# Pitchfork Music Reviews

Fabio Huang
Faculty of Engineering of the University of Porto
Porto, Portugal
up201806829@up.pt

Lisa Sonck
Faculty of Engineering of the University of Porto
Porto, Portugal
up202202272@up.pt

Temaco
Faculty of Engineering of the University of Porto
Porto, Portugal
up202200606@up.pt

## Abstract

Information communication technologies are ubiquitous in modern societies. An ever-growing number of activities depend on the ability to extract value from information. With this in mind a information retrieval system is built over a data-set of music reviews. The developing happened in three phases, the Milestones. The goal of the first milestone was to prepare, explore and characterize a selected data-set. As a result, a reproducible pipeline of data processing is achieved. In our case, a data-set about music reviews on Pitchfork (site best known for its daily output of music reviews) was retrieved from Kaggle. The data is analysed and explored using OpenRefine, which was then cleaned and refined using Python's Pandas Library [2]. To characterize the data, graphs about the data were formed and analysis was conducted. The second milestone was about Information Retrieval. For this the data-set was indexed using filters and tokens and represented in the tool Solr. To answer the defined information needs in milestone 1, queries were developed and executed on the system. These results were evaluated to see how good the indexed system can retrieve specific data.

***Keywords:*** Information Processing and Retrieval, Datasets, Pitchfork Music Reviews

## 1 Introduction

The amount of available data is growing every year [1] and with it, the importance of its correct management. Being able to efficiently use the data and extract interesting results is essential for further development of our current society. The course Information Processing and Retrieval at FEUP aims to enlighten us about information search systems and their use for large amounts of data. To put this knowledge into practice, we were tasked to develop a search system, with the corresponding tasks of data collection and preparation, and the information processing and retrieval. This project is divided into three milestones: 1. Data Preparation, 2. Information Retrieval and 3. Building the final Search System.
The first part of the report elaborates on the results and actions of the first milestone, which include:

- Choosing a dataset.
- Assessing the data quality and the authority of the source.
- Building a data processing pipeline.
- Cleaning of the data.
- Making a conceptual model of the data.
- Performing an exploratory data analysis.
- Characterizing the data.

The second part of the report talks more about the system built in Milestone 2 and the executed queries. The specific topics are:

- Solr.
- Our used Documents.
- The indexing procedure, with the used filters and tokens.
- The different evaluated queries with their results.

## 2 M1 - Data Prepatation

### 2.1 Data Collection

Developing a information search system requires a good and extensive dataset. The dataset should contain a lot of rows and columns, where at least one has an exhaustive, diverse body. This way text search can be implemented and interesting results can be obtained.

**2.1.1 Dataset Choice.** The choice of a dataset is important, since it determines the course of the project. At first the theme 'Music' was chosen, since a lot of large and comprehensive datasets exists for this subject and we have a big interest in music and its characteristics. We first found datasets were about artists, albums and songs, and the lyrics associated with each song, however, due to copyright reasons we were not able to get the complete content of the lyrics, and faced the problem of lacking textual data to work with. Luckily we managed to find a dataset that contained music reviews, which consisted of interesting comments and rich text that could be used for this project. [4]

**2.1.2 Dataset Content.** The selected dataset consists of a .sqlite file that contained the following tables:

- Artists table: contained the name of the artists (18.8k rows)
- Content table: contained the review's content (18.4 rows)
- Genres table: contained the song's genre (22.7k rows)
- Labels table: contained the album's label (20.2k rows)
- Reviews table: contained the title of the review, name of the artist, url link for the pitchfork's review page, rating score of the review, best new music, review's author name, author type, review's publication date and week day (18.4k rows)

- Years table: contained the year that the song was released (19.1k rows)

#### 2.1.3 Data Quality and Source.

Our dataset was retrieved from Kaggle [4], which is a popular community for data researchers and analysts and where the vast majority of the content is reliable. All of the content of this dataset was scraped from Pitchfork and since this dataset has over 10k downloads and over 8000 upvotes, we can infer that this source is reliable. The quality of the data meets our usage standard.

## 2.2 Pipeline

To build our Pipeline, we created python scripts using libraries such as Pandas [2], Matplotlib [5] and Seaborn [6] to manipulate the data and to generate graphs. OpenRefine [7] was also used to check for data facets. Our pipeline begins with a .sqlite file extracted from Kaggle, which contains multiple tables, as mentioned before. These tables are merged and a .json file is produced. This file is then processed and cleaned. At the end of the pipeline, the final dataset is used for data characterisation.

