Information Retrieval Project

Likollari Kelvin, Prendi Gerald

December 13, 2022

An advanced Web Crawler, used to crawl data from 3 different motorcycle web pages.

1. Introduction

For the Information Retrieval project, an innovative search engine has been created. This search engine is designed to display motorbike information. Three distinct, but motorcycle-related websites have been crawled in order to accommodate users' expectations for motorbike search when utilizing the search engine.

2. How to start the crawler

In case you would like to experience beforehand on your own what our Web Crawler looks like, then the following steps need to be followed. For the back end of our application, the commands stated below need to be followed:

```
python3 -m venv .venv

source .venv/bin/activate

pip install -r requirements.txt

solr create -c motorcycles__

cd src

scrapy crawl Motorcycle -O motorcycles.json

cd solr/api

python3 schema.py

post -c motorcycles__ motorcycles.json

cd ..

FLASK_APP=src flask run
```

Unlike the back end, our front end is significantly easier to be run. The needed-to-be-run command is:

```
npm run dev
```

Given the intricacy of all these instructions (particularly for the backend), a bash script file containing the commands required to start the application has been prepared to make your life simpler.

3. Frameworks and Project Structure

The project has been separated into 2 parts:

(1) The Backend

(2) The Frontend

Effectively, as the name states, the back-end of our application contains the structure of our project and the logic describing the way our application parts are connected. The official language used for the back-end is Python. For the User Interface, VueJS was used. VueJS is an open-source model–view–ViewModel front-end JavaScript framework, used mainly for building user interfaces and single-page applications. Effectively, VueJS is a JavaScript framework, aimed at personalizing user interfaces.

4. Application Backend

The code written to crawl the websites is located in the Motorbike, Motorbike2, and Motorbike3 python files.The data indexing section follows once we have adjusted our code to each of the webpage's needs.

5. Application Frontend

The frontend should be simpler to comprehend. As previously stated, VueJS was utilized to create the User Interface. Our front end has been broken into several components, each focused at a different part of the program. The search bar, the page's footer, the auto-completion component shown while the user is entering a query, the list of results after the user has finished formulating his question and wishes to receive results, and so on are common components. All of those components have been suitably styled and are included in the main.vue file, App.vue, which contains (includes) all of the User Interface components. App.vue is the main user-interface file that contains all of the components and views.

In addition, a directory containing the application views has been created. These views, unlike the components directory, are dynamic and are changed based on the current application state (i.e. if the user is formulating a query). Vite, last but not least, was utilized. Vite comes with several templates and supports various prominent front-end frameworks, including VueJS. Vite is a build tool that includes a development server that bundles production code. To begin the User Interface, use the aforementioned command:

```
npm run dev
```

, we are essentially running Vite, on port 5173. Vite provides many functionalities such as running an application with a developer server and has additional features such as a live server which reloads the page when a code modification has been performed. Moreover provides detailed descriptions in case of compiler errors.

Additionally, in the front end, you will find a **utils** directory containing various Javascript files. Those files contain some typical getters and setters for our application, written to make our lives easier.

Moreover, a router directory has been set up, containing a file describing the routes of our application and the components inserted, based on the URL path.

Furthermore, in the store directory, there exists a javascript file containing, in the beginning, the accessors and mutators necessary for our application. The interesting part though is at the end, where actions regarding our application are contained. They are divided into two categories, querying, and suggestions. After fetching the respective path, the results are stored in a JSON file.

Last but not least, a Dockerfile has been written in order to make the process of running the application on different computers easier.

6. User Evaluation

After finalizing our mini search engine implementation and testing the system on our own machines, we needed to acquire other testers with other points of view and perspectives to test, assess, and comment on our application. Even though the majority of people ignore it, user assessment is one of the most crucial components of product publishing. Typically, people/companies deprioritize user assessment in order to focus on what they consider to be more critical characteristics of their product. Skipping user evaluation might have serious repercussions. Four people thoroughly tested our system. The users ranged in age from 18 to 34 years.

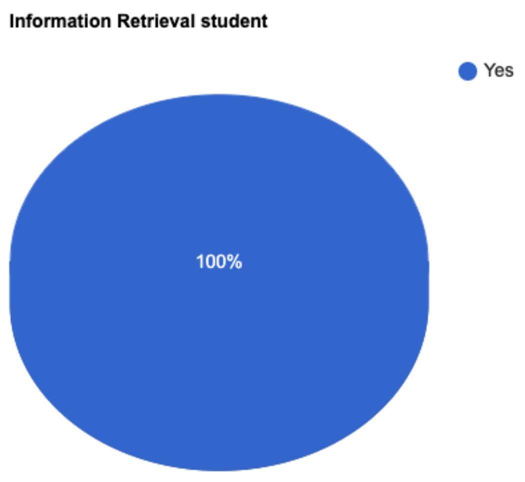

Figure 1: User Evaluation Target Audience

Identifying our target audience is the first stage in User Evaluation. The preceding pie chart shows that our program was completely and carefully examined by Information Retrieval students.

