

VerseVault
Your Key to Lyrics, Your Door to Music



(PRI - M3)

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**
 - 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**



Query Structure

Assuming 'query' is the user search input, the queries are structured as following:

```
{!parent which="doc_type:track" score=total}  
{!edismax qf="title^0.5  
content^4 content_phonetic^2" pf="content^10"}  
(query~3)
```

This way, we apply a field boosting in the title, content and content_phonetic fields, a phrase boost in the content and a slope of 3 in the query input.



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.

User Input: play guitar

Query:

```
{!parent which="doc_type:track" score=total}
{!edismax qf="title^0.5
content^4 content_phonetic^2" pf="content^10"}
((play guitar)~3)
```

oup 75

request-Handler (qt)

common



q

{parent which="doc_type,track" filters="{childfq score=max}" {edismax q="{content_en"}"}play

q op

OR

fq

sort

score desc

start, rows

fi



*[child]

df

paramset(s)

Select paramset(s)...


wt

☒
 indent on

☐
 debugQuery

defType

lucene 

☐
 hl



☐
 facet


☐
 spatial

☐
 spellcheck

Raw Query Parameters

childfq=0title=chorus

JSON Query 

```
{
  "responseHeader": {
    "status": 0,
    "QTime": 1,
    "params": [
      {
        "q": "{ \"parent which=\"doc_type:track\" filters=chilfig score=max}(elismax ef='content_em')(\"play guitar'=5)\",",
        "defType": "lucene",
        "indent": "true",
        "fz": "[*](chilfig)",
        "q.op": "OR",
        "sort": "\"score desc\"",
        "chilfig": "title:chorus",
        "rows": "20",
        "useParams": true,
        "id": "1780185026767"
      }
    ]
  },
  "response": {
    "numFound": 16,
    "start": 0,
    "highlightExact": false,
    "docs": [
      {
        "name": "Captain Crash & The Beauty Queen From Mars",
        "url": "https://www.last.fm/music/BonJovi/+CaptainCrash&TheBeautyQueenFromMars",
        "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": [91, 96, 175, 252, 291, 612, 662, 114],
        "entities.end": [149, 154, 181, 116, 121, 123, 129, 129],
        "entities.start": [44, 70, 185, 357, 599, 616, 678, 1144, 1159, 1176, 1191, 1260, 1228, 1238, 1258, 1265],
        "album.image": "https://lastfm.freetls.fastly.net/i/u/380x300/1763374c6e27545750280eb3b86b.png",
        "album.name": "Crush",
        "entity.type": ["DATE", "FAC", "GPE", "ORG", "PERSON"],
        "_version_": 17826897980536256,
        "doc_type": "track",
        "id": "652488c7390dfcacbd8fe",
        "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\\nHe wears a pi",
          "id": "174482",
          "nest_parent": "174481",
          "_version_": 17826897980536256,
          "doc_type": "lyric_section"
        }
      }
    ]
  }
}
```

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



Information Need 2

Search scenario: I want to find a song talking about a given topic.

Information Need: Find songs that talk about slavery in Africa.

User Input: slavery africa

Query:

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

```
{!edismax qf="title^0.5
```

```
content^4 content_phonetic^2" pf="content^10"}
```

```
((slavery apartheid africa)~3)
```

