Information Retrieval (2021-2-F1801Q110)

Customized Wikipedia search engine

Renzo Arturo Alva Principe

Matricola: 746799

Introduction

Wikipedia is the well known multilingual collaborative encyclopedia which stores milions of articles accross multiple domains. It containes 6,243,012 articles only considering the english version and it averages 598 new articles per day. It is clear that such a large amount of data can easily create confusion if it is not well organized. Furthermore, along with number of articles, the variety of topics it covers highlights the need for a systematic method for searching for contents. For this reasons the aim of this project is to create a customized Wikipedia search engine capable of search on documents based on its content by taking into account also the topics dynamically extracted for each document.

The solution adopts Elasticsearch as search engine and considers Wikipedia pages as the source for indexing. Three different levels of abstraction are individuated in each article and used for different purposes during the search phase.

Dataset

Wikipedia is big project and host millions of pages which deals with many domains. In order to try to organize such knowledge, Wikipedia arranges pages by categories which are groups of articles on similar topics.

Since the dataset used in this course project is obtained from Wikipedia pages in the following I present the parts of a Wikipedia page that i keep in my dataset. The main three parts are:

- **title:** (blue rectangle) it can be seen as an identifier of the page or at least of the concept described by the page (I do not consider multi-lingual versions of pages)
- **abstract:** (red recatangle) is a concise description of the article which help the reader quickly ascertain the page's content
- **text:** (green square) it contains the full description of the page. In this case i incorporated the abstract into the test for indexing purposes (see later in the report)

Furthermore, articles are organized by categories and this allows me to choose a set of categories and for each of these select some article as sample of the whole Wikipedia corpus.
In particular my choice falls on three categories:

1) Golden Globe Award-winning producers
2) Guitar manufacturing companies of the United States
3) Grammy Lifetime Achievement Award winners

My intention was to select categories whose pages had some intersections in terms of words. For example the third category which is related to musical artist may have commonalities with pages of the second category as many award-winning artist are guitarrists. In a similar way the first and the third category may be related since they deal with award-winning artists (musicians, actors, producers) and maybe workds like fame, award, etc, are shared across this categories.

In order to create the dataset I choose the Wikipedia-API for Python which wrappes the official Wikipedia APIs and provides an easy way to access to the pages and metadata.

Consequently I downloaded 100 pages (on average ~17k words each) for each category defined above and keep the following informations:

- **url:** the URL to the wikipedia page
- **title:** as above
- **abstract:** as above
- **text:** as above
- **citations:** number of pages citing the page
- **citations_norm:** number of citations normalized

This data is organized in a JSON file and stored in the file system as a dump.

Search Engine

The search engine chosen for this project is Elasticsearch which is based on Lucene.
Elasticseatch indexes are fed by data in the JSON dump.
In order to better explain the index configuration I first define the functional goal of the three most important fields:

- **title:** is used for exact matching since i assume that the user who searches in this field have a clear idea of what "entity" is looking for and therefore if it inputs "The Band" (an artist) , the document titled "The Band" should be prioritized wrt "Muse (band)".
- **abstract:** is meant for exploratory research since it contains most of the keywords in the document
- **text:** is meant for phrase searching. Note that text field includes the abstract content

So as to obtain this diversification of the roles of the fields custom analyzers are defined:

- **text_analyzer**: includes a *standard tokenizer*, a *lower case and a synonym filters*
- **my_stop_analyzer**: includes *standard tokenizer*, *lower case* and *english stops filters*
- **abstract_analyzer**: includes standard tokenizer, *lower case*, *english stops* and *porter stemmer* fitlers

Therefore the mappings have been defined as follows:

