MusicSurf

Raghav Mittal

U101114FCS111

NIIT University

19/April/2017

Introduction

MusicSurf is a search engine to search on id3 tags metadata of songs available on local machine. It is written in Python and leverages Elasticsearch for its functionality.

Purpose

Presently available music search engines don't search on song metadata like lyrics or song year. There are systems which search on music available online but we havn't found any system which do this on local machine.

System Description

Requirements

- Operating System: Windows or Linux
- Working Internet connection

- Appropriate RAM (>=4GB is recommended) to handle Elasticsearch server.

Input

- Path to directory which contains songs
- Search key

Output

- Search results with individual scores
- Highest score
- Total time taken to search
- Total Number of results

My Contribution

My major contribution is in User interface design and query engine design. Apart from this I was involved in overall idea conceptualization and system design. I wrote functions and designed Jinja2 tamplates to render the search results using Flask.

- Flask comes with Jinja2 templating engine by default.
- Jinja2 templating engine lets you create dynamic web page layouts which can be populated with inputs from a flask server.
- Google's Material Design Lite is used for styling.
- Each result is rendered into a card component of MDL and each card may contain a button component.
- WTForms are used to manage form data.

The UI consists of `layout.html`, `main.html` and `index.html`.

- 'layout.html' contains the base template which is common for two other pages.
- 'main.html' is the first page which is rendered when URL is loaded when no search is performed.
- 'index.html' contains two sections.
    - first half contains the query form similar to that in `main.html`, a `<div>` which contains statistics including maximum score, total results, time taken and button which gives option for downloading results as playlist.
    - second half contains search results sorted score in descending order.

I have also used `media queries` in styling to make the view adaptable to screen size.

Filters are shown in form of checkbox input. When user submits the query, these buttons are converted into a dictionary of boolean values. As showm below

```
def makeFiltersList(form):

    # filterDict contains the values for filters used
    # {title:"xyz", "author": "xyz", lyrics:"xyz"}
    filterDict = {}
    filterDict['title'] = 0
    filterDict['artist'] = 0
    filterDict['lyrics'] = 0
    if form.title.data:
        filterDict['title'] = 1
    if form.author.data:
        filterDict['artist'] = 1
    if form.lyrics.data:
        filterDict['lyrics'] = 1
    return filterDict
```

The dictionary is passed to `indexHandle` function of ESPAICall module along with index name and search key which converts returns a dictionary containing search results. These search results are rendered using MDL and Jinja2 Template.

```
{% if results %}
    <!-- check if given item is found or not -->
    {% if results['hits']['total'] == 0 %}
        <h3 style="color:grey;"> No results found! </h3>
    {% endif %}
    <!--else render results-->
    {% if results['hits']['total'] > 0 %}



        <!--iterate through dictionary to render each result
        {% for i in results['hits']['hits'] %}
            <div class="mdl-card mdl-shadow--2dp" style="widt
                <div class="mdl-card__title">
                    <h2 class="mdl-card__title-text" style="r
                </div>
                <div class="mdl-card__supporting-text" style=
                    <!--<b style="float: right; color: grey;'
                    <b>Album:</b> {{ i['_source']['album'] }}
                    <b>Artist:</b> {{ i['_source']['artist']
                    <b>Year:</b> {{ i['_source']['year'] }}<b
                    {% if  i['_source']['lyrics'] != '' %}
                        <div id="lyricsFull{{ i['_id'] }}" st
                            <pre>{{ i['_source']['lyrics'] }}
                        </div>
                    {% endif %}

                </div>
                <!--play song button. need to link it to musi
                {% if  i['_source']['lyrics'] != '' %}
                <div class="mdl-card__actions mdl-card--borde
                    <!--<a class="mdl-button mdl-button--colo
```

```
                    Play
                    </a>-->
                    <a class="mdl-button mdl-button--colored
                    Lyrics
                    </a>
                </div>
                {% endif %}
                <div class="mdl-card__menu" style="color: gre
                    score: {{ i['_score'] }}
                </div>
            </div><br>
        {% endfor %}
    {% endif %}
{% endif %}
```

It iterates through the dictionary and creates mdl-cards on the fly.

You can find more details about my contributions at our https://github.com/electron0zero/MusicSurf

Other work which was not implemented

Initially, I wrote modules for implementing Elasticsearch using dedicated python library but later on we found better ways to implement that and so that was discarded.

Contribution Graph

Contributions can be verified on GitHub and can be seen in Contribution graph