

Mini-Projet : Traitement du son avec Python 2024-2025

ING2 MI2



**Réalise par :**

MILLET Jérémy

CHOMETON Maxence

BATTAIS Nathan

SIGNOUD Nathan

BRETAGNE François

Estelle

**Sommaire**

**1. Choix du sujet et mise en place d’une stratégie de résolution**

Idées de projet :

**Détection et identification d'instruments de musique**

* **Applications** :
  + Transcription musicale automatique (conversion de musique en partitions).
  + Systèmes de recommandation de musique ou d’analyse musicale.
  + Applications éducatives pour apprendre la reconnaissance des instruments.
* **Travail possible** : Créer un programme qui détecte quel instrument est joué dans un enregistrement audio.

**Filtrage et réduction du bruit**

* **Applications** :
  + Casques à réduction de bruit active.
  + Amélioration de la qualité des appels dans les applications de vidéoconférence.
  + Systèmes de sécurité pour détecter des sons spécifiques (bruit de verre brisé, cris d’alarme).
* **Travail possible** : Concevoir un filtre de réduction de bruit en temps réel ou un filtre qui élimine un type spécifique de bruit (par exemple, un bruit de fond constant).

**Traitement du son dans la reconnaissance vocale**

* **Applications** :
  + Assistants vocaux (comme Siri, Alexa, Google Assistant).
  + Systèmes de commande vocale dans les voitures ou les dispositifs médicaux.
  + Transcription automatique d'entretiens ou de cours (pour les personnes malentendantes par exemple).
* **Travail possible** : Créer un modèle de reconnaissance vocale simplifié, ou explorer des méthodes de réduction du bruit pour améliorer la précision.

Sujet retenu : Application de reconnaissance musicale similaire à Shazam

Une d'application dans le domaine musical similaire à Shazam serait un système de reconnaissance musicale.

Application : Système de Reconnaissance Musicale

Objectif :

Développer une application capable d'identifier une chanson à partir d'un court extrait audio. Cela impliquerait de capturer le son ambiant, de l'analyser, et de trouver une correspondance avec une base de données de pistes musicales.

Étapes du Projet :

1. Collecte de données :

- Assemblez une base de données de chansons, en vous assurant d'avoir les fichiers audios numériques et les métadonnées associées (titre, artiste, album).

2. Extraction des caractéristiques audio :

- Utilisez des bibliothèques comme Librosa pour extraire des caractéristiques audios des fichiers, telles que :

- Spectrogrammes

- Fingerprints audio (empreintes digitales audio)

- Descripteurs de fréquence

3. Création de la base de données de référence :

- Stockez les caractéristiques extraites dans une base de données optimisée pour la recherche rapide et efficace.

4. Capture et traitement de l'audio en temps réel :

- Implementez une fonctionnalité pour capturer l'audio ambiant via le microphone.

- Traitez cette capture pour extraire ses caractéristiques.

5. Algorithme de correspondance :

- Confrontez les caractéristiques de l'extrait capturé avec celles de la base de données en utilisant des techniques de matching, telles que l'algorithme de recherche par nearest neighbors.

6. Interface utilisateur :

- Développez une interface où l'utilisateur peut activer la reconnaissance et voir les résultats (c'est-à-dire, le titre de la chanson, l'artiste, etc.).

7. Validation et optimisation :

- Testez l'application avec différents extraits dans divers environnements sonores.

- Optimisez le système pour améliorer la précision et réduire le temps de réponse.

Technologies et outils :

- Python avec des bibliothèques comme Librosa, NumPy, SciPy pour le traitement audio.

- Interface : Utiliser des outils pour construire l'interface utilisateur comme Tkinter en Python pour les applications de bureau.

**2. Choix de la base de donnée :**

GTZAN Genre Collection

Description : Ce dataset contient 1 000 morceaux de musique répartis en 10 genres différents (par exemple : rock, jazz, classique, etc.). Bien qu’il soit relativement petit, il est utilisé fréquemment pour des projets de reconnaissance musicale et d’apprentissage automatique.