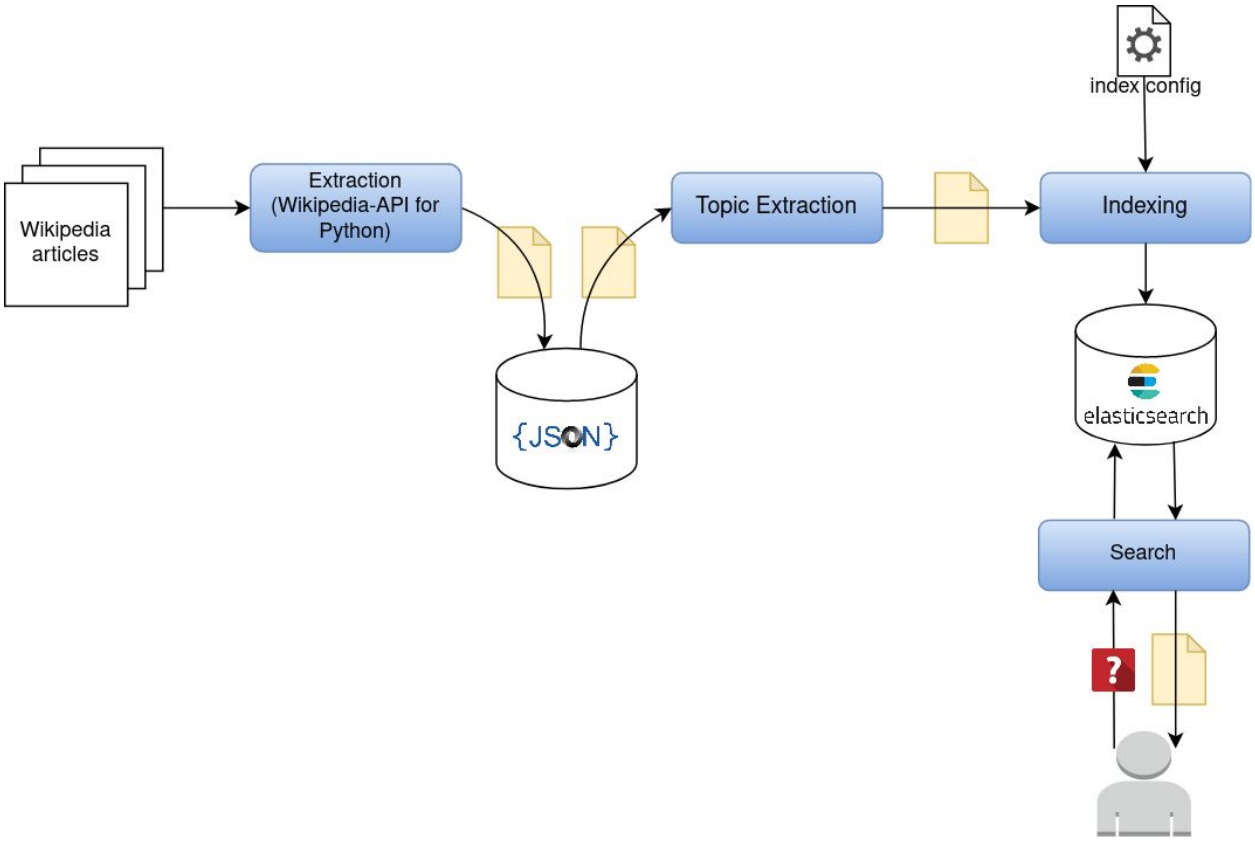
- **title**:
    - type: text
    - analyzer: whitespace
    - search_analyzer: whitespace
- **abstract**:
    - type: text
    - analyzer: abstract_analyzer
    - search_analyzer: abstract_analyzer

- **text**:
    - type: text
    - analyzer: text_analyzer
    - search_analyzer: my_stop_analyzer
    - search_quote_analyzer: text_analyzer
- **url**:
    - type: keyword
- **citations**:
    - type: long
- **citations_norm**:
    - type: double
- **topic**:
    - type: keword

Title field is analyzed both in indexing and search step with whitespace analyzer which just split into terms whenever it encounters a whitespace character. Abstract field is configured in index and query time to apply tokenization and set tokens to lowecase, remove stopwords and apply stemming. This choise should facilitate the user in a exploratory search. Finally text is set up for phrase search.
Concerning citations and citations_norm are used for filtering.

Finally the topic field was meant to filter the search results by topic. In particular, at query time, the system shows to the user a set of topics automatically extracted from the corpus during indexing phase and ask her/him to select one for filtering. Topic extraction is performed by using the latent Derichlet allocation which is a generative statistical model for automatically extract topics from documents by setting mainly the number of topic desired.