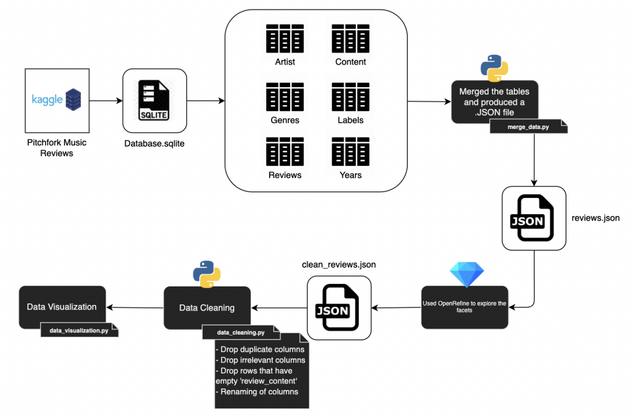


**Figure 1.** Pipeline Diagram

## 2.3 Data Cleaning

Before cleaning the data, we used OpenRefine [7] to explore the text facets on each column. After having an overall idea of the values of each column, duplicated and irrelevant columns were removed. Also entries with an empty review content were removed. At last some columns were renamed for an easier interpretation of the values.

## 2.4 Conceptual Model

The conceptual model is based on the initial database, extracted from Kaggle. However, some of the attributes are removed after the data was refined and cleaned. The final model can be seen in figure 2.

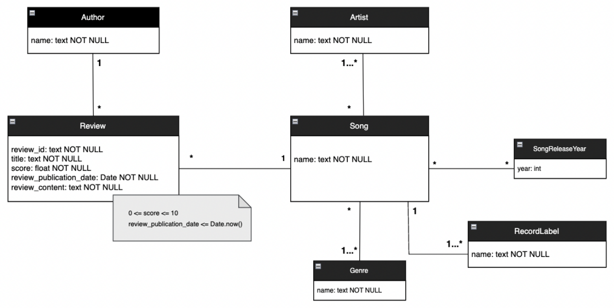


**Figure 2.** Conceptual Model

## 2.5 Dataset Characterisation

To better understand and grasp the data at hands, analysis and characterisation of the dataset was performed. To execute and visualize the characteristics of the dataset, the language Python was used with the packages Pandas [2] and Seaborn [6]. Pandas is very useful to handle big amounts of structured data and with Seaborn, graphs can be made easily. At first all the fields in the data set were investigated in a general way; Look at the possible values, what is the average value, are their outliers, etc. This initial research showed that the average score, given in a review, is 7.0 and the median is 7.2.

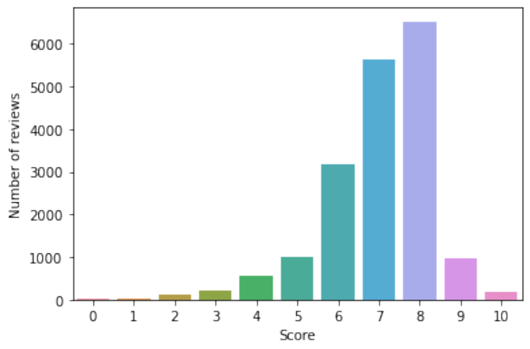


**Figure 3.** Number of reviews per rounded score

In figure 3 the number of reviews per score are represented. For every review the score is rounded to the nearest integer and this shows that the score eight is given the most, followed by seven.

After investigation of the genre of the songs of the different reviews, we concluded that most of the reviews are written of rock songs. This is seen in figure 4.

This first analysis about the data gave a general overview about its characteristics and how our data set was built. To see if certain fields of the data had a correlation, cross analysis over the different fields was performed. Figure 5 shows

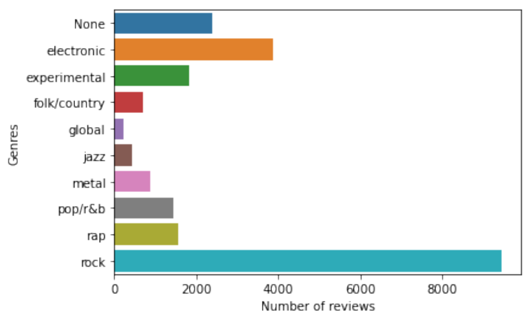


**Figure 4.** Number of reviews per song-genre

that the average score is approximately the same for every genre and equal to 7.0, which also establishes the analysis of the score over all the reviews.