Compatibilité avec Python : On peut utiliser des bibliothèques comme librosa ou pyAudioAnalysis pour extraire des caractéristiques audio de ce dataset.

C’est une bonne option pour un premier projet de reconnaissance de musique avec des genres prédéfinis.

Librosa et PyAudioAnalysis

Description : Ces deux bibliothèques Python permettent de travailler directement avec des fichiers audio, d'extraire des caractéristiques musicales et d’effectuer une analyse audio. Tu peux les utiliser avec n'importe quel jeu de données.

Compatibilité avec Python : librosa et pyAudioAnalysis sont entièrement compatibles avec Python et offrent une multitude de fonctionnalités pour extraire des descripteurs audio et des caractéristiques de signal nécessaires pour la reconnaissance musicale.

Extraction de caractéristiques audio : Utilise librosa pour extraire des caractéristiques telles que :

Spectrogrammes

MFCC

Chroma features (utile pour la reconnaissance de tonalité)

Zero crossing rate, etc.

Méthodes de comparaison : Utilise des techniques comme la recherche de similarité avec des descripteurs musicaux ou des approches basées sur des modèles d'apprentissage automatique comme les réseaux neuronaux ou les machines à vecteurs de support (SVM) pour comparer l'extrait de musique avec les chansons référencées.

Gestion des fichiers audio : Pour gérer des fichiers audio localement, on les a téléchargés depuis des plateformes comme Freesound ou GTZAN, puis on les a analysés avec librosa ou pyAudioAnalysis.

**3. Bibliothèque retenu et implémentation de notre BDD et nos titre musicaux.**

Pour réaliser un projet de reconnaissance musicale en Python, nous avons eubesoin de plusieurs bibliothèques pour manipuler les données audio, extraire des caractéristiques, effectuer des comparaisons et gérer les métadonnées. Voici une liste des bibliothèques essentielles que nous avons utilisées pour notre projet, avec des explications sur leur rôle :

**1. Librosa**

Utilisation : Librosa est une bibliothèque fondamentale pour l'analyse audio. Elle permet d'extraire des caractéristiques de signaux audio comme le spectrogramme, les MFCC (Mel-frequency cepstral coefficients), les chroma features, le pitch, etc.

Installation :

pip install librosa

Exemples d'utilisation :

Extraction de MFCC : librosa.feature.mfcc()

Calcul du spectrogramme : librosa.feature.melspectrogram()

**2. PyDub**

Utilisation : PyDub est une bibliothèque pour la manipulation de fichiers audio, notamment pour convertir des formats audio (par exemple WAV vers MP3), découper des fichiers ou changer leur fréquence d'échantillonnage.

Installation :

pip install pydub

**Exemple d'utilisation :**

Conversion de formats : AudioSegment.from\_file("song.wav")

Exportation : audio.export("output.mp3", format="mp3")

**3. NumPy**

Utilisation : NumPy est essentiel pour la manipulation de matrices et de vecteurs en Python. Les caractéristiques audio extraites avec Librosa sont souvent sous forme de matrices que tu manipuleras avec NumPy.

Installation :

pip install numpy

Exemple d'utilisation :

Traitement des vecteurs audio : numpy.array()

**4. SciPy**

Utilisation : SciPy fournit des fonctions pour les calculs scientifiques, comme le filtrage de signaux ou la conversion de Fourier. Elle est utile pour effectuer des transformations de signaux et pour le prétraitement des données.

Installation :

pip install scipy

Exemple d'utilisation :

Transformation de Fourier : scipy.fftpack.fft()

Filtrage de signal : scipy.signal.filtfilt()

**Matplotlib et Seaborn (pour visualiser les données)**

Utilisation : Ces bibliothèques permettront de visualiser des graphiques, comme des spectrogrammes, des courbes de MFCC, ou d’autres visualisations utiles pour l’analyse des données audio.

