

# Rapport Prediction de Langues



Fait par : ABDALLAOUI Mohamed  
encadrer par :  
PR KHARROUBI Jamal

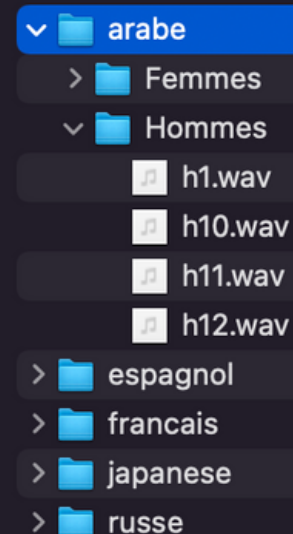
Avant de se lancer dans le projet il faut bien avant faire un data cleaning pour avoir une base de données plus lisible et facile à manipuler

On va commencer par créer un fichier .txt qui va ordonner notre data tel que dans chaque langue on aura deux autres fichiers Femmes et Homme et dans chacun de ces fichiers est inclus un fichier wav que l'on va ordonner ( de H1 jusqu'à Hn ) pour homme et de ( F1 jusqu'à Fn ) pour Femmes

le code ci-dessous nous permettra de faire cela après l'avoir exécuté sur le terminal de votre PC

```
1 #!/bin/bash
2
3 lang_dir="/Users/mohamedabdallaoui/Desktop/langue"
4 prefix=""
5
6 echo "Enter common prefix for files:"
7 read prefix
8
9 for lang in $(ls "$lang_dir"); do
10     if [[ -d "$lang_dir/$lang/hommes" ]]; then
11         i=1
12         for file in $(ls -1 "$lang_dir/$lang/hommes"/*.wav); do
13             new_name="$prefix"h$i.wav
14             mv "$file" "$lang_dir/$lang/hommes/$new_name"
15             i=$((i+1))
16         done
17     fi
18
19     if [[ -d "$lang_dir/$lang/femmes" ]]; then
20         i=1
21         for file in $(ls -1 "$lang_dir/$lang/femmes"/*.wav); do
22             new_name="$prefix"f$i.wav
23             mv "$file" "$lang_dir/$lang/femmes/$new_name"
24             i=$((i+1))
25         done
26     fi
27 done
28
29 # Clear the prefix variable
30 prefix=""
```

## Résultats



- arabe
  - Femmes
  - Hommes
    - h1.wav
    - h10.wav
    - h11.wav
    - h12.wav
- espanol
- français
- japanese
- russe

## TP1 reconnaissances de Langues

Pour prédire la langue française ou n'importe quel autre langues respectivement en utilisant le modèle GMM entraîné, nous devons suivre les étapes suivantes:

1. Charger les fichiers audio en français pour lesquels vous souhaitez prédire la langue.
2. Extraire les MFCC de chaque fichier audio et les mettre dans une liste.
3. Effectuer un padding des MFCC de chaque fichier audio pour les mettre à la même longueur que les MFCC de la base de données d'entraînement.
4. Concatener les MFCC de tous les fichiers audio en une seule matrice.

### Extract the Mfcc features and remove noise

```
from sklearn.model_selection import train_test_split

path = "/Users/mohamedabdallaoui/Desktop/Langue"
language = "russe"
genders = ["Femmes", "Hommes"]

mfcc_folder_path = "/Users/mohamedabdallaoui/Desktop/Russe_MFCC"
mfcc_folder_path_femmes = os.path.join(mfcc_folder_path, "Femmes")
mfcc_folder_path_hommes = os.path.join(mfcc_folder_path, "Hommes")

mfcc_features_list = [] # To store MFCC features for all audio files

#This will loop through all the audio files in each gender, and load each audio file us
for gender in genders:
    audio_files_path = os.path.join(path, language, gender)
    audio_files = os.listdir(audio_files_path)
    for audio_file in audio_files:
        audio_file_path = os.path.join(audio_files_path, audio_file)
        audio_signal, sample_rate = librosa.load(audio_file_path)

        #remove silence
        audio_signal = effects.trim(audio_signal, top_db=40)[0]

        # Extract MFCC features
        mfcc_features = librosa.feature.mfcc(y=audio_signal, sr=sample_rate)
        mfcc_features_list.append(mfcc_features)

    # Save MFCC features in a new file in desktop
    if gender == "Hommes":
        for i in range(1, len(audio_files)+1):
            if f"h{i}.wav" in audio_files:
                mfcc_file_path = os.path.join(mfcc_folder_path_hommes, f"{language}")
                np.save(mfcc_file_path, mfcc_features)
                break
    elif gender == "Femmes":
        for i in range(1, len(audio_files)+1):
            if f"f{i}.wav" in audio_files:
                mfcc_file_path = os.path.join(mfcc_folder_path_femmes, f"{language}")
                np.save(mfcc_file_path, mfcc_features)
                break

# Pad MFCC features with zeros to make them of equal shape
# Maximum length of MFCC features
max_length = np.max([mfcc_features.shape[1] for mfcc_features in mfcc_features_list])
padded_features_list = []
for mfcc_features in mfcc_features_list:
    padded_features = librosa.util.pad_center(mfcc_features, size=max_length, axis=1)
    padded_features_list.append(padded_features)
```

