Formula 1 Search System

Diogo Nunes
up201808546@up.pt
FEUP
Portugal

Jéssica Nascimento
up201806723@up.pt
FEUP
Portugal

João Vítor Fernandes
up201806724@up.pt
FEUP
Portugal

Abstract

There is much information on the Internet about Formula 1. However, as it keeps growing, information sources also increase and some details might be in one and not in another. In this project, we want to keep a solid database with every detail about Formula 1 since the very first day. We have data since day 1 and we will create a Search System around that information, lettings users search about any race, driver, circuit, regulation, etc. In this Milestone, we focused on 3 datasets of our set of datasets directly about the races, we merged and indexed them to Solr to make queries about their qualifyings and races.

Keywords: datasets, search system, formula 1, scraping, crawling, wikipedia, solr, tokenizers, filters, queries

1 Introduction

Formula 1 is a rich topic in terms of information. We can talk about drivers, constructors (teams), circuits, each race, each championship, even about the regulations that are constantly changing and their history. In this project, we will try to construct a consistent database that joins all these topics described, so that everyone can search for some keywords and have results ordered by their relevance.

In this Milestone, we will only use the information about each race. On the next and last Milestone, we will include also the information about drivers, constructors, seasons, and circuits.

Obviously, we won't start with zero information: we resorted to datasets present in Kaggle [1], more specifically Formula 1 World Championship (1950-2021) dataset [2] and Wikipedia [3].

1.1 Search Scenarios

With the datasets provided, our goal is to be able to create results for the most trivial researches on Formula 1 but also some much more complex. Below we have examples of searches that we want to achieve:

- Searches on Drivers
  Search: Antonio Giovinazzi
  Answer: Personal data information, in which constructor team is included, important results and important moments in which he was included (incidents, important overtakes, ...) and career description.
- Searches on Constructors
  Search: Ferrari
  Answer: The history of the team in F1, memorable results, best drivers racing for the team, actual drivers on the team (if it is the case).
- Searches on Seasons (year)
  Search: 2020
  Answer: Summary of the full season, crucial highlights, driver and constructor champions, ...
- Searches on Circuits
  Search: Monaco
  Answer: Location, how many times that city/country hosted a race (if it is the case), most successful drivers on that track, description about the track (how many turns, how many DRS zones, ...).
- Searches on specific moments
  Search: Youngest driver winners
  Answer: List of races and drivers that, at that time, were the youngest drivers to win a race, to become champions.
  Search: Kimi Räikkönen made contact with Antonio Giovinazzi
  Answer: Moments where both drivers crashed each other, maybe crashing with other drivers as well.
  Search: Kimi Räikkönen overtook Antonio Giovinazzi
  Answer: Moments where both drivers were fighting for position, and a brief description about it, as well as individual information about both drivers.
  Search: Kimi Räikkönen started on pole
  Answer: Races where a certain driver started on P1, most successful qualifyings from him, best results, and driver career description.
  Search: Who started on grid 5 on 2009 Australia GP?
  Answer: The driver that started on grid 5 on Australia's 2009 Grand Prix.
  Search: Qualifications for Belgian Grand Prix on 2018
  Answer: Report paragraphs on the main qualification events for the Belgian Grand Prix on 2018
  Search: Hamilton crash
  Answer: Report of the races where the driver Hamilton crashed

With these search scenarios in mind, the goal is to obtain multiple paragraphs that include all those pieces of information. For instance, when searching about a driver we can get multiple paragraphs where we have information about each race he had competed in or about his personal life. To get this information we used Wikipedia, as will be described further down.