The following is some general comments about our application:

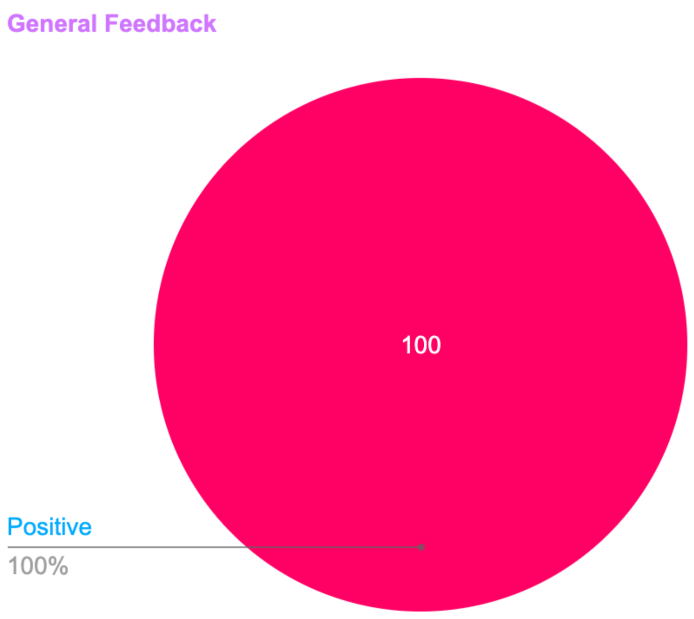

Figure 2: General Search Engine Feedback

As can be seen above, the great majority of our users were really satisfied with how our application worked.

Regarding the complexity of our application, despite the fact that the User Interface we used was basic and straightforward, with a simple search bar in the middle, an input box, and a search button (Google-style), one of our users considered our IR system a bit confusing. This piqued our interest, and it appeared that users who found our system a bit sophisticated would utilize search engines with a lot of information on their main page, such as Yahoo.

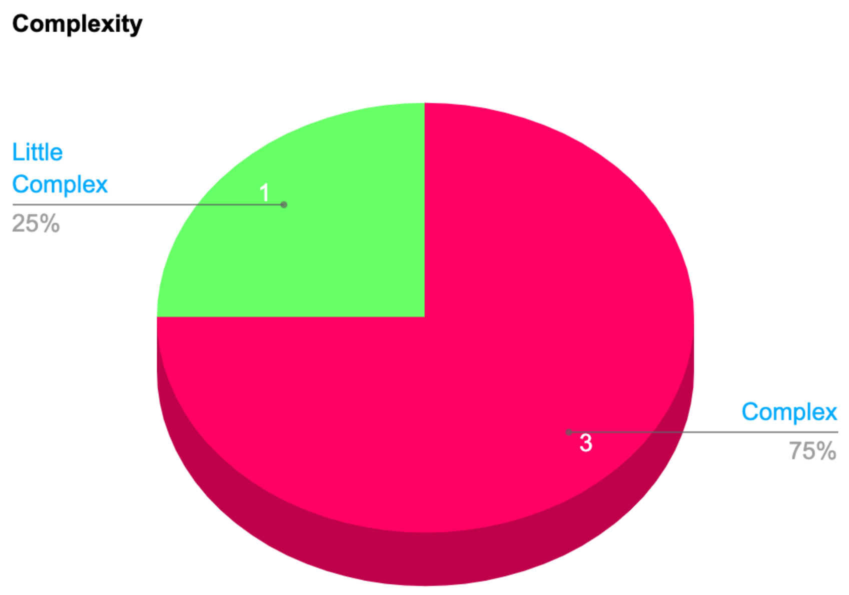

Figure 3: Application Complexity

Fortunately, none of our testers would require external assistance to use/test our program, which was a relief because when we created this application, our primary aim was simplicity, i.e. not requiring other resources to utilize it.

