Università della Svizzera italiana

Institute of Computing CI

Information Retrieval **2024**

Students: Matteo Borioli & Stipe Peran

Final Report: Pokémon Information Retrieval System

This report documents the design and implementation of a Pokémon-themed Information Retrieval (IR) system. The project spans the entire pipeline from web crawling to gather data, indexing the data for efficient retrieval, and creating a responsive front-end interface. By combining basic and advanced IR features, the system provides a comprehensive and interactive user experience. Notable features include advanced filtering, tabular result presentation, and automatic recommendations based on weighted attributes. This report details the tools, methods, and design choices, along with a discussion of implemented and unimplemented features.

Contents

1. Introduction

The objective of this mini-project was to design and develop an Information Retrieval (IR) system centered around Pokémon. The project involved building an end-to-end pipeline comprising web crawling to gather data, indexing the data for efficient retrieval, and creating a user-friendly frontend for searching and filtering Pokémon-related information. By implementing a mix of basic and advanced IR features, we aimed to provide users with a comprehensive and highly interactive search experience. This report documents the tools and methods used, as well as the features implemented in the system.

2. Tools

1. Web Crawling

To collect data for this project, we employed the Python library `scrapy`, a robust and efficient framework for web scraping. The data was sourced from several primary websites, with different spiders tailored for specific data extraction tasks. The main spiders used are:

- `pokemonBulbapedia_listAllPokemon`: Extracted Pokémon images from `https://bulbapedia.bulbagarden.net/`.
- `pokemonDatabase_listAllPokemon`: Gathered detailed Pokémon statistics, names, and descriptions from `https://pokemondb.net/`.
- `pokemon_fandom`: Extracted evolution line information from `https://pokemon.fandom.com/`, ensuring that this data field was populated for every Pokémon entry.
- `pokemon_megaforms`: Identified and filtered Mega Evolutions images for relevant Pokémon entries.
- `pokemon_otherforms`: Retrieved images and information about alternate Pokémon forms, such as Alolan and Galarian forms.

These spiders were designed to efficiently navigate the website structures and extract the required data. By distributing the data collection tasks among multiple spiders, we ensured that the scraping process was both modular and efficient.

The scraped data was initially stored in separate JSON files, each corresponding to a specific spider. These files were subsequently merged using auxiliary functions, sorted by Pokémon ID for consistency, and saved as `bigPokemonData.json` in the `data/final_data` folder.

2. Indexing

The back-end of our system was built using Apache Solr, an open-source enterprise search platform. Solr was chosen for its powerful indexing and query capabilities, which allowed us to efficiently handle the Pokémon dataset and implement advanced search functionalities.

To integrate the scraped data with Solr, we modified the `managed-schema.xml` file, configuring it to define the structure of the indexed data. This schema setup ensured that key attributes such as Pokémon ID, name, type, and evolution line were indexed correctly. To further enhance the relevance of search results, we applied custom weights to certain attributes during query formulation. This allowed us to prioritize specific fields, improving the overall user experience.

3. Front-end

The front-end of our IR system was developed using React, a modern JavaScript library for building dynamic and responsive user interfaces. We utilized Tailwind CSS for styling, which provided a utility-first approach to quickly design a clean and aesthetically pleasing layout.

The front-end features a well-structured interface, including a search bar for quick specific queries, a filtering sidebar refining search results, a tabular layout for displaying multiple results simultaneously, and a button for browsing all Pokémon. This design was aimed at maximizing usability and providing a seamless experience for users.

3. Implemented Features

1. Simple Features

1.1. Results Presentation

One of the fundamental features of our system is the tabular presentation of search results. This approach allows users to view multiple results at a glance, with each table cell containing some information about a Pokémon, such as its name, alternative form (if existing), type(s), number, and its image. This format ensures that relevant details are easily accessible.

1.2. Filtering

Filtering is another key feature, enabling users to narrow down search results based on specific criteria. Users can utilize the filtering sidebar to apply attribute-based filters, such as Pokémon type, base statistics (Total, hp, defense, attack, etc.). In addition, the search bar supports attribute-specific queries. While the system prioritizes exact matches, it also includes related Pokémon based on attributes such as description, type or evolution line, ensuring a broader range of results.

1.3. Result Snippets

To provide a quick overview of each result, the system displays a snippet of the Pokémon's details, which is a mix of both organic results and enriched results for better visibility. This is because we do not display the description, but we do for the image. Due to the limited number of available attributes, the snippet feature is partially implemented but still offers valuable context for each result.

2. Complex Features

2.1. Automatic Recommendation

We implemented an automatic recommendation feature leveraging Solr's advanced query capabilities. This feature suggests similar Pokémon based on weighted attributes such as name, types, and description. For example, when a user opens the specific Pokémon page results, the system retrieves additional Pokémon that share similar characteristics, presenting them as recommendations in the "More like this" section. This functionality enhances the user experience by providing relevant suggestions.