The following schema explicit better details about the data flow:



The first step is the extraction of wikipedia articles from the three chosen categories. Through Wikipedia-API i'm able to extract only the information needed from articles. Data is then stored in the file system as a JSON dump. Next step is automatic topic extraction using LDA model. Three topics are set to be extracted and then for each document one topic along with the other fields are indexed in an Elasticsearch instance. After the offline phase Elasticsearch can be queried by users.

Demonstration plan

Textual search on specific field:

query: {query: {match: {abstract:is an american pianist}}}

notes: it returns both alive and dead pianists (is/was) due to the analyzer.

score	title	citations	citations_norm	topic	url
5.07298	Morton Gould	361	0.0295	2	https://en.wikipedia.org/wiki/Morton_Gould
4.65563	Van Cliburn	385	0.0314	1	https://en.wikipedia.org/wiki/Van_Cliburn
4.55922	John Coltrane	2896	0.2363	0	https://en.wikipedia.org/wiki/John_Coltrane
4.44595	Charlie Haden	1310	0.1069	2	https://en.wikipedia.org/wiki/Charlie_Haden
4.31071	Nat King Cole	2121	0.1731	0	https://en.wikipedia.org/wiki/Nat_King_Cole
4.12157	Glenn Gould	594	0.0485	2	https://en.wikipedia.org/wiki/Glenn_Gould
3.62043	Count Basie	2355	0.1922	2	https://en.wikipedia.org/wiki/Count_Basie
3.59523	Bill Evans	1435	0.1171	0	https://en.wikipedia.org/wiki/Bill_Evans
3.49627	Duke Ellington	4612	0.3763	2	https://en.wikipedia.org/wiki/Duke_Ellington
3.41202	Fats Domino	1005	0.082	2	https://en.wikipedia.org/wiki/Fats_Domino
3.41029	Leonard Bernstein	3147	0.2568	1	https://en.wikipedia.org/wiki/Leonard_Bernstein
3.22629	Herbie Hancock	2693	0.2197	0	https://en.wikipedia.org/wiki/Herbie_Hancock

query: {query: {match_phrase: {text:was an american pianist}}}

notes: it returns only dead pianists

score	title	citations	citations_norm	topic	url
2.28328	Van Cliburn	385	0.0314	1	https://en.wikipedia.org/wiki/Van_Cliburn
1.96647	Fats Domino	1005	0.082	2	https://en.wikipedia.org/wiki/Fats_Domino

query: {query: {match_phrase: {text:is an american pianist}}}

notes: it returns only alive pianists

score	title	citations	citations_norm	topic	url
1.7958	Herbie Hancock	2693	0.2197	0	https://en.wikipedia.org/wiki/Herbie_Hancock

Textual search on a combination of fields

query:

```
{query: {bool: {
        must: {match: {abstract: guitarist}},
        must_not: [{match: {abstract: company}}, {match: {abstract: manufacturer}}],
        must: {range: {citations_norm: {gt: 0.500}}}}
}}
```

notes: it returns artists related to guitarists in the abstract of pages with many citations on Wikipedia

score	title	citations	citations_norm	topic	url
1	The Beatles	12256	1	0	https://en.wikipedia.org/wiki/The_Beatles
1	David Bowie	6587	0.5375	0	https://en.wikipedia.org/wiki/David_Bowie
1	Bob Dylan	9006	0.7348	0	https://en.wikipedia.org/wiki/Bob_Dylan

query:

```
{query: {bool: {
        must: {match: {abstract: guitarist}},
        must: {match: {text: drugs}}}
}}
```