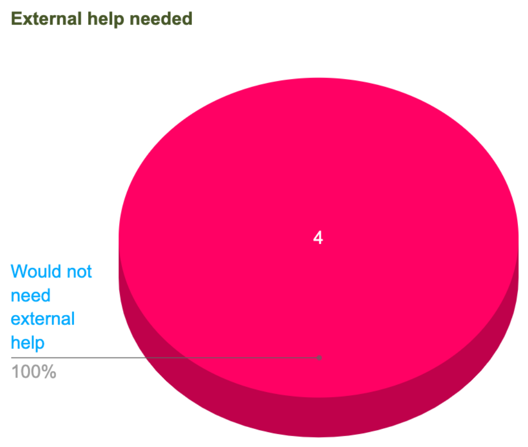

Figure 4: External Help

When building this system, we prioritized consistency. We wanted the user to be able to see all of the results that were relevant to their query. Unfortunately, we did not entirely meet our users' expectations in this regard, as seen by their ratings: one out of two believed the system was a bit erratic/inconsistent, while the other discovered no errors at all.

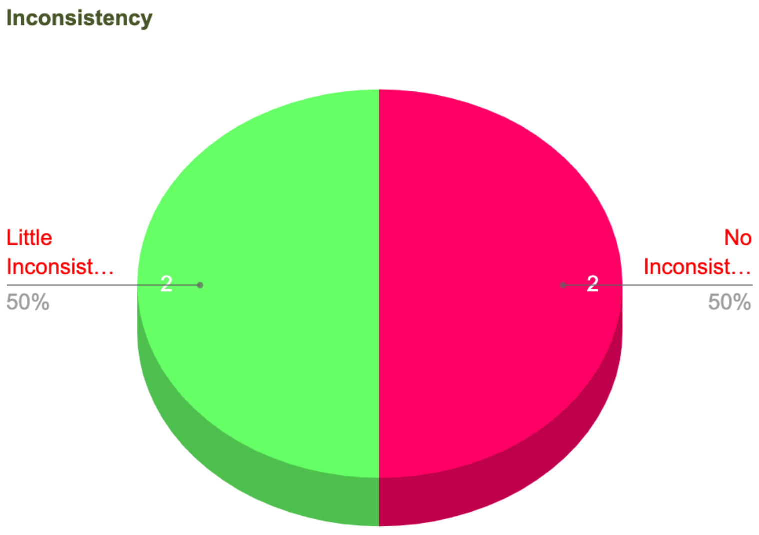

Figure 5: System inconsistency

Our customers were convinced that they would achieve their goals by utilizing our system.

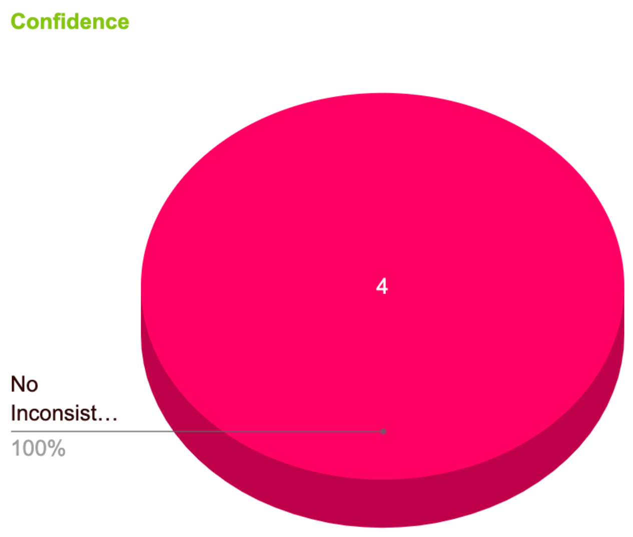

Figure 6: User Confidence

In terms of the efficacy of the results, the great majority of our users discovered the motorcycle they were looking for.

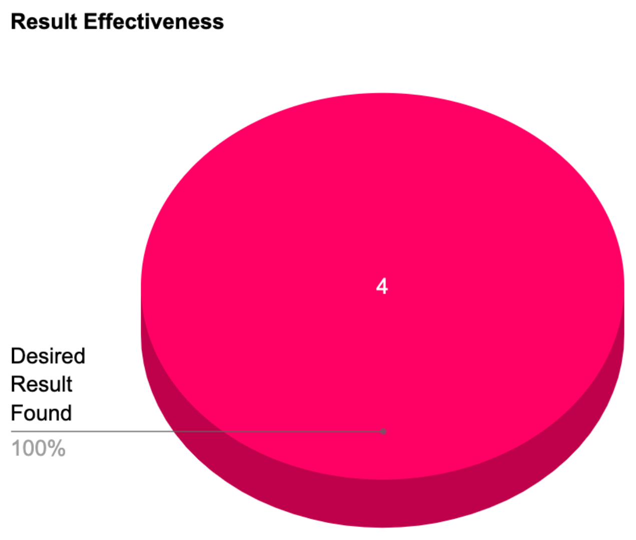

Figure 7: Result Effectiveness

We requested users to assess our search engine at the end of the user evaluation session, taking into consideration the aforementioned pie charts and their overall experience while testing our crawler. The outcomes were mostly positive, as seen below:

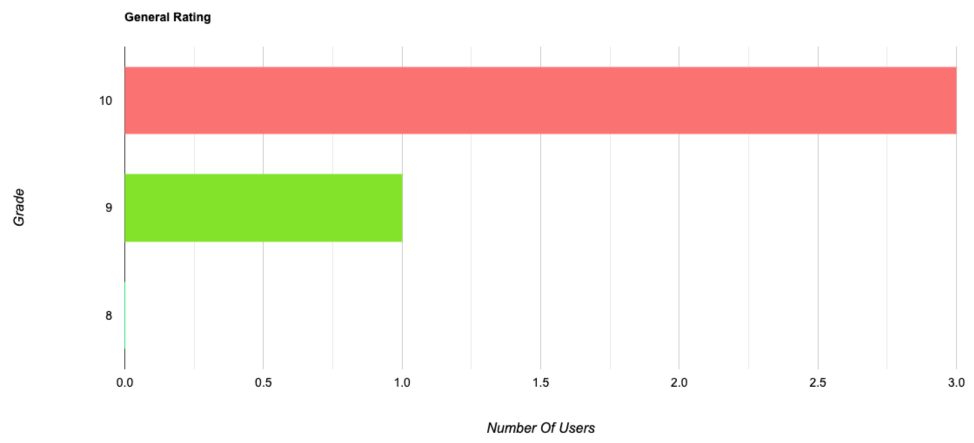

Figure 8: System Rating

As an extra step, we asked users what might have been done better while building this search engine. We listened to their suggestions and worked hard to improve our crawler while there was still time. Typical suggestions include, but are not limited to:

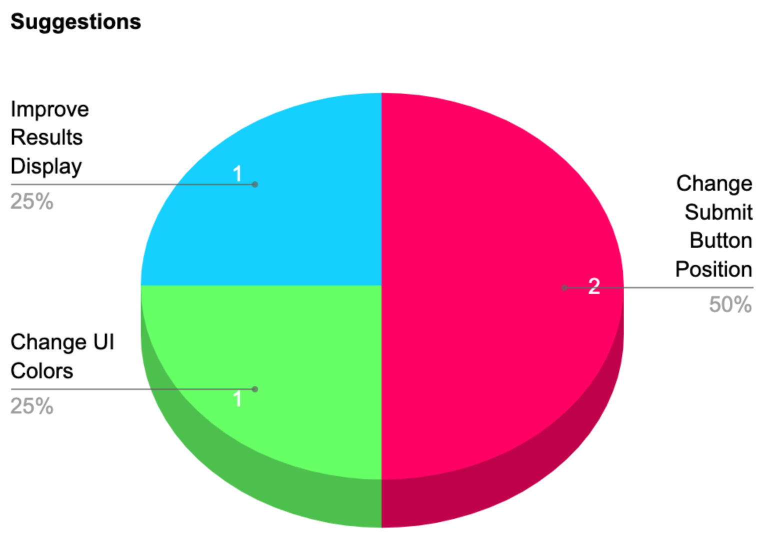

Figure 9: User's Suggestions

Last but not least, we asked users their thoughts about our advanced features. Our advanced feature was **automatic recommendation**. 75% of our testers liked the automatic recommendation feature, while the rest 25% were not really impressed by it.

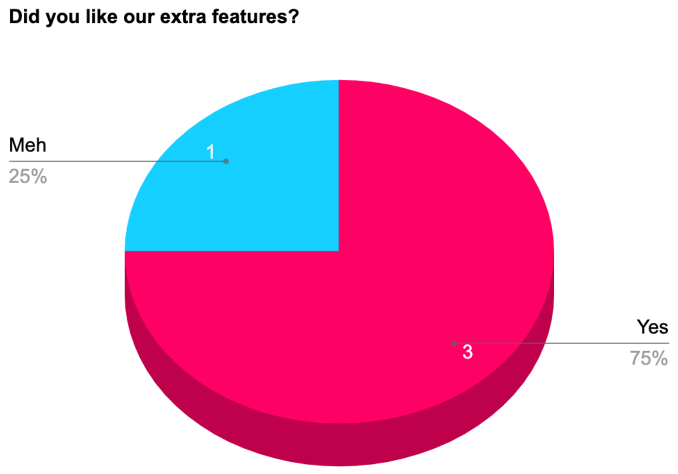

Figure 10: Recommendation Review