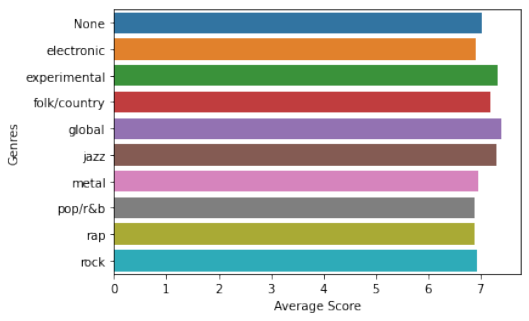


**Figure 5.** Average score per genre.

To have an overview about the length of a review, an analysis about the amount of words within one review was executed. In figure 6 the density of the amount of words in a review is plotted. This shows that most of the reviews consist of approximately 700 words. This is also concluded from the average word count and median, which are respectively equal to 703 and 649. To be able to make a representative further analysis, all the reviews who have a word count higher than 1500 are considered together.

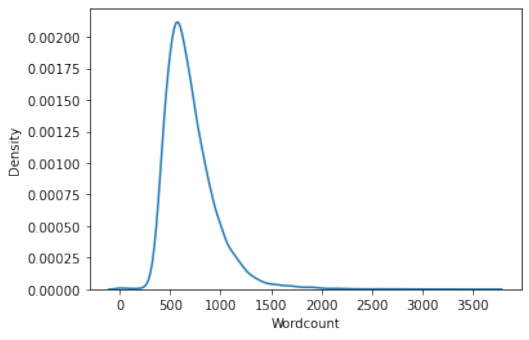


**Figure 6.** Density of the amount of words in a review.

As seen in figure 7 reviews for every genre have approximately the same amount of words. However, figure 8 shows an interesting result. It shows that the higher the word count of a review, the higher the average score is, with a difference of two points, on a scale of 10, between the shortest and longest reviews.

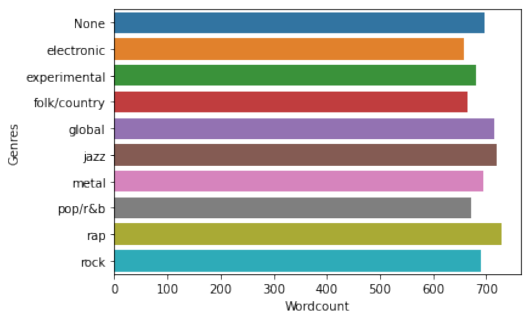


**Figure 7.** Average amount of words of a review per genre.

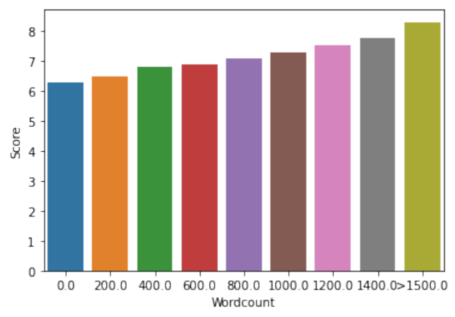


**Figure 8.** Average score of a review based on its word count.

Besides this, text analysis of the reviews has been executed, where the most common words of reviews with a score above the average are found. Also for reviews with a score above the average the most common words are searched for. These findings can be further explored during the next milestone.

## 2.6 Search tasks

After retrieving, cleaning and characterizing the data, insights about the properties of the data were acquired. With these insights, it was possible to devise useful queries for our search system.

- I want to find reviews of albums that include jazz and piano
  - Returns a list of albums where the review_content talks about jazz and piano.
- I want to find albums that are reviewed as boring.
  - Returns a list of albums where the review_content contains the word boring.
- I want to find songs and albums that are ideal for studying.

  - Returns a list of albums where the review_content contains words like calm, quiet, ambient, etc.
- I want to find albums that Rebecca Haithcoat reviewed as bad, vulgair, bland.
  - Returns a list of reviews by Rebecca Haithcoat, where the review_content contains words like bad, vulgair, bland, etc.

## 3 M2 - Information Retrieval

### 3.1 Solr

The chosen information retrieval tool is Solr. Solr can be used in a docker container and is highly reliable, scalable and fault tolerant. It provides distributed indexing, replication and load-balanced querying, automated failover and recovery, centralized configuration and more [8].