notes: it returns all the guitarists that have a relation with drugs

score	title	citations	citations_norm	topic	url
3.41788	Jimi Hendrix	3936	0.3211	0	https://en.wikipedia.org/wiki/Jimi_Hendrix
3.10469	Stanley R. Jaffe	52	0.0042	1	https://en.wikipedia.org/wiki/Stanley_R._Jaffe
3.0805	Johnny Cash	4352	0.3551	0	https://en.wikipedia.org/wiki/Johnny_Cash
2.88893	Art Blakey	1925	0.1571	0	https://en.wikipedia.org/wiki/Art_Blakey
2.5434	The Allman Brothers Band	897	0.0732	0	https://en.wikipedia.org/wiki/The_Allman_Brothers_Band
2.52669	George Clinton (funk musician)	1499	0.1223	2	https://en.wikipedia.org/wiki/George_Clinton_(funk_musician)
2.50406	Miles Davis	4051	0.3305	0	https://en.wikipedia.org/wiki/Miles_Davis
2.47254	Rosemary Clooney	1249	0.1019	0	https://en.wikipedia.org/wiki/Rosemary_Clooney
2.47077	Billie Holiday	2617	0.2135	0	https://en.wikipedia.org/wiki/Billie_Holiday
2.32316	Al Green	1160	0.0946	0	https://en.wikipedia.org/wiki/Al_Green

Rank the pages taking into account the topics selected by the user

Topic 0: music, record, album, song, film, award, jazz, fame, includ, hall

Topic 1: film, award, best, academi, pictur, bear, music, director, nomin, includ

Topic 2: guitar, music, compani, instrument, manufactur, band, record, bass, electr, includ

Judging by the top words contained in the lists we can say that:

- topic 0 represent the categoryof the Grammy award winning artists
- topic 1 represents the category of the award winning producers
- topic 2 represents the category of the guitar manufacturing companies

In the following queries I searched for “guitar” and then i filter by topic 0 in the first example. Search returns artist that are guitarists. In the second example I filter by topic 2 and results returned are related to guitar manifacturing companies.

```
insert a keyword
guitar
insert topic id
0
```

score	title	citations	citations_norm	topic	url
1.48442	Carter Family	677	0.0552	0	https://en.wikipedia.org/wiki/Carter_Family
1.35722	Buddy Guy	1050	0.0857	0	https://en.wikipedia.org/wiki/Buddy_Guy
1.25954	The Allman Brothers Band	897	0.0732	0	https://en.wikipedia.org/wiki/The_Allman_Brothers_Band
1.24451	The Band	1600	0.1305	0	https://en.wikipedia.org/wiki/The_Band
1.11239	The Everly Brothers	1121	0.0915	0	https://en.wikipedia.org/wiki/The_Everly_Brothers
1.06707	Jimi Hendrix	3936	0.3211	0	https://en.wikipedia.org/wiki/Jimi_Hendrix
1.0253	Buddy Holly	1318	0.1075	0	https://en.wikipedia.org/wiki/Buddy_Holly
1.00529	Bo Diddley	1244	0.1015	0	https://en.wikipedia.org/wiki/Bo_Diddley
0.857422	George Harrison	3721	0.3036	0	https://en.wikipedia.org/wiki/George_Harrison
0.817293	Chuck Berry	2456	0.2004	0	https://en.wikipedia.org/wiki/Chuck_Berry

```
insert a keyword
guitar
insert topic id
2
```

score	title	citations	citations_norm	topic	url
1.92941	ES Guitars	1	0.0001	2	https://en.wikipedia.org/wiki/ES_Guitars
1.92031	Oktober Guitars	3	0.0002	2	https://en.wikipedia.org/wiki/Oktober_Guitars
1.90089	Moniker Guitars	151	0.0123	2	https://en.wikipedia.org/wiki/Moniker_Guitars
1.9004	Becker guitars	1	0.0001	2	https://en.wikipedia.org/wiki/Becker_guitars
1.89267	Kiesel Guitars	173	0.0141	2	https://en.wikipedia.org/wiki/Kiesel_Guitars
1.88756	Kramer Guitars	267	0.0218	2	https://en.wikipedia.org/wiki/Kramer_Guitars
1.88247	MotorAve	153	0.0125	2	https://en.wikipedia.org/wiki/MotorAve
1.87507	C. F. Martin & Company	344	0.0281	2	https://en.wikipedia.org/wiki/C._F._Martin_%26_Company
1.87212	Earthwood	5	0.0004	2	https://en.wikipedia.org/wiki/Earthwood
1.87212	Jerry Jones Guitars	8	0.0007	2	https://en.wikipedia.org/wiki/Jerry_Jones_Guitars

