

VerseVault
Your Key to Lyrics, Your Door to Music



(PRI - M2)

André Lima - 202008169
Guilherme Almeida - 202006137
Jorge Sousa - 202006140
José Castro - 202006963

Group 75



Introduction - **What is it?** (Recap)

- **Lyric-based** Search Engine for **finding songs**
- Aggregates **various metadata** regarding musical content
- Aims to:
 - Allow **text-based querying** with the aid of context-sensitive **filters**
 - Observe a specific song's **composition** (both **lyrically** and **structurally**)



Introduction - **Why do it?** (Recap)

Lyrical and structural musical
data is **vast but mostly
unexplored**

**Centralized and rich data
sources** for our data
collection

In its final stage, it will be a **fun and
interactive way** to interact and **perceive
music production and its history**



Domain Model (Recap)

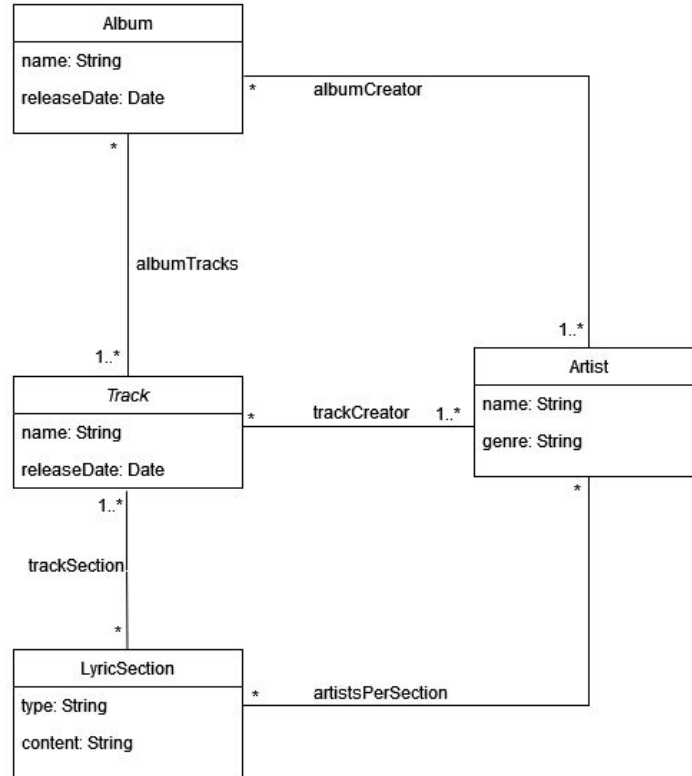
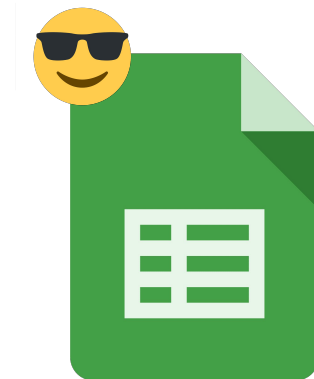


Fig 1: Verse Vault's Domain Model



Tooling



Group 75



Search Scenarios

We have **four search scenarios**:

- I want to find a track talking about a **specific subject** inside a **particular structural section**.
- I want to find a song talking about a **given topic**.
- I want to find a song of a **given genre** that talks about a **specific topic**.
- I want to find a track that talks about a **given topic** even if the **spelling is wrong**.

Indexing - Indexed Fields

Field Name	Type	Indexed
_id	string	false
name	basic_text_t	true
duration	plong	true
url	string	false
artist	artist_t	true
publishedAt	pdate	true
genres	basic_text_t	true
lyrics.title	lyrics_title_t	true
lyrics.content	text_general	false
lyrics.content_en	content_en_t	true
lyrics.content_es	content_es_t	true
lyrics.content_phonetic_en	content_phonetic_en_t	true
lyrics.content_phonetic_es	content_phonetic_es_t	true
album.name	text_general	true
album.image	string	false
entities.text	basic_text_t	true
entities.start	plong	false
entities.end	plong	false
entities.type	string	false

Table 1: Fields defined in the refined schema

Indexing - Indexed Analyzers

- Tokenizers:
 - Standard - track name, lyrics section content, album name, etc.
 - Letter - artist (e.g. "L.M.F.A.O." matches with "L M F A O")
 - Whitespace - lyrics section title (e.g. "Pre-Chorus" -> "PreChorus")
- Filters:
 - **Lowercase**
 - **Pattern Replace**
 - **ASCII Folding** - useful to convert UTF-8 characters to ASCII
 - **Keyword Repeat** - useful to preserve non-stemmed data
 - **Porter Stem** - english language stemmer
 - **Spanish Light Stem** - spanish language stemmer
 - **Remove Duplicates**
 - **Beider-Morse Phonetic Matching** - phonetic token matching



Information Need 1

Search scenario: I want to find a track talking about something inside a particular structural section.

Information Need: Find songs that talk about playing the guitar in the chorus.