### 3.2 Used documents

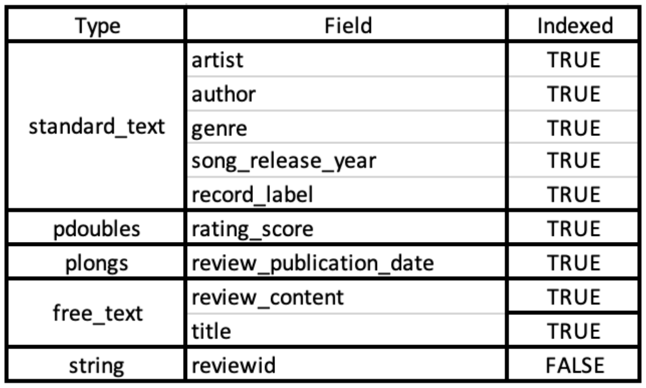
One review, together with all it's information, was used as a document in Solr. An example of a document, with all it's attributes, can be seen in figure 9. This choice of documents was made, because the main topic of our search system are the reviews. With the use of these documents all information regarding one review is kept together and querying over them is made easier .

```
{
  "reviewid":[13088],
  "title":["sketches of spain: legacy edition"],
  "artist":["miles davis"],
  "rating_score":[8.0],
  "author":["mark richardson"],
  "review_publication_date":[1244160000000],
  "genre":["jazz"],
  "song_release_year":"2009",
  "record_label":["legacy"],
  "id":"bd92093a-9f2b-4405-9a75-80ba845e12a4",
  "review_content":["Is this even jazz? Sketches of Spain was
  "_version_":1749576559401893888,
  "score":8.093981},
{
```

**Figure 9.** Example of a document.

## 4 Indexing

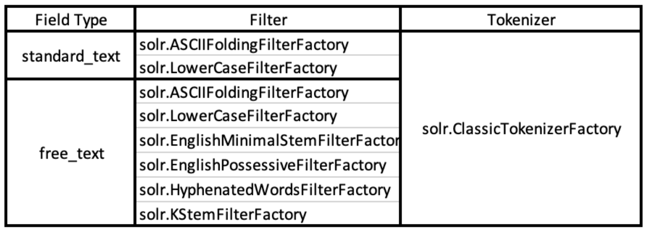
During the indexing process, we indexed most of our fields, except from the one that would not contribute for our information needs. Below in the figure 10, shows the schema that we have implemented:



| Type | Field | Indexed |
|---|---|---|
| standard_text | artist | TRUE |
| | author | TRUE |
| | genre | TRUE |
| | song_release_year | TRUE |
| | record_label | TRUE |
| pdoubles | rating_score | TRUE |
| plongs | review_publication_date | TRUE |
| free_text | review_content | TRUE |
| | title | TRUE |
| string | reviewid | FALSE |

**Figure 10.** Schema Fields.

There were field which used the Solr default field type, in case, 'review_publication_date' and 'reviewid' that were defined using the types 'plongs' and 'string' respectively. Custom field types were also defined, specifically 'standard_text' and 'free_text', which both contained the tokenizer *solr.ClassicTokenizerFactory*, additionally, different filters were also applied to each of them, which can be visualized in the figure below. Each filter has a different functionality, which alters the way the tokens can be used in a query [9].



| Field Type | Filter | Tokenizer |
|---|---|---|
| standard_text | solr.ASCIIFoldingFilterFactory | solr.ClassicTokenizerFactory |
| | solr.LowerCaseFilterFactory | |
| free_text | solr.ASCIIFoldingFilterFactory | |
| | solr.LowerCaseFilterFactory | |
| | solr.EnglishMinimalStemFilterFactor | |
| | solr.EnglishPossessiveFilterFactory | |
| | solr.HyphenatedWordsFilterFactory | |
| | solr.KStemFilterFactory | |

**Figure 11.** Schema Field Types.

### 4.1 Query 1: Boring songs

The first question answers the information need of finding albums that were criticized as boring by the authors of the reviews. The query searches for the word "boring" inside the review_content. If boring is found, the result will count as 5 times in the end-score, so reviews that often name the word "boring" are higher in rank. Besides this the query penalizes when the album has boring in the title, since this alters correct results and when boring occurs in the review_content near "never", "not" or "doesn't". The query also favors when "is boring" occurs in the review_content. At last the query also penalizes the results if boring is near a quotation mark, because this means that the word boring is not used as the exact definition of the word. Below the query insert in the q-field in Solr can be found.