### Split the data into train and test and save it in their respective files

```
1: import os
import numpy as np
from sklearn.model_selection import train_test_split

# Load all the MFCC files for a language and gender
language = "russe"
genders = ["Femmes", "Hommes"]
mfcc_folder_path = "/Users/mohamedabdallaoui/Desktop/Russe_MFCC"

mfcc_files = []
for gender in genders:
    audio_files_path = os.path.join(mfcc_folder_path, gender)
    audio_files = os.listdir(audio_files_path)
    for audio_file in audio_files:
        if audio_file.endswith(".npy"):
            mfcc_file_path = os.path.join(audio_files_path, audio_file)
            mfcc_files.append(mfcc_file_path)

# Split the MFCC files into train and test sets
train_files, test_files = train_test_split(mfcc_files, test_size=0.2, random_state=42)
```

On divise notre data en Train et Test set on utilisant la fonction `train_test_split` de la bibliothèque `sklearn.model_selection`.

On défini test a 20% et training a 80%.

Le paramètre `random` est a 42 pour assurer la variabilité des resultat a chaque exécution.

```
# Create the train and test directories if they do not exist
train_dir = "/Users/mohamedabdallaoui/Desktop/train_russe"
test_dir = "/Users/mohamedabdallaoui/Desktop/test_russe"

if not os.path.exists(train_dir):
    os.makedirs(train_dir)

if not os.path.exists(test_dir):
    os.makedirs(test_dir)

# Save the train and test files in their respective directories

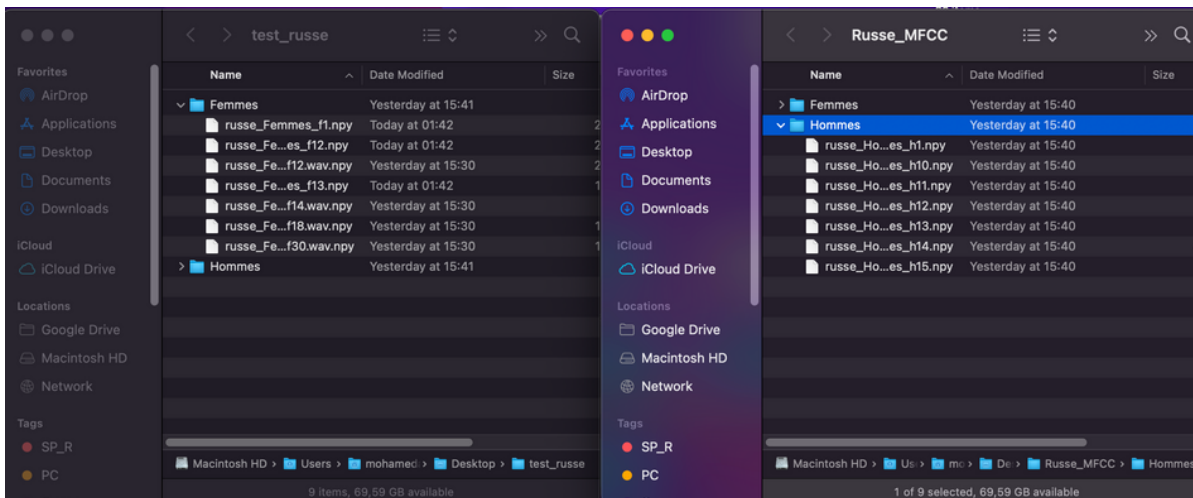
for file_path in train_files:
    if "russe_Femmes" in file_path:
        gender_folder = "Femmes"
    elif "russe_Hommes" in file_path:
        gender_folder = "Hommes"
    else:
        continue

    filename = os.path.basename(file_path)
    dest_path = os.path.join(train_dir, gender_folder, filename)
    np.save(dest_path, np.load(file_path))

for file_path in test_files:
    if "russe_Femmes" in file_path:
        gender_folder = "Femmes"
    elif "russe_Hommes" in file_path:
        gender_folder = "Hommes"
    else:
        continue

    filename = os.path.basename(file_path)
    dest_path = os.path.join(test_dir, gender_folder, filename)
    np.save(dest_path, np.load(file_path))
```

Ensuite on a sauvegarder notre data dans deux nouveau fichier **train\_russe** et **test\_russe** dans notre desktop pour y avoir acces plusra pidement et faciliter laccée



### Training the model using Gmm

```
: import os
import numpy as np
from sklearn.mixture import GaussianMixture

language = "russe"
genders = ["Femmes", "Hommes"]
train_dir = "/Users/mohamedabdallaoui/Desktop/train_russe"

# Load the train data
train_features = []
for gender in genders:
    gender_train_dir = os.path.join(train_dir, gender)
    audio_files = os.listdir(gender_train_dir)
    for audio_file in audio_files:
        if audio_file.endswith(".npz"):
            audio_file_path = os.path.join(gender_train_dir, audio_file)
            mfcc_features = np.load(audio_file_path)
            train_features.append(mfcc_features)

train_features = np.concatenate(train_features, axis=1)

: # Fit the GMM on the train data
n_components = 64 #Number of mixture components
gmm = GaussianMixture(n_components=n_components, covariance_type="diag")
gmm.fit(train_features.T)

: GaussianMixture
GaussianMixture(covariance_type='diag', n_components=64)
```

Après avoir diviser notre data en train et test split on passe a l'entraînement de notre model en utilisant l'algorithme GMM ici on utilise 64comportements

