Deep Web

Bekzhan Omirzak, omirzbek@fit.cvut.cz
Roman Vozivoda, vozivrom@fit.cvut.cz

1. Introduction

The project focuses on **deep web** data retrieval, specifically aiming to simulate a web application where data is fetched through a form-based interface. These web applications, often used in the **deep web**, do not display information directly via search engines and require form submission to retrieve results. In the first part of the project we created a web application that simulates the interface, allowing queries to be sent to the database. The second part is a script that extracts data and reconstructs the original database.

2. Method of Solution

Form Design:

A **form interface** was designed to access the database. We created a web page using Streamlit library where user can choose filters he wants to use for searching

Virtual Tree Structure:

During data retrieval, we progressively fill the form fields and create a **virtual tree** through which we navigate using recursion. Each level of the tree corresponds to a different form field.

Traversing the Tree:

While traversing the tree, at each step, we examine how many results the base application returned. If **k ≥ kmax**, we fill the corresponding form field attribute at that level of the tree and move to the next level. If **k < kmax**, we store the results in the reconstructed database and return to the previous level.

Database Design:

We obtained the **database** of more than 20000 Spotify songs from the kaggle

Simulated Form Access:

The access to the form is **simulated** using a REST API. The interface always returns a limited number of results, defined by the constant **kmax**.

Order of Form Fields:

The order of filling the **form fields** is important. Internal nodes of the tree correspond to elements with a limited number of selection options (checkboxes, radio buttons, dropdown menus). The leaves correspond to text fields where we're trying to enter words from a chosen dictionary.

3.Implementation

Backend:Python(FastAPI)

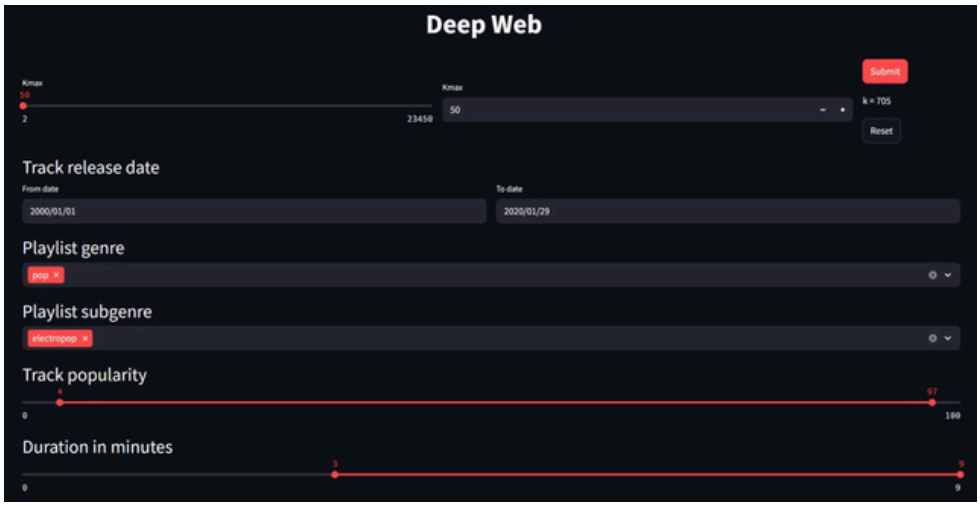
Frontend: JavaScript, Python(Streamlit)

Libraries Used: Pandas, NumPy, Json, Requests

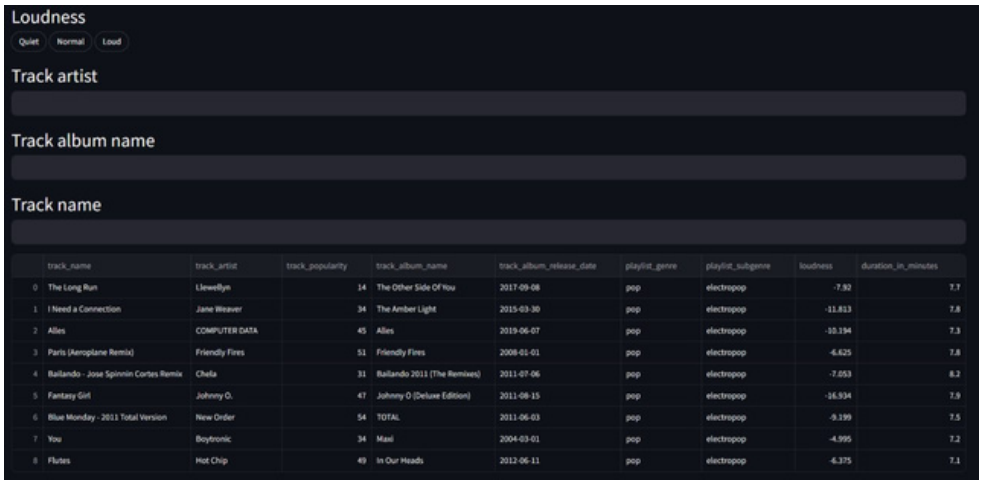
How to run the Application:

pip install -r requirements.txt
python main.py

4.Examples of our web application



A form used for retrieving data from database. It allows users to set any chosen **features**



if **k<k_max**, then we can retrieve the data

5.Experiments

In this section, we present the results of experiments conducted to observe how the order of parameters in the **query tree** and different **k_max** values affect the speed of rebuilding the database.

Default Order of Parameters
The default order of parameters was as follows:

- Track Album Release Date
- Playlist Genre
- Playlist Subgenre
- Track Popularity
- Duration in Minutes
- Loudness
- Track Artist
- Track Album Name
- Track Name

Modified Parameter Order
Modified order of parameters was as follows:

- Track Popularity
- Track Album Release Date
- Playlist Subgenre
- Playlist Genre
- Duration in Minutes
- Loudness
- Track Artist
- Track Album Name
- Track Name

Experiment 1: Default Order	Experiment 2: Modified Order	Experiment 3: Modified Order (with changed intervals)
• k_max = 100: 263 seconds • k_max = 500: 31 seconds • k_max = 1000: 10 seconds • k_max = 2000: 3.3 seconds • k_max = 5000: 2.7 seconds	• k_max = 100: 181 seconds • k_max = 500: 57 seconds • k_max = 1000: 30 seconds • k_max = 2000: 5 seconds • k_max = 5000: 3 seconds	• k_max = 100: 113 seconds • k_max = 500: 44 seconds • k_max = 1000: 22 seconds • k_max = 2000: 14 seconds • k_max = 5000: 2 seconds

Conclusion

Higher k_max values and modified parameter order improve the speed of database rebuilding. We have also tested different intervals for the features like **Track Popularity, Duration in minutes, Album Release Date** and the best results were achieved when we divided intervals by 5 digits for **Track Popularity** and **Duration in minutes** and 2 years for **Album Release Date** .

6. Conclusion

The project successfully demonstrates how to simulate a form-based data retrieval system for the deep web. By using recursive queries and a tree traversal approach, the system effectively reconstructs the database by fetching data iteratively. The results highlight the importance of **k_max** value and optimizing the **querying process** to reconstruct dataset in the most efficient way.

7. References

- Streamlit
- FastApi
- Deep web