Query:

```
{!parent which="doc_type:track" filters=
$childfq score=max}{!edismax qf="content_en"}
("play guitar"~5)
```

request-handler (qt)

common

q

q op

fq

sort

score desc

start, rows

f

df

parameter(s)

wt

☒ indent on

☐ debugQuery

defType

☐ hl

☐ facet

☐ spatial

☐ spellcheck

Raw Query Parameters

JSON Query

```
{
  "responseHeader": {
    "status": 0,
    "QTime": 1,
    "params": {
      "q": "(parent which=\\\"doc_type:\\\"track\\\" filters=schildfq score=max){!edismax qf=\\\"content_em\\\"jl(\\\"play guitar\\\"-5)}",
      "deType": "lucene",
      "indent": "true",
      "fl": "[*,childe]",
      "q.op": "OR",
      "sort": "\"score desc\"",
      "childfq": "\"title:chorus\"",
      "rows": "20",
      "useParams": "true",
      "..." : "1780165028767"
    }
  },
  "response": {
    "numFound": 16,
    "start": 0,
    "numFoundExact": true,
    "docs": [
      {
        "name": "Captain Crash & The Beauty Queen From Mars",
        "url": "https://www.last.fm/music/Bon+Jovi/_/Captain+Crash+&+The+Beauty+Queen+From+Mars",
        "artist": "Bon Jovi",
        "genres": ["rock", "hard rock"],
        "id": "174481",
        "entities.text": ["Halloween day", "Fourth of July", "Cinderella", "Crash", "Watching", "days", "Playing Superman", "Sid", "Nancy'nFred", "Ginger'n Clyde", "Bonnie'n Liz", "Richard'n Kurt", ...],
        "entities.start": [31, 96, 175, 252, 591, 612, 662, 1140, 1149, 1164, 1181, 1196, 1212, 1233, 1239, 1259],
        "entities.end": [44, 70, 185, 357, 599, 616, 678, 1144, 1159, 1176, 1191, 1208, 1228, 1238, 1250, 1265],
        "album.image": "https://lastfm.freetls.fastly.net/i/u/380x300/1763374c0e272547d5e20eb0ba3866b.png",
        "album.name": "Crush",
        "entities.type": ["DATE", "FAC", "GPE", "ORG", "PERSON"],
        "_version_": "178260897908536256",
        "doc_type": "track",
        "id": "652488ca7390df1ac6bd8fa",
        "lyrics": {
          "title": "Verse 1",
          "content": "Dressed up for a big date like Halloween day but it was Fourth of July now\\nA car crash with a suitcase and a painted face\\nShe was one of a kind\\nShe wears a p...",
          "id": "174482",
          "nest_parent": "174481",
          "_version_": "178260897908536256",
          "doc_type": "lyric_section"
        }
      }
    ]
  }
}
```

Fig 2: Information Need 1 - Query result (improved query)



Information Need 1 - **Evaluation**



Fig 3: Information Need 1 - Precision@ Values

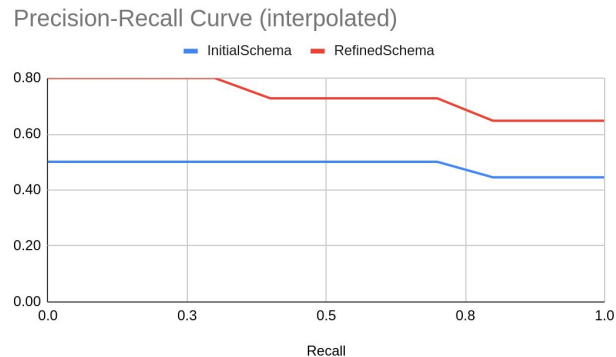


Fig 4: Information Need 1 - P-R Curve

Information Need 2 - Evaluation

Precision@ values

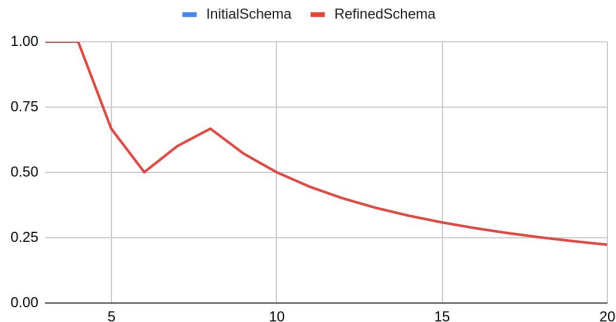


Fig 6: Information Need 2 - Precision@ Values

Precision-Recall Curve (interpolated)

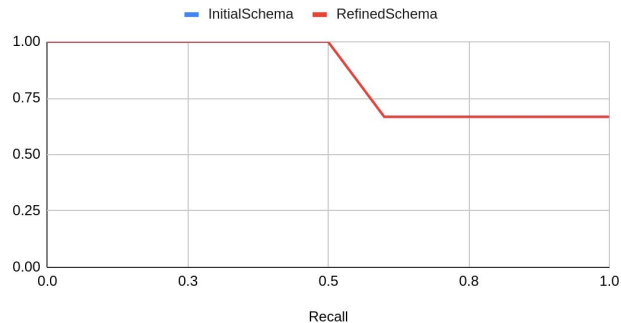


Fig 7: Information Need 2 - P-R Curve

Information Need 3

Search scenario: I want to find a song of a given genre that talks about a specific topic.

