Information Retrieval - Anime List -

DAPI

Master in Informatics and Computing Engineering

2018/2019

Andreia Rodrigues - up201404691
Francisco Queirós - up201404326
Miriam Gonçalves - up201403441

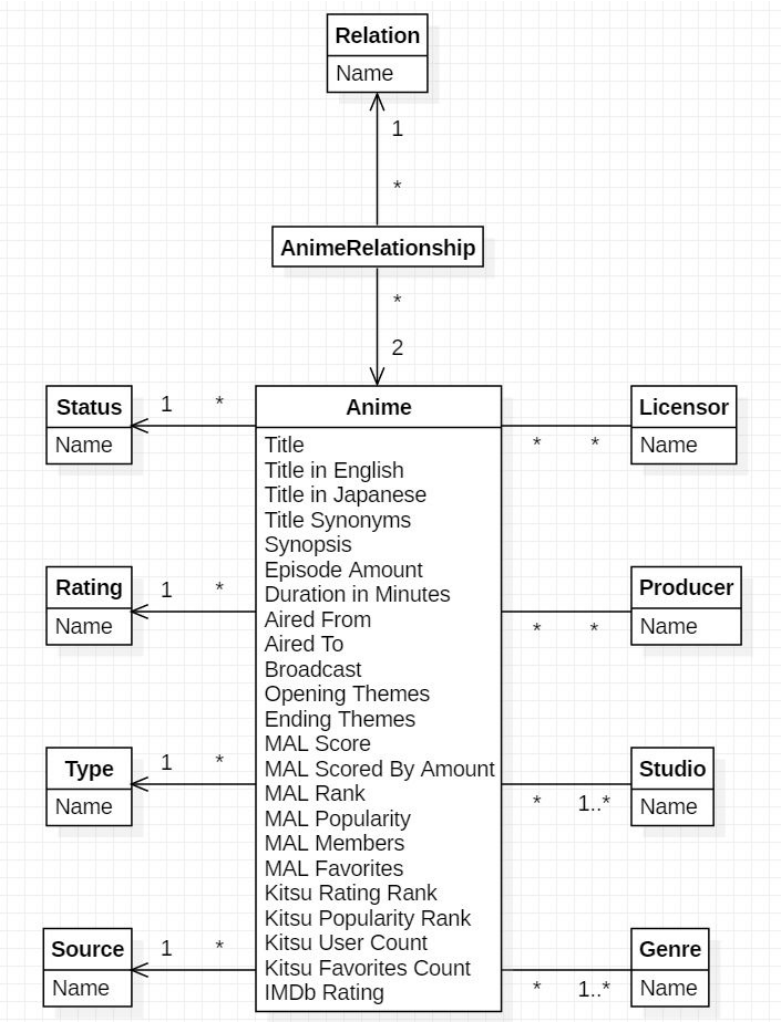
U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Dataset

➜ **Collection:** All animes in the dataset obtained from milestone 1.

➜ **Document:** An anime.

➜ The dataset is in a single **CSV** file.



Information Retrieval Tool

Elasticsearch + Logstash + Kibana

Information retrieval engine that is central to the solution package offered by Elastic.

Tool used to insert information into Elasticsearch's index.

Tool that provides a user interface that sends HTTP requests to Elasticsearch, allowing to make queries and visualize the results better.

Data processing

Configuration file to import dataset file to Elasticsearch:

```
input {
    file {
        path => <csv_path>
        start_position => "beginning"
    }
}
```

Data processing

```
filter {
    csv {
      separator => ";"
       columns => ["anime_id","title","title_english",...]
      convert => {
            "anime_id" => "integer"
            "episodes" => "integer"
            …
            }
      json{
            source => "aired"
            target => "aired"
      }
      ...
}
```

Data processing

```
output {
    stdout { codec => rubydebug }
    elasticsearch {
        action => "index"
        hosts => ["127.0.0.1:9200"]
        index => "anime"
       document_id => "%{anime_id}"
        workers => 1
    }
}
```

Queries

➜ **Bool query:** matches documents matching boolean combinations of other queries

➜ **Multi-match query:** built on the match query to allow multi-field queries

➜ **Common terms query:** an alternative to stopwords, improving both precision and recall of results

➜ **Function-score query:** allows modifying the score of documents that are retrieved by a query

Queries

➜ This query can be used to search for an anime for its different titles, synopse or genre. It uses different metrics in order to achieve a higher grade of relevance.

```
{ "size" : 10,
  "_source": ["title", "title_english", "title_japanese", "title_synonyms", "synopse", "rating"],
  "query": {
    "function_score": {
      "query": {
        "bool": {
            "should": [ {
                "multi_match": {
                  "type": "phrase",
                  "query": "Attack on Titan",
                  "fields": ["title^10", "title_english^10", "title_japanese^10", "title_synonyms^9", "synopse^5", "genre"],
                  "boost":3
                }
              },{
                "multi_match": {
                  "type": "best_fields",
                  "query": "Attack on Titan",
                  "fields": ["title^10", "title_english^10", "title_japanese^10", "title_synonyms^9", "synopse^5", "genre"]}}]},
      "field_value_factor": {
        "field": "scored_by",
        "modifier": "log2p",
        "missing" : 1}}}}
```

Different boosts are given to each fields relating to their potential to be relevant to the user's information need.

➜ In this query, we find the anime from studio Sunrise (as an example) that has the highest value in the scored_by field, which represents the number of score votes for that anime.

```
{"query": {
    "bool": {
      "filter": [{
          "match": {
            "studio": "Sunrise"}}]}},
  "sort": [{
      "scored_by": {
        "order": "desc"}}],
  "size": 1}
```

➜ In this query, we select the anime which have "Nozomi Entertainment" as a licensor then aggregate these anime and calculate the average value of the score field. This way we know the average score of the anime from this licensor.

```
{"aggs" : {
        "anime" : {
            "filter" : { "match": { "licensor": "Nozomi Entertainment" } },
            "aggs" : {
                "avg_score" : { "avg" : { "field" : "score" }}}}}}
```

Information Retrieval Tool Evaluation

➜ Was found easy to use with the help of **Logstash** and **Kibana**

➜ Good **detailed documentation** provided helped to understand the **core functionalities** and existing **types of queries**

Difficulties:

➜ Figure out the differences between the types of queries and the right context where to use each

➜ Generate a data file (CSV file) that would be correctly interpreted by Elasticsearch