review_content:boringˆ5
-title:boring

```
-review_content:"doesn't boring"~5
-review_content:"not boring"~5
-review_content:"never boring"~5
 review_content:"is boring"~5
-review_content:" "(")" boring "(")" "~3ˆ2
```

### 4.1.1 Obtained results.

This query was applied on our system, one time with the schema, as discussed in section **??**, and one time without schema, where Solr decides the indexing of documents automatically. In table 1 the values of the calculated metrics for our system can be found. The average precision was calculated, together with the precision at 5, 10, 15, 20 and 15 fetched documents. The values were determined for both setups.

| Metric | Values With Schema | Values Without Schema |
|---|---|---|
| Average Precision | 0.719355 | 0.740282 |
| Precision at 5 (P@05) | 1.0 | 0.8 |
| Precision at 10 (P@10) | 0.6 | 0.7 |
| Precision at 15 (P@15) | 0.6 | 0.666667 |
| Precision at 20 (P@20) | 0.65 | 0.65 |
| Precision at 25 (P@25) | 0.68 | 0.56 |

**Table 1.** Results with and without an applied schema.

Figure 12 shows the precision recall curves for both setups. Precision tells how many of the retrieved documents were relevant and recall tells how many of the relevant documents were retrieved.

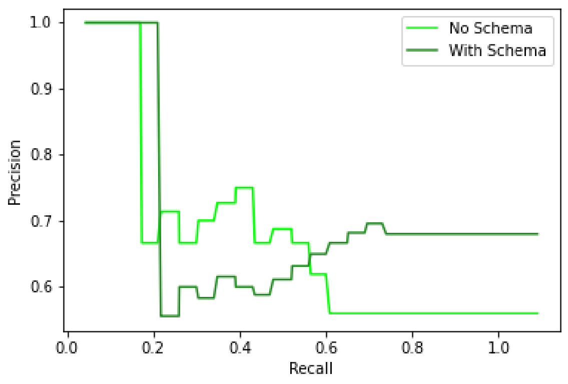


**Figure 12.** Precision recall curves of the results with and without an applied schema.

### 4.1.2 Evaluation.

When our schema was applied, the query returned 1114 results and without schema only 576 documents were retrieved. This is because of the usage of the different filters in our free-text filed-type. All words related to boring, e.g. bored, are also considered when executing the query. Because these amounts of documents are too much to evaluate manually, only the first 25 results in both setups are taken into account when calculating the metrics.

Using the schema yields a better precision for the first few results, as seen in table 1. Until 5 retrieved documents, the system with a schema has a precision while the system without a schema only has a precision of 80%. After that the precision of the system with schema drops suddenly to 60%. This is because the word bored or boring is used in sarcastic way, so not with it's original meaning. Our filters cannot handle these cases and thus yield a faulty result. The system without a schema also suffers from these cases, but only for the cases with boring and not bored. The moment an author uses a sarcastic comment with boring or bored, he often does it again, to illustrate how exciting the song is, which gives these results a high ranking in the query.

After these cases the precision of our system with schema increases again with documents that use the word boring in it's literal meaning. All kinds of words related to boring are taken into account in the system with schema, which yields a higher precision compared to the system without a schema. Without a schema only the word boring is taken into account and when it is used in another context than to describe the album, it has a higher influence on the results than in the one with schema, since no other words can be used as proof.

The precision-recall curves of both setups, as seen in figure 12 show a similar conclusion. The setup with schema reaches a recall of 0.2 with a precision of 1.0, while without a schema only a recall of less than 0.2 can be reached with precision 1.0. This means that the more first results are relevant in the system than in the system without schema. Between a recall of 0.2 and a little less than 0.6, the precision is better for the system without a schema. This can be explained by the use of sarcasm in the reviews (and that with schema the system is more sensitive to it, since it takes more words related to boring into account). When the recall reaches 0.6, the precision for the system with schema has again a better precision. It reaches a precision of 68% when 25 documents are retrieved, compared to 56% for the setup without a schema. This means that more relevant documents were retrieved in total for the system with schema.

## 4.2 Query 2: Songs for studying

The second question answers the information need of songs and albums which are ideal for studying. Although it depends person to person which kind of music is ideal for studying, here, we consider them to be songs that are calm and peaceful. The query searches for the words "calm", "quiet", "ambient", "relaxing" and "classical music" that are present inside the review_content and exclude the word "violent". We favor "jazz" above the others, because jazz music tend to be more appropriate compare to the other genres available, which are (Rock, Electronic, Hip-Hop, R&B, Country...). After the query is done, we filter the ones that are considered to be really good (in terms of rating score).

