PyMeno

Aplikacje, 7

Klaudia Olejniczak, Oleksandr Kuzhel

Opis ogólny tematu :

Aplikacja przeszukująca wybraną przez użytkownika bibliotekę muzyczną i na jej podstawie odnalezienia podobnych utworów, wszystko na podstawie analizy tekstu utworów.

Wymagane dodatkowe biblioteki :

PyLyrics; nltk; mutagen

Opis szczegółowy projektu :

Projekt składa się z elementów zajmujących się tworzeniem biblioteki muzycznej do porównywania posiadanych przez użytkownika utworów, analizy biblioteki użytkownika, wyszukiwania podobieństw oraz GUI.

Tworzenie bazy danych dla porównań.

Biblioteka ta jest zapisywana raz, dla skrócenia czasu przeszukiwania. (Sparsowanie 500 wykonawców zajęło nam łącznie 10h)

Do stworzenia listy autorów użyliśmy udostępnione przez last.fm API, do pobrania albumów i tekstów piosenek biblioteki PyLyrics . Zaś do zapisania ich użyliśmy pickle. Na bibliotekę łącznie składa się 6 plików(pickle200, pickle303, pickle500 zawierają ten sam rodzaj danych, ale z faktu czasu jaką zajmuje ściąganie tekstów rozłożyliśmy to na 3, żeby uniknąć utraty już wcześniej sparsowanych danych) .

Zbiory słów:

PickleLil200, pickleLil303, pickleLil500 – dane [autor,album] = Counter({ word : amount })

PickleLilEvery.p – dane[autor] = Counter({ word : amount })

Zbiory średniej ilości słów na utwór :

PickleLilWordPerSong.p – dane[autor] = średnia

PickleLilFromArtistPerSong.p - dane[autor,album] = średnia

W trakcie rozkładu piosenek na zbiory słów wykorzystana została jeszcze biblioteka ntlk

(Natural Language Tool Kit), dzięki której usunięto wszystko stopwords'y oraz ujednolicono formę słów (np. running → run). Rozkład tekstu powstał na bazie bag_of_words.

Analiza biblioteki użytkownika

Przeszukiwanie biblioteki użytkownika wykonano przy pomocy biblioteki mutagen. Następnie analogicznie jak przy tworzeniu bazy, z różnicą w przechowaniu słownika, w tym wypadku jako zmiennych globalnych.

GUI

Interfejs został stworzony przy pomocy Tkintera

Wyszukiwanie podobnych utworów.

Powstały tutaj dwa podejścia co do wyszukiwania podobnych utworów. Obydwa wykorzystują miarę podobieństwa wektorów, w tym wypadku na podstawie słowników.

Algorytm I

Na podstawie wektorów dla każdego artysty w bazie i wektora wszystkich słów w naszej bibliotece wybieramy najbliższe 1 wyniki, po czym porównujemy je ze względu na średnią ilość słów na utwór. (Naszym zdaniem istnieje tutaj zależność względem tempa utworu, im więcej, tym dynamiczniejsza). Po czym wyszukuje się ponownie za pomocą miary podobieństw wektorów, tylko na bazie artysta,album, z zachowaniem najlepszych wyników z poprzedniej eliminacji.

Algorytm II

Na podstawie słowników z bibliotek i zbioru słów powstaje przecięcie pomiędzy nimi. Wybieramy te zbiory które są największe, ograniczamy poprzez ilość słów(analogicznie jak w alg I) i na wyniku wykorzystujemy podobieństwo wektorów z wykorzystaniem artysta,album.

Wybór przykładowych utworów powstaje za pomocą biblioteki PyLyrics, dzięki której ściągamy wszystkie utwory wybranego albumu danego autora i losujemy któryś z nich.

Linki do wybranych utworów.

Do wybranych utworów zostaną podane url do youtube, gdzie będzie można sobie je przesłuchać.

To Do

Dokończyć alg I (Obecnie domyślnie jest II) , dopisać tworzenie url do utworu na youtube.

How to run

Uruchomić main.py, następnie w oknie File->”Choose folder with music” i poczekać.

(przeszukiwanie i ściąganie trwa chwile)

W Data mamy opcje :

Download_artists_List : zapisuje nam wynik z last.fm do scrobble.xml . Pobiera liste najpopularniejszych 500 utworów.

Parse_information_to_database : tworzy bibliotekę wykonawców

Show : pokazuje nam wszystkich artystów znajdujących się w bibliotece.

Show by album **:** pokazuje nam wszystkich artystów i ich albumy znajdujązych sięw bibliotece