The screenshot shows a Solr query interface. On the left, the 'Request-Handler (qt)' is set to '/select'. The 'q' field contains the query: `{!parent which="doc_type:track" score=total}{!edismax qf="content_en" (+!slave apartheid)~3}+africa`. The 'sort' field is set to 'score desc'. The 'start' field is set to '0' and the 'rows' field is set to '10'. The 'wt' field is set to 'wt'. The 'indent on' checkbox is checked. The 'defType' field is set to 'lucene'. The 'Raw Query Parameters' field is empty. On the right, the JSON response is displayed. It shows a 'responseHeader' with 'status': 0, 'QTime': 1, and 'params': {'q': '{!parent which="doc_type:track" score=total}{!edismax qf="content_en" (+!slave apartheid)~3}+africa', 'indent': 'true', 'fl': '*', 'child': 'true', 'q.op': 'AND', 'sort': 'score desc', 'usePz': 'true', 'wt': 'wt', 'indent': '10001001278'}. The 'response' object shows 'numFound': 7, 'start': 0, 'numFoundExact': 'true', and 'docs': [{ 'name': 'The Day the Nigger Took Over', 'duration': 27800, 'url': 'https://www.last.fm/music/Dave/~/The-Day-the-Nigger-Took-Over', 'artist': 'Dave', 'publishedAt': '2014-04-08T00:12:00Z', 'genres': ['hip-hop', 'rap', 'gangsta rap', 'hip hop'], 'id': '131644', 'entities.text': ['Africans', 'the United States', 'South Africa', 'us', 'Africans', 'Break', 'Break', 'den', 'Rodney King', 'Compton', 'den', 'Long Beach', 'L.A.', 'den', 'Nigger'], 'entities.start': [50, 75, 160, 237, 276, 303, 432, 481, 502, 512, 524, 596, 599, 597, 905, 1089, 1091, 1140, 1147, 1190, 1202, 1423, 1673, 1699, 1922, 1993, 1999, 2117, 2332, 2325, 2346, 2349, 2352, 2444], 'entities.end': [60, 92, 173, 259, 284, 308, 437, 486, 750, 863, 915, 927, 946, 963, 974, 1081, 1097, 1146, 1156, 1201, 1264, 1427, 1672, 1704, 1926, 1935, 1998, 2130, 2143, 2228, 2245, 2348, 2355, 2447], 'album image': 'https://lastfm.freetls.fastly.net/i/u/380x380/64690826f4a09c067a7cffe43992.png', 'album name': 'The Chronic', 'entities.type': ['CARDINAL', 'DATE', 'GPE', 'LOC', 'MORP', 'ORG', 'PERSON', 'PRODUCT', 'TIME'], '_version_': '1762680903394811904', 'doc_type': 'track', '_id': '65488c73980f3c4dc0c98', 'lyrics': [{ 'state': 'Protectus', 'content': 'I na say this and I na and mine if you ain't down for the Africans here in the United States vPeriod, point-blank if you ain't down for the ones that suffered', 'id': '131644', '_next_parent_': '131644', '_version_': '1762680903394811904', 'doc_type': 'track' }] }

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



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.

User Input: foreigners immigration

Query:

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

```
{!edismax qf="title^0.5 content^4
```

```
content_phonetic^2" pf="content^10"} ((foreigners
```

```
immigration)~3)
```

Group 75

The screenshot shows the Request-Handler (qt) interface. On the left, the query is entered as: `{!parent which="doc_type:track" score=total}{!edismax qf="title^0.5 content^4 content_phonetic^2" pf="content^10"} ((foreigners immigration)~3)`. The response is a JSON object representing a search result for a song titled "Say It". The response includes fields like status, qtime, ptime, q, ptime, score, and doc. The doc object contains metadata about the song, including the artist (Tory Lanez), album (Tory Lanez), and track title (Say It).

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



Semantic Search vs Synonym Usage

lyrical content is **disorganized, unstructured, unconventional** and very **misleading...**



...how to stop context from being
lost in translation?

Two possible solutions...
Semantic Search and **Synonym Usage**



Semantic Search vs Synonym Usage

Semantic Search:

- Sentence-transformers
 - *all-MiniLM-L6-v2* Model
- 
- content_vector** containing embeddings (song lyrics)
- Query using *DenseVectorSearch* (KNN)>
- 
- target_vector** containing embeddings (query)

Synonym Usage:

- Solr's Synonym filter (query-time only)