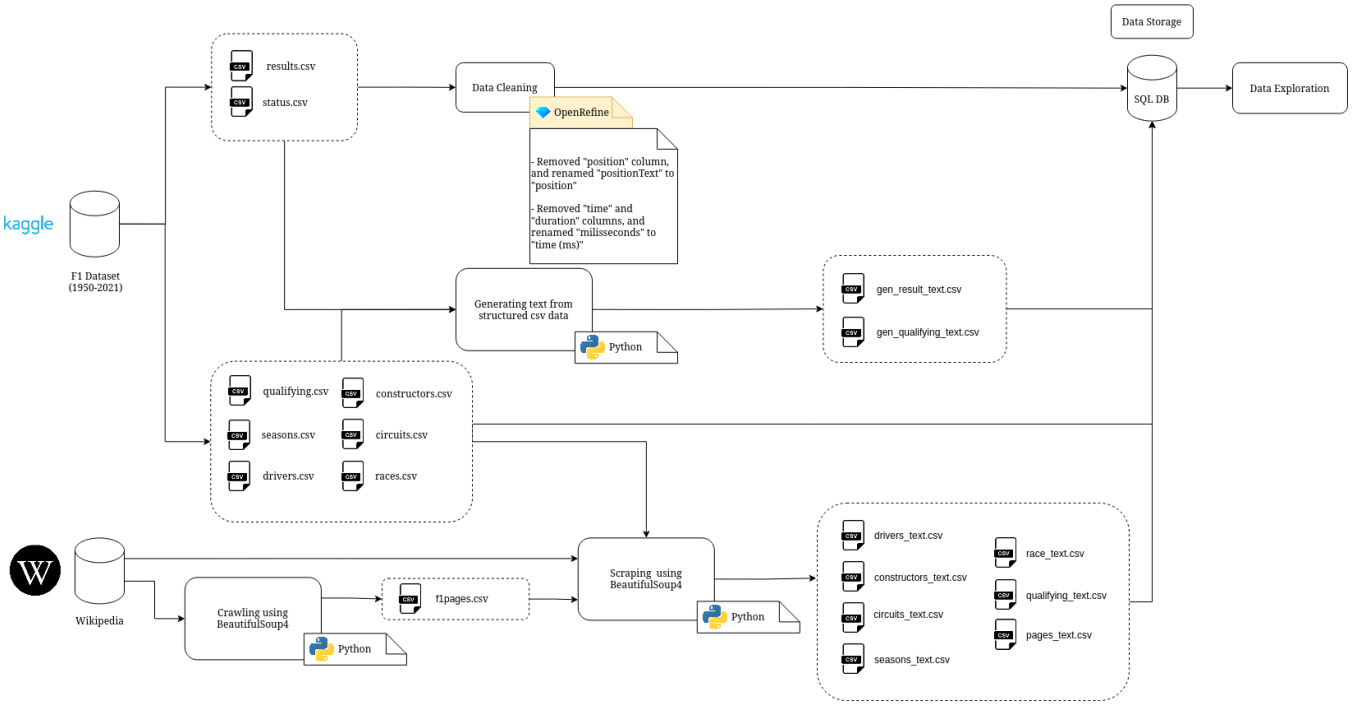

Figure 1. Data Processing Pipeline

2 Data Collection

2.1 Datasets

This dataset contains information about almost every detail about Formula 1 since the very beginning of the sport (1950) and it was compiled from Ergast Developer API [4].

The dataset is composed of 13 CSV files: circuits, drivers, constructors, seasons, qualifying, races, status, results, lap_times, driver_standings, constructor_standings, constructor_results, pit_stops.

However, we believe we have already so much information using only 5 of them: races, seasons, drivers, constructors, and circuits.

races.csv: raceId, year, round, circuitId, name, date, time, url.

circuits.csv: circuitId, circuitRef, name, location, country, lat, lng, alt, url.

drivers.csv: driverId, driverRef, number, code, forename, surname, dob, nationality, url.

constructors.csv: constructorId, constructorRef, name, nationality, url.

seasons.csv: year, url.

In this Milestone, we will focus on races.csv and only search for that information.

2.2 Wikipedia

We have URL links to Wikipedia on 5 CSV files: races, seasons, circuits, constructors, and drivers. We extracted data from them to enrich our database with text information, and not only concrete data.

