Rapport Projet SI
-
POC d’une base d’enregistrements d’émissions LSF

Encadrant :
Mathieu DELALANDRE

Etudiant :
Loris PERDEREAU

Polytech Tours - 2022/2023

Sommaire

Introduction

LSF signifie langue des signes française.
Une émission traduite en LSF affiche un encart, une portion de l'image, dédié à un interprète traduisant l'émission en langue des signes française.

Le but du projet a été de concevoir un proof of concept (POC) de base contenant des vidéos d'émissions traduites en LSF. Nous souhaitons aussi extraire l'encart de traduction LSF dans des vidéos séparées.

A travers ces objectifs, plusieurs points avaient déjà été travaillés dans d'autres projets. D'autres qui ont dû être abordés. Ce sont les suivants :

- Établir par recherche internet une liste de programmes, de contenus, dédiés aux sourds et malentendants.
- Automatiser l'extraction des sous-titres des programmes.
- Extraire les encarts vidéo de traduction LSF.

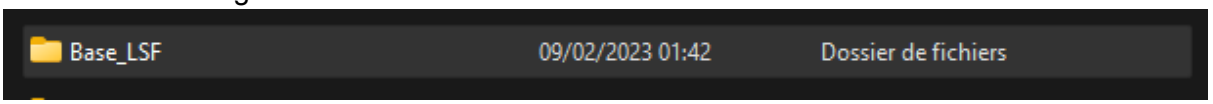
Les applications pour ce projet sont multiples. On peut par exemple penser à réaliser par la suite une plateforme web permettant la consultation d'émissions LSF.

Ce projet est en lien avec le projet station TV. Ce dernier vise à développer des services liés à la télévision.
La station TV dispose de tuners et reçoit des flux vidéos télévisés.
Elle peut afficher plusieurs chaînes à la fois et est utilisée dans l'extraction d'émissions.
Les fichiers vidéos que j'ai utilisés proviennent de celle-ci.

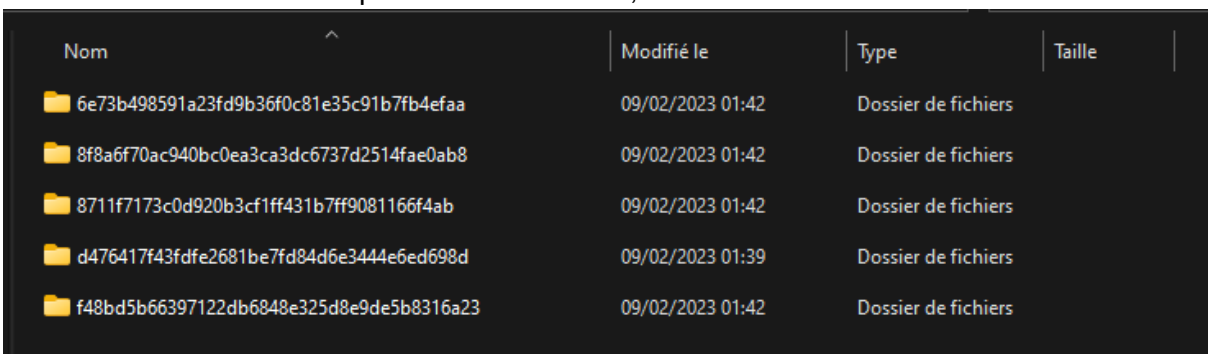
La station TV était déjà mise en place et on avait déjà une extraction vidéo de 3 chaînes de

télé entre le 14 juin et le 18 juillet 2022.

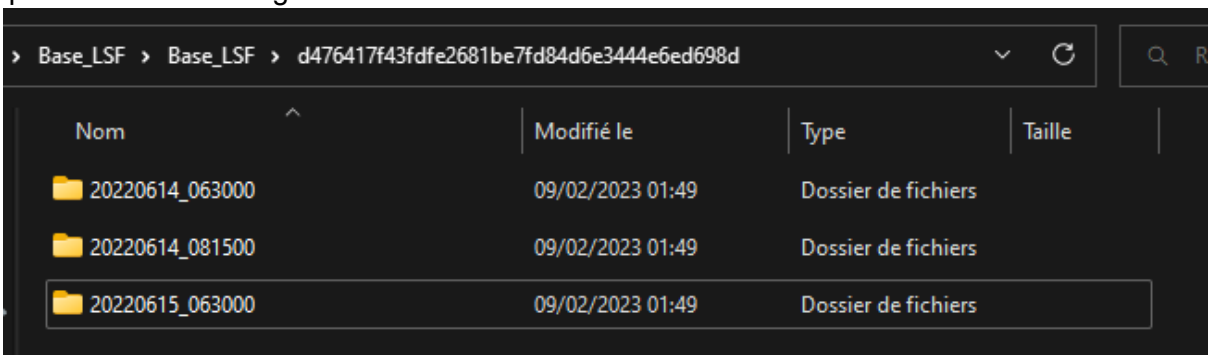
On avait aussi plusieurs scripts Python qui étaient dédiés à la réalisation de bases de vidéos.
Je n'ai pas inventé la structure de la base. En effet, elle était déjà définie.
Les bases vidéos générées consistent en un dossier :