Evaluation - Initial Schema

Precision-Recall Curves (interpolated) for Initial Schema

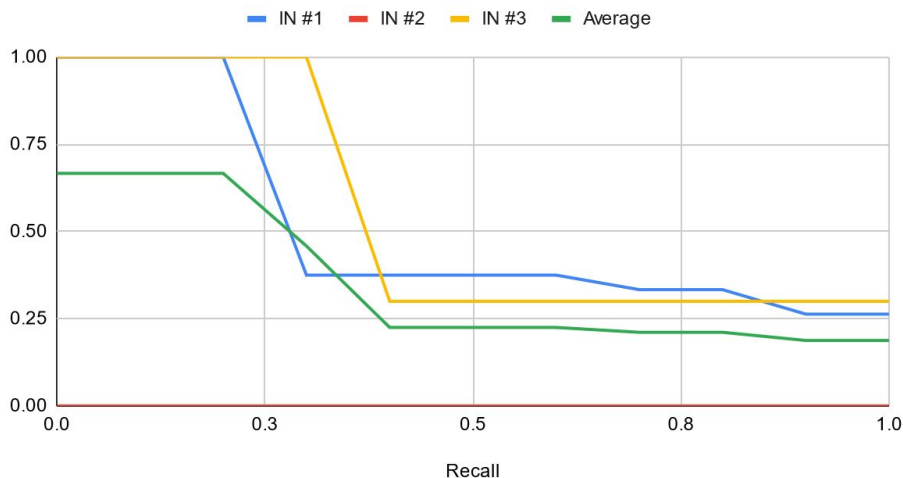


Fig 4: Interpolated P-R Curve for Initial Schema

Precision@ values for Initial Schema

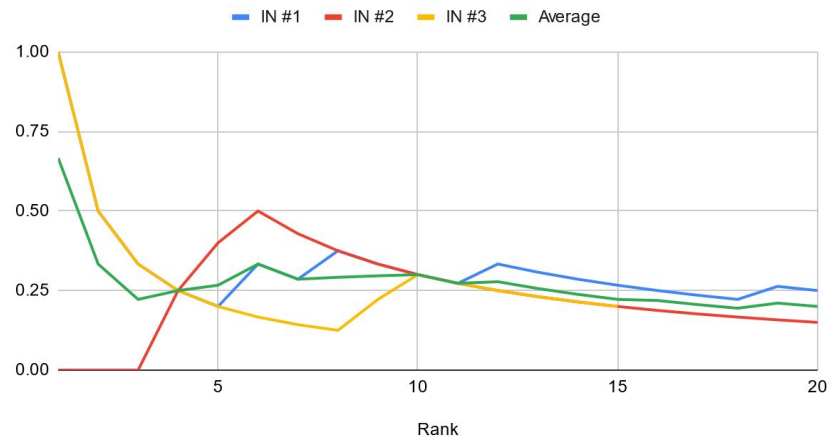


Fig 5: Precision@ values for Initial Schema

Group 75

MAP: 45%



Evaluation - Refined Schema

Precision-Recall Curves (interpolated) for Refined Schema

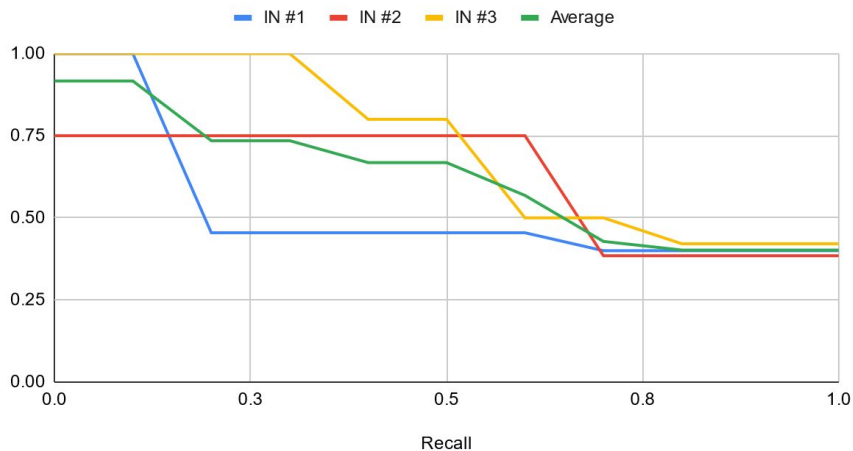