**q:**

- genre:jazzˆ2
- review_content: calm
- review_content: quiet
- review_content: ambient
- review_content: relaxing
- review_content: "classical music"
- review_content: "violent"

**fq:**

- rating_score: [8.0 TO 10.0]

#### 4.2.1 Obtained results.

In table 2 the values of the calculated metrics for our system can be found. The average precision was calculated, together with the precision at 5, 10, 15, 20 and 15 fetched documents. The values were determined for both setups.

| Metric | Values With Schema | Values Without Schema |
|---|---|---|
| Average Precision | 0.475046 | 0.48338 |
| Precision at 5 (P@05) | 0.6 | 0.8 |
| Precision at 10 (P@10) | 0.6 | 0.6 |
| Precision at 15 (P@15) | 0.466667 | 0.466667 |
| Precision at 20 (P@20) | 0.35 | 0.35 |
| Precision at 25 (P@25) | 0.28 | 0.28 |

**Table 2.** Results with and without an applied schema.

Figure 13 shows the precision recall curves for both setups for query 2.

#### 4.2.2 Evaluation.

Since the results were quite similar and difficult to distinguish which one was better, the AUC (Area Under Curve) was calculated:

| System | AUC value |
|---|---|
| Query 2 Without Schema | 0.4964 |
| Query 2 With Schema | 0.5309 |

**Table 3.** AUC for Query 2.

From the table 3 above, we can conclude that the system with Schema does indeed has a slightly improved compared with the one without

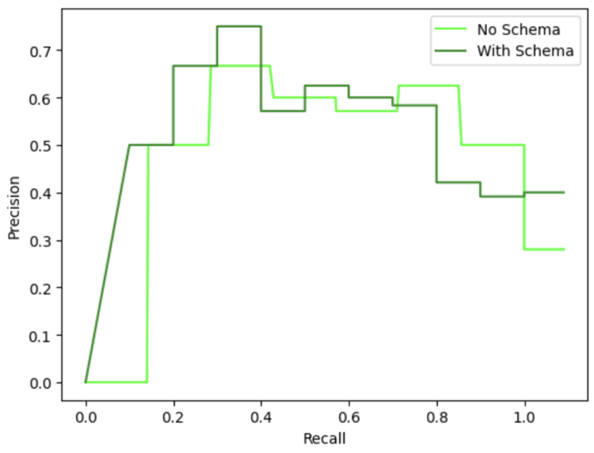


**Figure 13.** Precision recall curves of the results with and without an applied schema for query 2.

### 4.3 Changes to Milestone 1

The search tasks in Milestone 1 were not optimal after the first iteration. We changed these during the second Milestone, so the information needs are more clear and querying could be done in a more optimized way.

## 5 Conclusion

The first milestone was successfully completed, since a relevant dataset for the project has been selected, prepared and characterised. The quality of the data and authority of the source have been assessed and approved. The data has been analysed and characteristics have been composed. A data processing pipeline was developed and a conceptual model for the data has been established.
For the second milestone, we were able to get familiar with an information retrieval tool (Solr). During this milestone, the dataset resulting from the previous milestone was uploaded to Solr, by testing our information needs using 2 different schemas (one with schema and one without schema), we found out that the use of schema indeed helps us boost the performance to search for our information needs. However only a slight improvement was seen when comparing the results, but a better choice of schema could improve it more. For future work and improvements, we could add a few more information needs, additionally, we could also carry out more experiments using different queries and improving our schema.

## References

[1] Idera. Data Growth. https://www.idera.com/glossary/data-growth/
[2] [n.d.]. Pandas. https://pandas.pydata.org/
[3] [n.d.]. Beatiful Soap. https://www.crummy.com/software/BeautifulSoup/bs4/doc/
[4] Nolan Conaway. 18,393 Pitchfork Reviews. https://www.kaggle.com/datasets/nolanbconaway/pitchfork-data
[5] [n.d.]. Matplotlib. https://matplotlib.org/
[6] [n.d.]. Seaborn. https://seaborn.pydata.org/
[7] [n.d.]. OpenRefine. https://openrefine.org/
[8] [n.d.] Solr. https://solr.apache.org/
[9] [n.d] Query Syntax and Parsing. https://solr.apache.org/guide/8_10/que
syntax-and-parsing.html