2.3 Conceptual Model

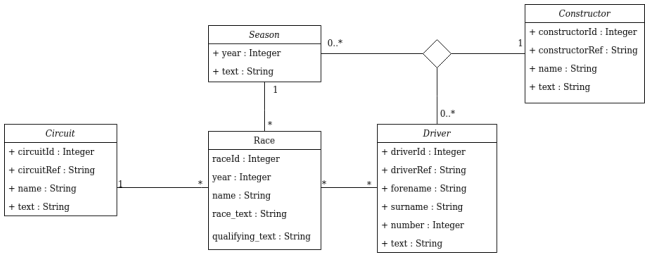

Figure 2. Data Domain Model

In the figure 2, we can further understand the relationships between our data. A circuit represents the race track where a race occurred. There are many races every season (yearly), and drivers participate in them. These drivers can change their constructor (team) but not in the middle of a season. All these classes have a textual field, scraped from different sources.

3 Data Preparation

Since seasons, drivers, constructors, circuits, and races files had URLs to Wikipedia pages, scraping was done using BeautifulSoup4, in order to get more large textual information. Thus, we got all paragraphs existing on each page.

For this milestone, we merged races.csv (content described above) with qualifying_text.csv and race_text.csv into a file named races_joined.csv, that have all the paragraphs extracted from each race related to their qualifying and/or race. This means that, at this point, one document is a single race, described by its raceId, year, round (number of the race on that year), circuitId, name, date, time, URL, qualifying_text (column with all qualifying paragraphs of that race) and race_text (column with all race paragraphs of that race).

To understand what information we have, we created a python program to obtain the average number of words in our races_joined.csv, that is composed by 1058 documents about each race performed until now:

Field name	Average word number
title	3
race_text	625
qualifying_text	146

Figure 3. Most common words in paragraphs about races

Figure 4. Most common words in paragraphs about the qualifying sessions

4 Information Indexing and Retrieval

The information retrieving tool used was Solr 8.10. A schema of the indexed documents was defined on a schema.json file, which defines how the documents are indexed into Solr. This schema defines how documents are processed during indexing and querying. This is done through creating and assigning field-types, as well as using the built-in tokenizers and filters.

We named the field-type that processes our scraped textual information as "text". This is the case for fields such as:

- race_text
- qualifying_text
- name

The default tokenizer was used to break the streams of text into tokens, since this satisfied our needs. White-space is treated as a delimiter, which was the main desired behavior.

When it came to filtering results, as the first step, all characters are converted to their ASCII equivalents, this defines the number of different characters that are taken into account, avoiding that rarer or non-standard characters disrupt the searching process. Then tokens are then turned into their respective lower-case and singular form, this prevents having to case match or provide the exact grammatical number for a noun, in order for relevant documents to be shown.

This refinement is applied both on indexing and on querying time, with the help of the following Solr's built-in filter elements:

- ASCIIFoldingFilterFactory
- LowerCaseFilterFactory
- EnglishMinimalStemFilterFactory
- ManagedSynonymGraphFilterFactor
- FlattenGraphFilterFactory

Information collected was joined into a single file named races_joined.csv, combining the gathered reports from the qualifying section and the final race. The resulting documents have the following attributes:

- raceId
- year
- round
- circuitId
- name
- date
- time
- url
- qualifying_text
- race_text

A result document could, for example, return the following information:

	Values
raceId	1054
year	2021
name	Portuguese Grand Prix
qualifying_text	Qualifying started at 16:00 local time (...)
race_text	The 2021 Portuguese Grand Prix (...)

hidden columns will only be used on the next Milestone

5 Evaluation

In this section, we will evaluate the results obtained, so that we can have an objective picture of the performance of our system. Basically, we will present Average Precision, Precision at 10 and Recall at 10 to evaluate if our queries are presenting the relevant results we wanted to present.

5.1 Query 1

Information need: Youngest drivers to ever win a race or a championship?