Fig 6: Interpolated P-R Curve for Refined Schema

Precision@ values for Refined Schema

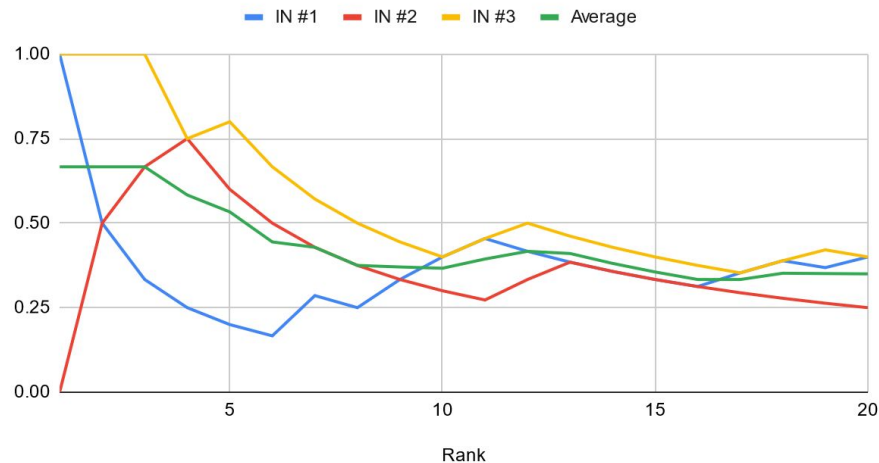


Fig 7: Precision@ values for Refined Schema

Group 75

MAP: 56%



Evaluation - Semantic Schema

Precision-Recall Curves (interpolated) for Semantic Schema

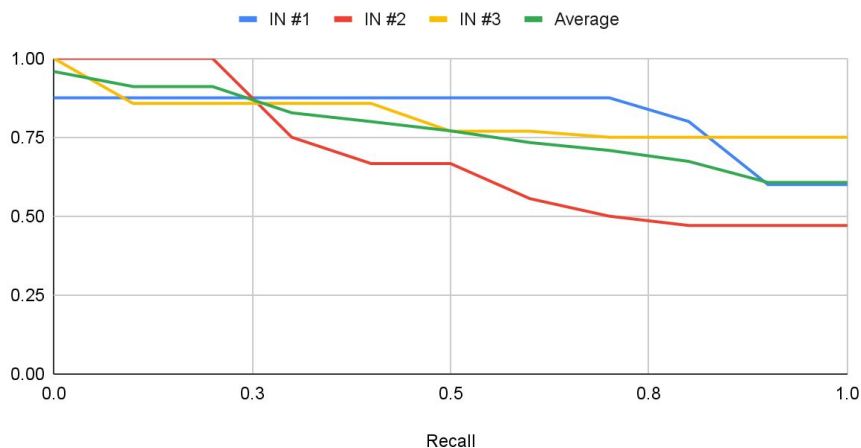


Fig 8: Interpolated P-R Curve for Semantic Schema

Precision@ values for Semantic Schema

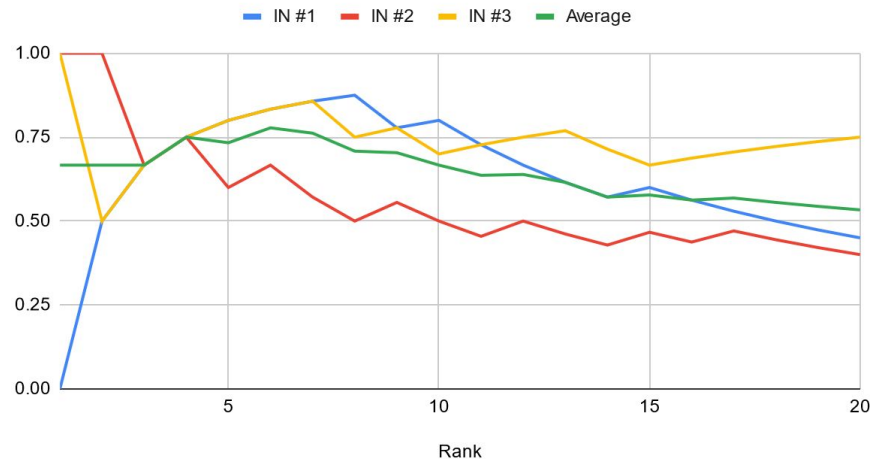


