operate

After getting this data, we enhanced it with additional geographical information using pycountry and pycountry_convert libraries. This let us add continental classifications based on each organization's headquarters location. We also improved the data structure - renaming fields to be consistent with the data fro and reorganizing the information about operational territories to make it more useful for analysis.

The Charity Navigator system needed a different approach since they have a modern API. We used their GraphQL API instead of a regular REST API because it's more efficient - we can specify exactly what data we want, avoid making multiple calls, and handle nested data better. The system can handle large amounts of data, processing up to 10,000 records in small batches to avoid overwhelming their server. One of our biggest technical challenges was automatically finding logos on charity websites. We built a system that uses multiple approaches to find the right logo:

First, it looks through the webpage's structure for anything marked as a logo Then it checks special webpage tags that often contain logos It also looks for icon links and any images that might be logos Finally, it carefully checks each potential logo to make sure it's not picking up the wrong things (like menu icons)

We used several standard Python libraries to build these systems. BeautifulSoup handles the messy HTML we sometimes get from websites. The requests library manages all our web connections. We used pycountry for standardizing country names and codes. All the data gets saved in JSON format. We chose JSON because:

- It works with pretty much any system
- It's easy to read when debugging
- It handles nested data well
- Other tools can easily work with it

We made sure the systems could handle problems gracefully. Instead of crashing when something goes wrong, they skip the problem and keep going. This means we get as much data as possible while keeping it accurate. This whole setup gives us a solid foundation for analyzing charitable organizations. The systems are built to:

- Keep the data high quality and complete
- Work reliably even when things go wrong
- Handle large amounts of data efficiently
- Output data in a standard format
- Allow for future improvements

These choices mean we can analyze organizations in detail - looking at where they work, what they focus on, and how they operate, while keeping the systems maintainable and reliable.

4. Backend and Indexing/Retrieval

Agnese is focusing on the backend and indexing/retrieval part. We have chosen PyTerrier for indexing and searching the data, and FastAPI to create a simple Python backend that will interface with PyTerrier. The necessary endpoints for the frontend-backend communication are being defined. An draft of the main endpoint comprehends parameters for the user query, the filters and the feedback on a result.

```
GET /search?q=userQuery&filters=filter1,filter2,filter3&feedback={docId:123,score:1}
```

5. User Evaluation

6. Appendix