3. Unimplemented Features

While our system supports features like query relevance and recommendations, certain advanced functionalities such as user feedback for result ranking and clustering of results into topics were not implemented due to time constraints. These features represent potential areas for future development.

4. User Evaluation

1. Overview

To conduct the user evaluation, we designed a series of tasks for the participants. The evaluation focused on measuring the performance of the participants, specifically their speed, by recording the time taken to complete each task. The tasks are detailed below:

2. Tasks

Task 1: Basic Search Functionality

Objective: Test the search bar's ability to retrieve Pokémon based on user queries.
Instructions:

1. Use the **search bar** to search for a Pokémon with the attribute *"Bulbasaur"*.
2. Record the Pokémon that appear in the search results.
3. If no Pokémon appear, describe what you see on the screen.

Expected Outcomes:

- Verify if relevant Pokémon are retrieved.
- Ensure the *"no results"* image appears if no Pokémon match the query.

Task 2: Filter Usage and Reset

Objective: Test the filter sliders, dropdown menus, and reset button.
Instructions:

1. Open the **filter panel**.
2. Set the sliders to filter Pokémon with the following criteria:
   - Attack: Between 50 and 100.
   - Defense: Between 40 and 80.
   - Speed: Between 60 and 200.
3. Select *"Fire"* as the primary Pokémon type in one dropdown menu.
4. Select *"Poison"* as the secondary Pokémon type in one dropdown menu.
5. Apply the filters by clicking the **Submit** button.
6. Check the retrieved Pokémon and take note of the results. (Should be only *Salazzle* and *Iron Moth*).
7. Use the **Reset** button to restore the filters to default.
8. Verify that the filters have been reset and perform a new search to confirm.

Expected Outcomes:

- Filters should update the search results based on the provided criteria.
- Reset functionality should clear the filters and restore the default state.

Task 3: Pokémon Details Page and Recommender

Objective: Test navigation to the detailed Pokémon page and validate the *"More Like This"* recommender.
Instructions:

1. From the list of retrieved Pokémon, click on a **specific Pokémon** to navigate to its detailed page.
2. Review the information displayed on the page and note the Pokémon's details.
3. Scroll down to the *"More Like This"* section.
4. Observe the Pokémon recommended as *"similar"* and describe whether they align with the Pokémon's characteristics (e.g., same type or attributes).

Expected Outcomes:

- Ensure the detailed page displays all relevant information about the selected Pokémon.
- Verify that the *"More Like This"* section shows Pokémon that are meaningfully similar.

Task 4: Edge Case – Empty Results

Objective: Test the system's behavior when no results match the query.
Instructions:

1. Use the **search bar** to enter a query that intentionally *should return no Pokémon.* For example, search for a Pokémon with *"Bulbachu"* (assuming no such Pokémon exists).
2. Verify what is displayed when the search yields no results.

Expected Outcomes:

- Confirm that the *"No Pokémon Found"* image or message is shown.
- Ensure the interface remains functional for new searches or filter adjustments.

3. Participant Recorded Times

The user evaluation has been performed on different categories of people in terms of experience (computer science students, and non computer science students). In this section we report the measured speed times performed by the participants:

Participant id	Task1	Task 2	Task 3	Task 4
1	7	53	45	13
2	10	51	39	17
3	13	49	48	11
4	8	39	32	9

Table 1: Recorded times for each participant in seconds.

4. Feedback and Suggestions

Participants provided the following feedback and suggestions to improve the search engine:

1. Provide a way to clear the numerical input filters, as users cannot currently delete all digits. The input field always retains at least one digit, requiring users to use arrow keys to adjust the leftmost digit. Despite this visual problem, the system works.

2. Implement a clear button to empty the search bar.

3. Show the loading page first, and then display the results page.

4. Add a button to browse through all Pokémon in the database without needing a specific query.

We listened to the participants and implemented part of the suggestions; specifically (2), (3) and (4), now we have an updated and upgraded version of the search engine.

5. Conclusion

In conclusion, our Pokémon IR system successfully combines web crawling, data indexing, and front-end development to deliver a robust and user-friendly platform for searching and exploring Pokémon data. By implementing features such as tabular result presentation, advanced filtering, and automatic recommendations, we have created a system that effectively addresses the needs of users.

This project highlights the power of integrating open-source tools such as Scrapy, Solr, React, and Tailwind CSS in building specialized IR systems. Moving forward, future enhancements could include implementing user feedback mechanisms, result clustering, and further improving the snippet generation feature to make the system even more comprehensive and versatile.

6. Code Repository

The complete source code for the Pokémon Information Retrieval System, including the web scraping scripts, Solr configurations, and the React front-end implementation, is available on GitHub at the following link:

`https://github.com/Luculele/IR_project`