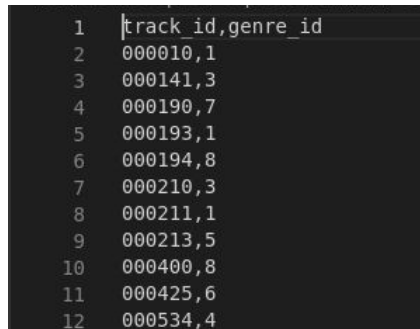


Rendu 1 Projet TSMA

Piet Thomas
Heurtel Thomas

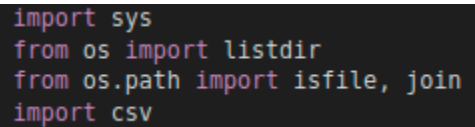
Nous avons commencé par associer des genres aléatoires. Pour ça, nous avons créé un fichier `random_genre.py`, qui copie les lignes du fichier `test.csv` (sauf le header) et qui rajoute à chaque ligne un chiffre random entre 1 et 8, correspondant à un genre. On copie ensuite cette liste dans un nouveau fichier `random.csv`, après avoir ajouté le header. On obtient donc dans le fichier `random.csv`, le format attendu, la première colonne étant les `track_id` et la seconde les `genre_id` générés aléatoirement.



```
1 track_id,genre_id
2 000010,1
3 000141,3
4 000190,7
5 000193,1
6 000194,8
7 000210,3
8 000211,1
9 000213,5
10 000400,8
11 000425,6
12 000534,4
```

Figure 1: Début du fichier `random.csv`

Dans `main.py`, on utilise l'algorithme NN. On aura besoin des bibliothèques suivantes :



```
import sys
from os import listdir
from os.path import isfile, join
import csv
```

Pour exécuter le programme, il faut utiliser la syntaxe suivante : `"python3 main.py <test folder> <train folder> <test csv> <train csv> <out csv>"`

On a écrit plusieurs fonctions :

- `computeScore()` permet de récupérer la différence entre deux mfcc (plus elle est petite, plus c'est intéressant)
- `mfcc.read()` retourne les valeurs d'un fichier mfcc sous la forme d'une liste de flottants
- `eligible_files()` parcourt le répertoire d'entrée et retient tous ses fichiers mfcc avant de les retourner sous forme de liste de strings
- `mfcc_files()` parcourt cette liste et retourne le contenu de chaque fichier mfcc
- `nearest()` prend en entrée un fichier et un répertoire, et retourne le fichier du répertoire le plus proche de celui d'entrée
- `id_from_filename()` retourne l'id d'un fichier selon son nom et son chemin d'accès (fonctionne dans le cas où le nom du fichier est de la forme `"chemin/dacces/id.extension"`)

Puis grâce à la bibliothèque csv, on parcourt les lignes du fichier csv d'entraînement et on stocke chaque id (élément 0 de la ligne) et le genre correspondant (élément 1) dans un dictionnaire afin d'y accéder facilement par la suite.

On ouvre ensuite le fichier de sortie en écriture et on écrit la ligne d'en-tête. On ouvre le fichier csv de test, contenant tous les id nécessaires, et pour chaque id on calcule son plus proche voisin dans le répertoire train grâce à notre fonction nearest(). Important : le nom du fichier doit être "id.wav.mfcc" - on peut cependant changer l'extension en modifiant la variable extension définie plus haut.

Puis on écrit une nouvelle ligne dans le fichier de sortie, avec notre id courant en 0 et en 1 le genre de son plus proche voisin, que l'on pioche dans notre dictionnaire.

Pour utiliser le programme, il faut avoir déjà nos mfcc dans un répertoire. On peut utiliser le code fourni dans main.c pour cela, sachant qu'il ne fonctionne qu'avec des fichiers wav (le fichier mp3_to_wav.py permet de convertir si besoin). Quelques-uns des fichiers mp3 fournis étant corrompus (moins d'une dizaine), le programme C n'a pas pu créer les mfcc correspondants. Dans ces cas là nous avons créé nous même les fichiers et copié dedans les valeurs d'un mfcc au hasard. Les valeurs ne sont donc pas bonnes sur ces quelques mfcc mais cela permet au programme de tourner.

Pour le moment nous avons obtenu le score de 0.27744.