Fig 9: Precision@ values for Semantic Schema

Group 75

MAP: 73%



Evaluation - Synonym Schema

Precision-Recall Curves (interpolated) for Synonym Schema

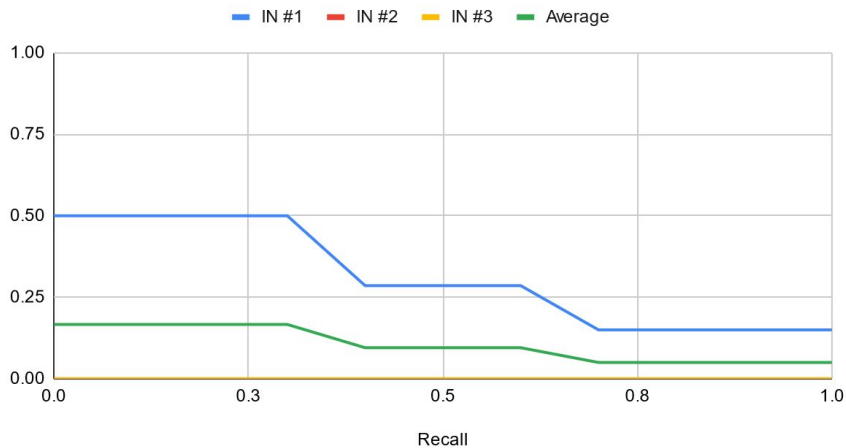


Fig 10: Interpolated P-R Curve for Synonym Schema

Precision@ values for Synonym Schema

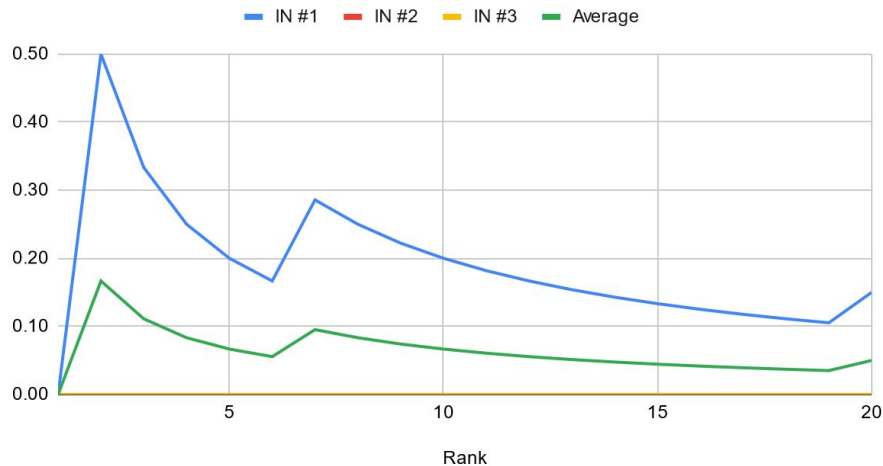


Fig 11: Precision@ values for Synonym Schema

Group 75

MAP: 10.3%



Evaluation - Comparing Schemas

Average Precision-Recall Curves (interpolated)