In a similar way i searched for the keyword “suicide” filtering by topic 0 and then by topic 1. Results returned an artist which wife committed suicide for the first case. In the second case returned two actors who worked in films that contain the word “suicide”

```
insert a keyword
suicide
insert topic id
0
+---------+----------+-------------+------------------+---------+-----------------------------------------+
|   score | title    |   citations |   citations_norm |   topic | url                                     |
|---------+----------+-------------+------------------+---------+-----------------------------------------|
| 4.34247 | Al Green |        1160 |           0.0946 |       0 | https://en.wikipedia.org/wiki/Al_Green  |
+---------+----------+-------------+------------------+---------+-----------------------------------------+
```

```
insert a keyword
suicide
insert topic id
1
+---------+---------------+-------------+------------------+---------+-------------------------------------------+
|   score | title         |   citations |   citations_norm |   topic | url                                       |
|---------+---------------+-------------+------------------+---------+-------------------------------------------|
| 6.36679 | Charles Roven |         127 |           0.0104 |       1 | https://en.wikipedia.org/wiki/Charles_Roven |
| 3.83448 | Sofia Coppola |         875 |           0.0714 |       1 | https://en.wikipedia.org/wiki/Sofia_Coppola |
+---------+---------------+-------------+------------------+---------+-------------------------------------------+
```

Fuzzy query

query: {query: {fuzzy: {title: {value: batles}}}}

notes: it returns "The Beatles" despite the misspelling

```
+---------+-------------+-------------+------------------+---------+-----------------------------------------+
|   score | title       |   citations |   citations_norm |   topic | url                                     |
|---------+-------------+-------------+------------------+---------+-----------------------------------------|
| 3.71521 | The Beatles |       12256 |                1 |       0 | https://en.wikipedia.org/wiki/The_Beatles |
+---------+-------------+-------------+------------------+---------+-----------------------------------------+
```

Expand the search adding synonyms of the words in the query

query: { query: {match: {abstract: philanthropist}}}

notes: it returns al the philantropist from the corpus which are producers

score	title	citations	citations_norm	topic	url
6.61818	Robert Chartoff	72	0.0059	1	https://en.wikipedia.org/wiki/Robert_Chartoff
3.79657	Kirk Douglas	1791	0.1461	1	https://en.wikipedia.org/wiki/Kirk_Douglas
2.93213	Ben Affleck	1615	0.1318	1	https://en.wikipedia.org/wiki/Ben_Affleck
2.72543	George Clooney	2081	0.1698	1	https://en.wikipedia.org/wiki/George_Clooney

query: {query: {match_phrase: {text: philanthropist}}}

notes: Since i intentionally declare "philanthropist" as synonym of "rock" in the text_analyzer filter, this query returns rocks stars

score	title	citations	citations_norm	topic	url
1.99976	Chuck Berry	2456	0.2004	0	https://en.wikipedia.org/wiki/Chuck_Berry
1.98269	Daisy Rock Girl Guitars	172	0.014	2	https://en.wikipedia.org/wiki/Daisy_Rock_Girl_Guitars
1.96761	Fats Domino	1005	0.082	2	https://en.wikipedia.org/wiki/Fats_Domino
1.95633	The Beach Boys	3272	0.267	2	https://en.wikipedia.org/wiki/The_Beach_Boys
1.9466	The Doors	2017	0.1646	0	https://en.wikipedia.org/wiki/The_Doors
1.94332	The Band	1600	0.1305	0	https://en.wikipedia.org/wiki/The_Band
1.94148	The Allman Brothers Band	897	0.0732	0	https://en.wikipedia.org/wiki/The_Allman_Brothers_Band
1.93776	Cream (band)	1322	0.1079	0	https://en.wikipedia.org/wiki/Cream_(band)
1.93042	Hal Blaine	596	0.0486	0	https://en.wikipedia.org/wiki/Hal_Blaine
1.92398	John Entwistle	554	0.0452	2	https://en.wikipedia.org/wiki/John_Entwistle

Conclusions

We can see how powerful and flexible a search engine like Elasticsearch can be. It allows a rich customization of document fields during indexing and search phase. It allows boolean operators, similarity by vectors, synonyms, fuzzy queries, etc.
Furthermore, the topic extraction of documents allows to filter results and get a more clean result.

All of this potential makes it clear how advanced search engines like Google can achieve such efficiency and effectiveness over a huge corpus.