Installation :

pip install matplotlib seaborn

Exemple d'utilisation :

Visualisation d'un spectrogramme : plt.imshow(spec, cmap='inferno')

Visualisation de l’évolution du pitch : seaborn.lineplot()

**4. Exctraction des audios et affichage du spectre**

**Voir code en Annexe :**

Ce code est écrit en Python et utilise la bibliothèque Librosa pour extraire des empreintes audio et afficher des spectrogrammes à partir de fichiers audio. Voici un résumé des principales fonctionnalités et étapes du code :

1. Extraction de l'empreinte audio :

- La fonction `extract\_empreinte` prend un fichier audio comme entrée et utilise la transformée de Fourier à courte durée (STFT) pour analyser le signal.

- Elle charge le fichier audio et le convertit en un signal numérique, puis applique la STFT pour obtenir un spectrogramme représentant les amplitudes des fréquences au fil du temps.

- L'empreinte audio est ensuite calculée en prenant la moyenne des amplitudes pour chaque fréquence, produisant ainsi un vecteur qui résume l'information audio.

2. Exemples d'utilisation :

- Le code extrait d'abord l'empreinte audio d'un exemple de trompette fourni par Librosa.

- Ensuite, il tente d'extraire les empreintes audio de deux fichiers de musique de 'The Weeknd', nommés "The Weeknd.mp3" et "The Weeknd2.mp3". Les empreintes résultantes sont affichées dans la console.

3. Affichage du spectrogramme :

- La fonction `afficher\_spectrogramme` charge également un fichier audio et utilise la STFT pour créer un spectrogramme.

- Les amplitudes sont converties en décibels pour une meilleure visualisation, et le spectrogramme est affiché à l'aide de Matplotlib, permettant d'analyser visuellement l'évolution des fréquences dans le temps.

Ce code est précieux pour l'analyse et la comparaison des empreintes sonores d'enregistrements audio, utile dans des applications telles que la reconnaissance de musique ou l'analyse de la qualité du son. Les spectrogrammes offrent également une visualisation intuitive des caractéristiques sonores du contenu audio.

**5. Comparaison des pistes audios et exemples**

**Voir code en Annexe :**

Ce code en Python utilise la bibliothèque Librosa pour comparer des empreintes audio et rechercher des correspondances dans une base de données. Voici un résumé de ses principales fonctionnalités et opérations :

1. Comparaison des empreintes audio :

- La fonction `comparer\_empreintesOneVSOne` compare deux empreintes audio et renvoie `True` si elles sont similaires (c'est-à-dire si la distance cosinus entre elles est inférieure à une marge d'erreur définie). Cela implique :

- L'ajustement de la longueur des deux empreintes pour s'assurer qu'elles sont de taille identique.

- Le calcul de la distance cosinus, qui mesure la similarité entre deux vecteurs (ici, les empreintes audio).

2. Recherche dans une base de données :

- La fonction `comparer\_empreintesData` cherche à comparer une empreinte audio donnée contre tous les fichiers audio (.wav et .mp3) d'un dossier spécifié et de ses sous-dossiers. Les étapes incluent :

- Parcourir tous les fichiers audio du dossier choisi.

- Extraire l'empreinte de chaque fichier audio.

- Comparer l'empreinte choisie avec chaque empreinte trouvée dans la base de données en utilisant la distance cosinus.

- Imprimer le nom de tout fichier avec lequel une correspondance a été trouvée et retourner son chemin complet.

3. Visualisation des spectrogrammes :

- Des exemples d'utilisation de la fonction `afficher\_spectrogramme` sont fournis pour visualiser les spectrogrammes de plusieurs fichiers audio, notamment "The Weeknd.mp3", "The Weeknd2.mp3", et un exemple de trompette. Cela aide à observer comment les différentes fréquences évoluent dans le temps.

4. Cas d'utilisation :

- Le code montre comment extraire l'empreinte d'un fichier audio et rechercher des musiques similaires dans une base de données de genres musicaux, notamment en cherchant des correspondances pour des fichiers spécifiques de "The Weeknd" et d'autres genres comme le blues.

En résumé, notre code permet de réaliser analyse comparative des empreintes audio, facilitant des applications comme la reconnaissance musicale et l'identification de fichiers audio similaires dans une vaste collection de données. Les visualisations des spectrogrammes ajoutent une dimension informative pour comprendre les caractéristiques des sons observés.