Information Need: Find rap songs that talk about immigration.

Query:

```
{!parent which="doc_type:track" score=total}
```

```
{!edismax qf="content_en"}foreigners,immigration}
```

The screenshot shows a search interface with a query input field containing the query: `{!parent which="doc_type:track" score=total}{!edismax qf="content_en"}foreigners,immigration}`. The interface includes various filters and options, such as "q.op" set to "OR", "sort" set to "score desc", and "start rows" set to 0 to 10. The "debugQuery" checkbox is checked. The "defType" dropdown is set to "edismax". The "Raw Query Parameters" section is visible at the bottom.

The results section displays a JSON response for a song titled "Say It". The response includes metadata such as "status", "qtype", "pazases", "score desc", and "useParams". The "response" object contains details about the song, including the artist "Toyz Lanez", the album "I Told You", and the track "Say It". The "doc_type" is "track".

Fig 8: Information Need 3 - Query result (improved query)



Information Need 3 - **Evaluation**

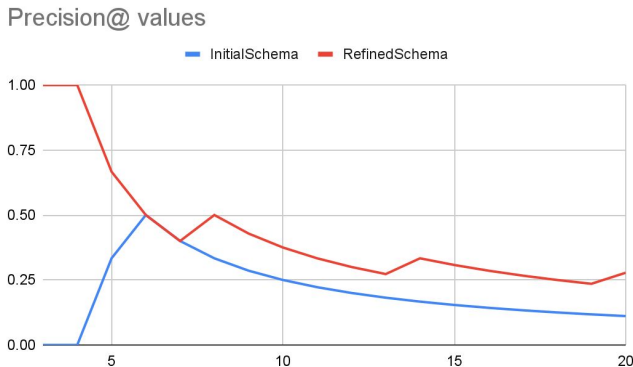


Fig 9: Information Need 3 - Precision@ Values

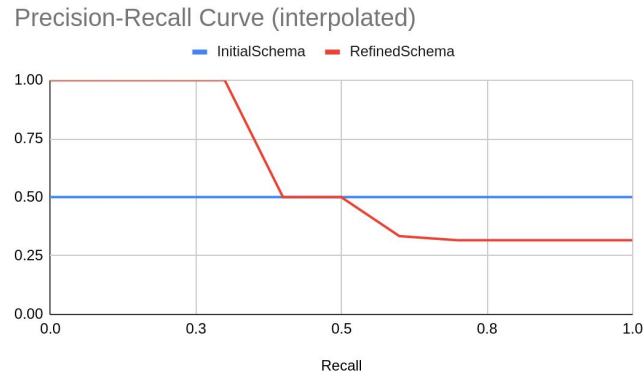


Fig 10: Information Need 3 - P-R Curve

Information Need 4 - **Evaluation**

Precision@ values

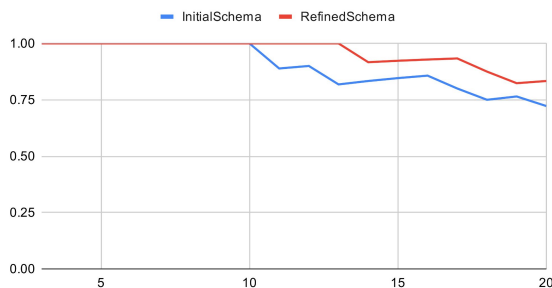


Fig 13: Information Need 4 - Precision@ Values

Precision-Recall Curve (interpolated)

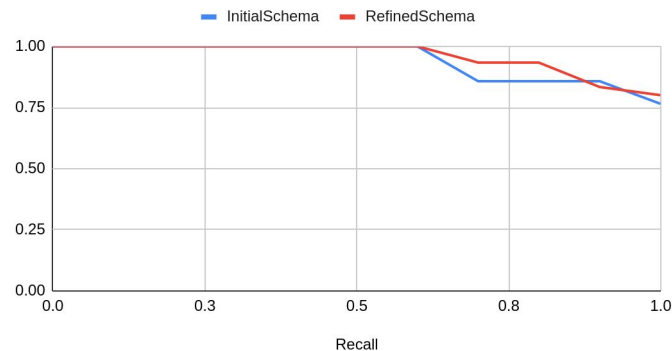


Fig 14: Information Need 4 - P-R Curve

Conclusions and Future Work

- **Search scenarios** were **covered**
- A **well-defined schema** and **query optimization** are necessary
- Handled **Nested Documents**
- There could be **more Information Needs**
- **Query Optimization** could be **more refined**
- **Develop** a **Frontend** for the end-user
- Include fields from multiple nesting levels for the scoring