Celui-ci contient un dossier par émission extraite, sous forme d'un hashcode :



Par exemple, d476417f43fdfe2681be7fd84d6e3444e6ed698d est le hashcode correspondant à l'émission Télématin.
Dans le dossier d'une émission, on retrouve un dossier pour chaque diffusion de l'émission que nous avons enregistrée. Dans le dossier de Télématin :



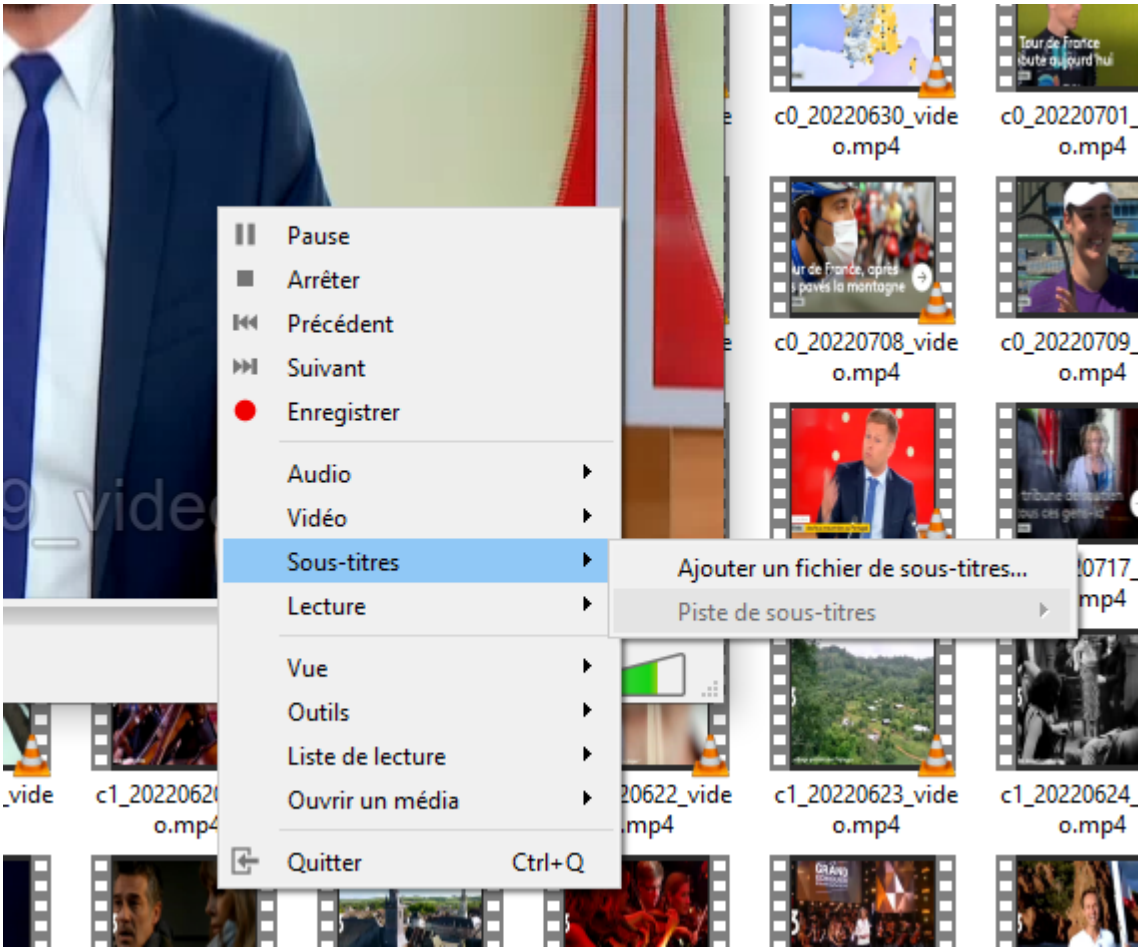
Le nom du dossier correspond à la date du début de la diffusion. On sait donc qu'on a eu une émission de télématin à 6H30 les 14 et 15 juin 2022 et une autre émission à 8H15 le 14 juin 2022. Dans chaque dossier d'émission, on retrouve un fichier audio et un fichier vidéo, tous deux au format mp4.