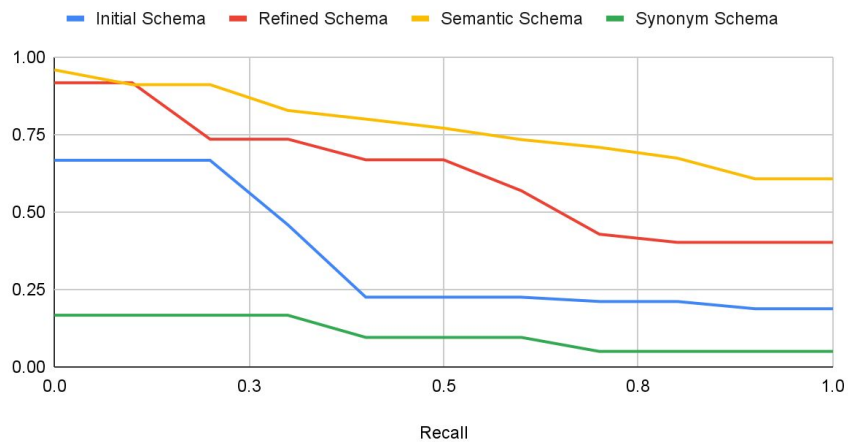


Fig 12: Average Interpolated P-R Curves for all schemas

Average Precision@ values

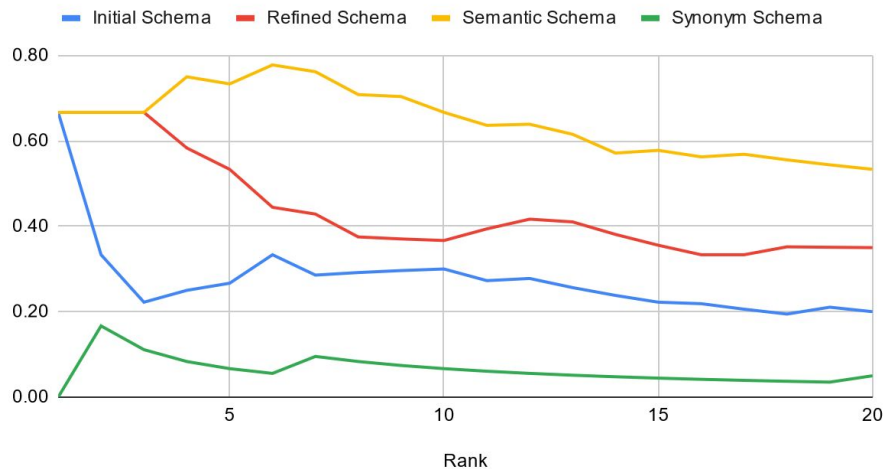


Fig 13: Average Precision@ values for Synonym Schema

Frontend

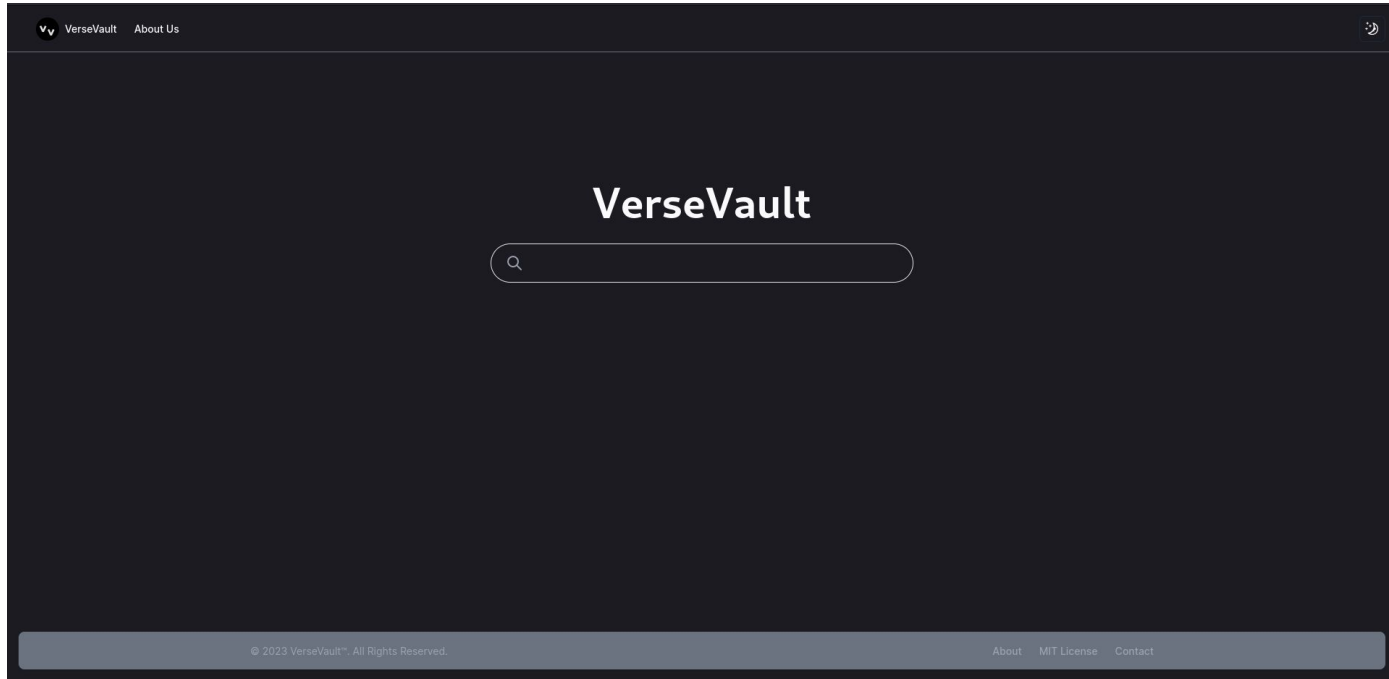


Fig 14: VerseVault's Home Page