- q: race_text:youngest race_text:driver race_text:win
- q.op: AND

Results:

	0	1
0	Metric	Value
1	Average Precision	0.689546
2	Precision at 10 (P@10)	0.6
3	Recall at 10 (R@10)	0.6

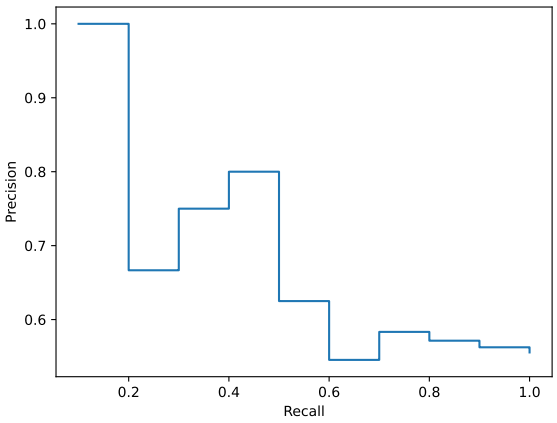

Figure 5. Precision-Recall Curve of Query 1

5.2 Query 2

Information need: F1 incidents in the last 2 years that involved a certain driver

- q: race_text:"incident Verstappen"~10 race_text:"crash Verstappen"~10 race_text:"accident Verstappen"~10 race_text:"collision Verstappen"~10
- q.op: OR
- fq: date:[NOW-1YEAR/YEAR TO NOW]

Results:

	0	1
0	Metric	Value
1	Average Precision	0.479167
2	Precision at 10 (P@10)	0.4
3	Recall at 10 (R@10)	1.0

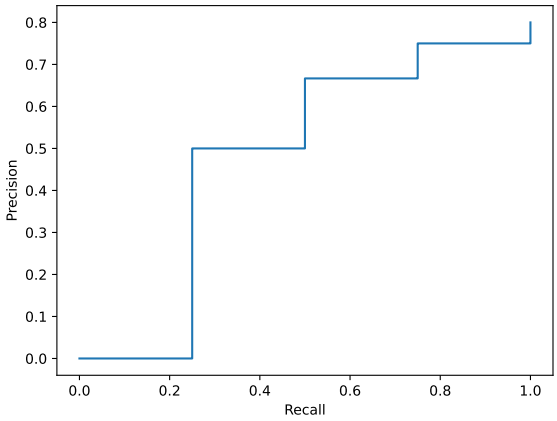

Figure 6. Precision-Recall Curve of Query 2

5.3 Query 3

Information need: Races where a driver X wins the race and rains during the race

- q: race_text:"rain" race_text:"Vettel win"~5
- q.op: AND

Results:

	0	1
0	Metric	Value
1	Average Precision	0.543333
2	Precision at 10 (P@10)	0.5
3	Recall at 10 (R@10)	1.0

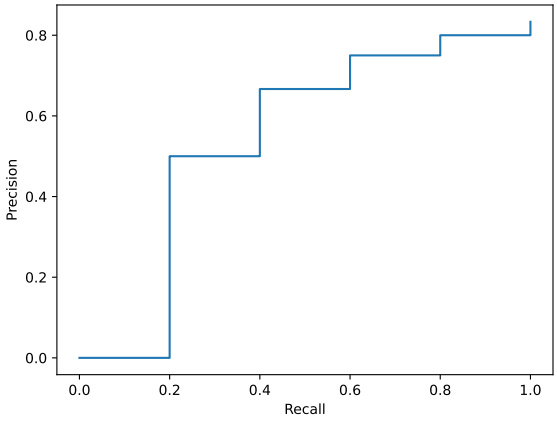

Figure 7. Precision-Recall Curve of Query 3

6 Conclusion

In the end of this Milestone, we constructed a well designed Search System capable of being "asked" (make queries) about

every single race realized until today. Some information might not be refreshed yet, because all of our text data was extracted from Wikipedia, but it's enough, at this moment, to show its potential.

In the next Milestone we are expecting to join other documents, so that we can also get results about each driver, team, circuit or even from a season summary.

References

[1] https://www.kaggle.com/
[2] https://www.kaggle.com/rohanrao/formula-1-world-championship-1950-2020
[3] https://en.wikipedia.org/wiki/Main_Page
[4] http://ergast.com/mrd/