Group 75



Frontend

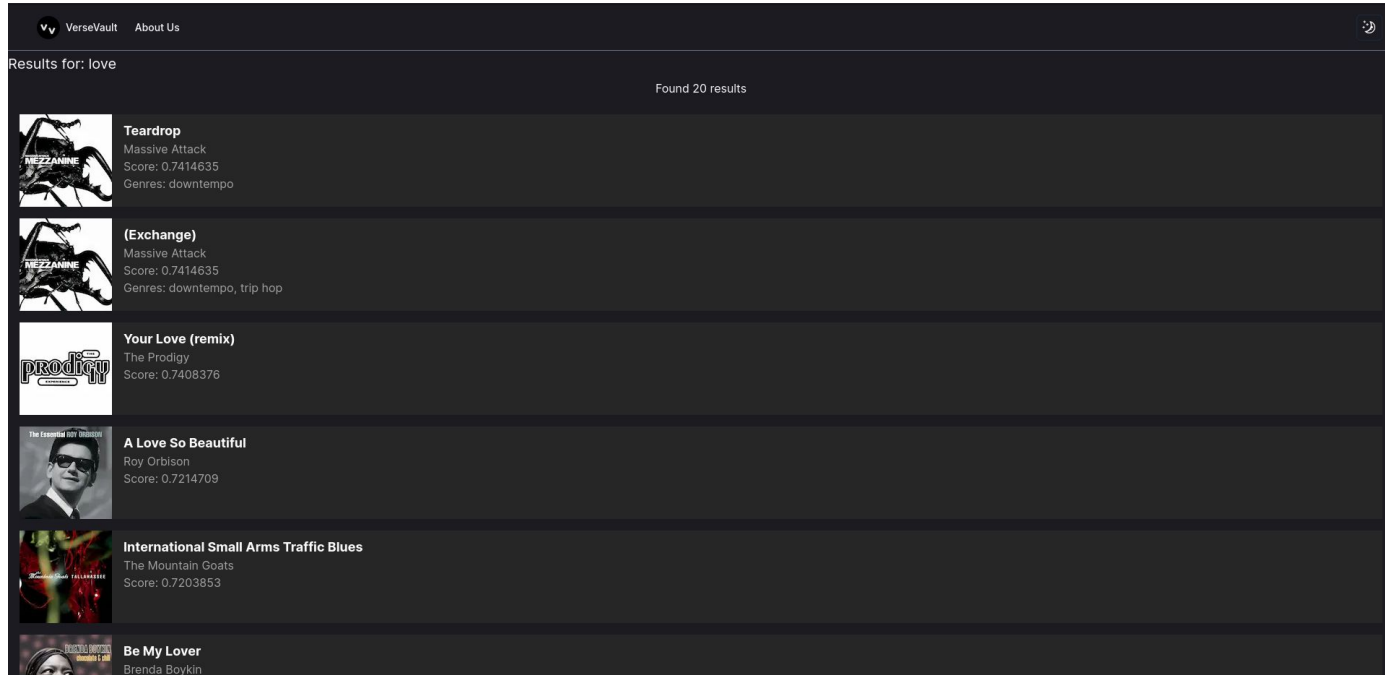


Fig 15: VerseVault's Results Page

Group 75



Frontend

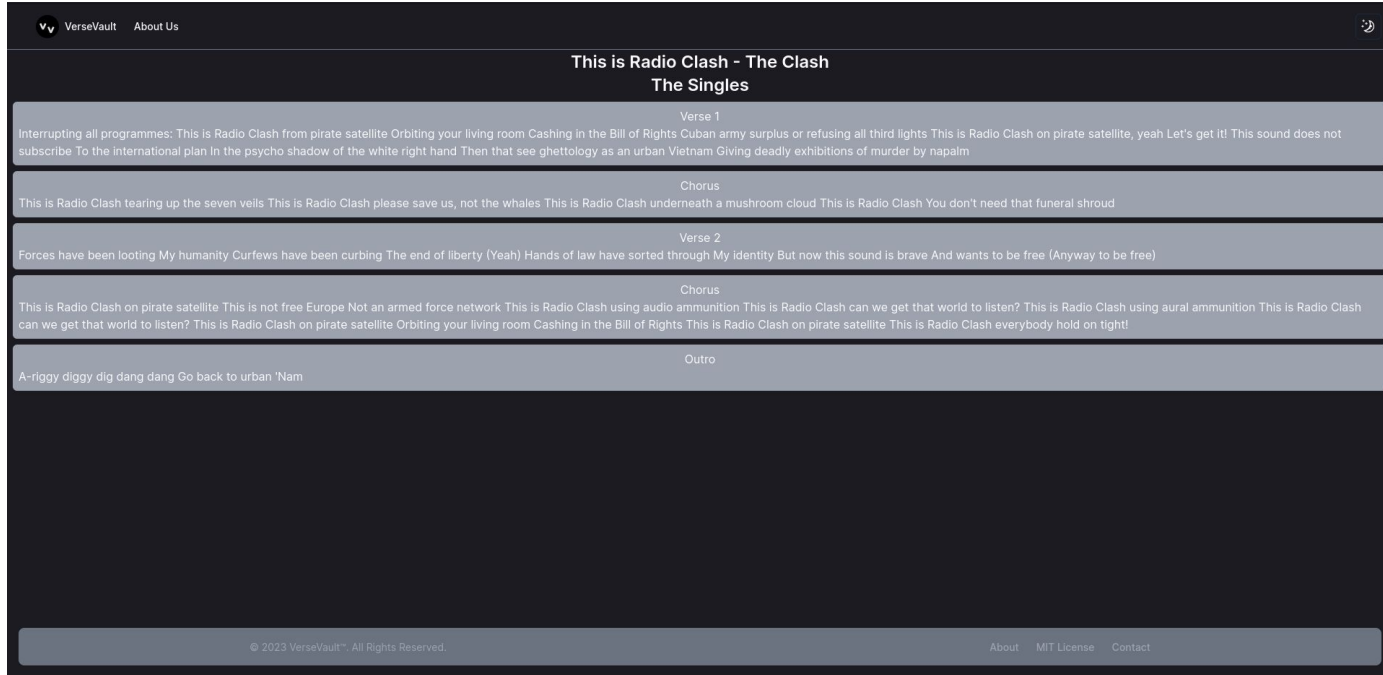


Fig 16: VerseVault's Track Page

Group 75



Frontend

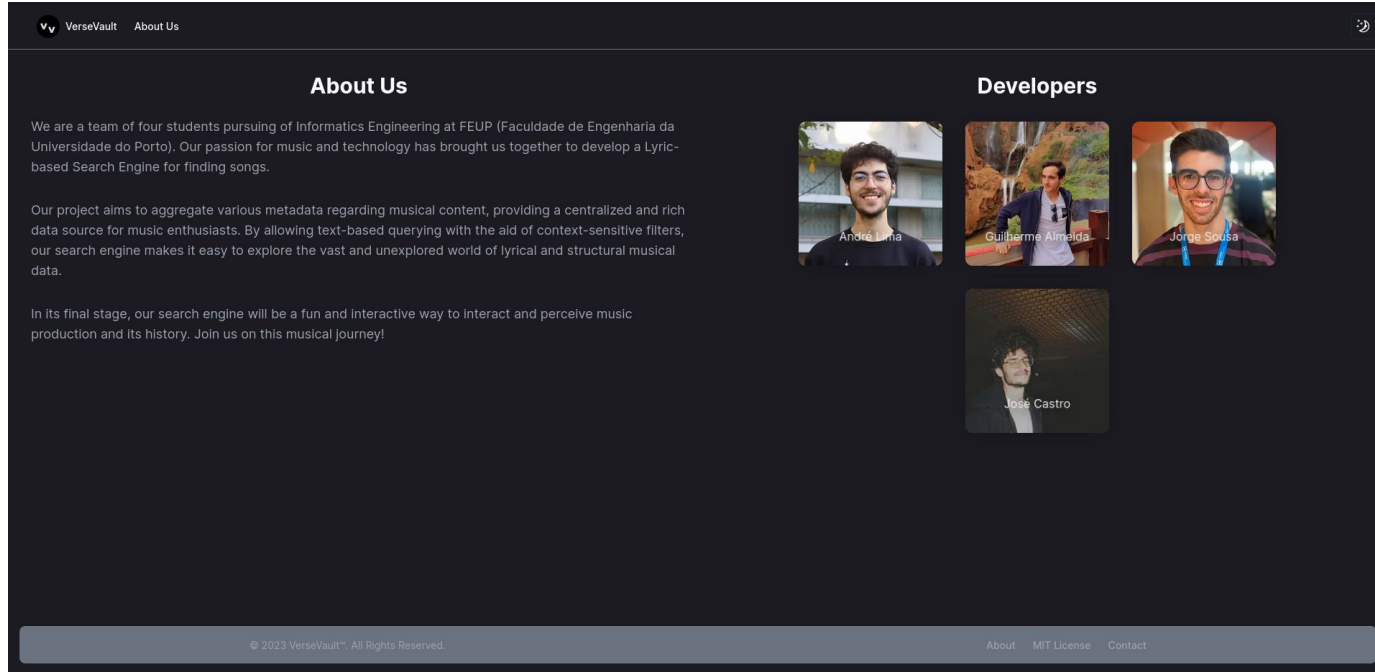


Fig 17: VerseVault's About Page

Conclusions and Future Work

- The refined schema provided better results compared to the initial schema, as expected.
- The use of synonyms revealed some limitations due to the broad matching of terms.
- As a signal, we could prioritize tracks that have more lyrical sections.



Questions

Thank you for your time!