J'ai repris exactement le même format, si ce n'est que j'ai ajouté un autre fichier vidéo contenant l'encart LSF rogné.

Travail effectué

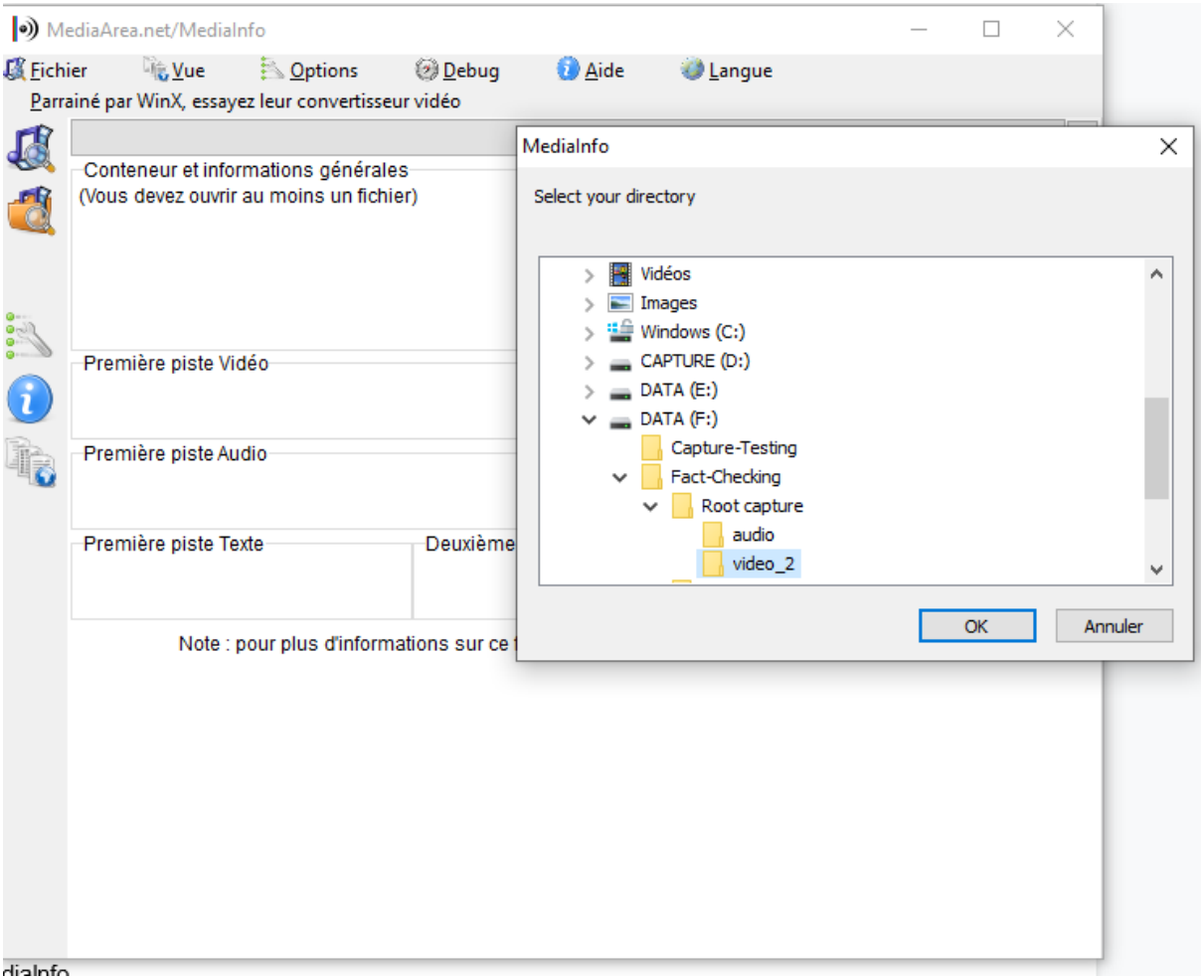
Pour ce qu'il en est du travail que j'ai effectué, ma première mission a été de déterminer la présence de sous-titres encodés dans l'extraction originale.
Il y a deux types d'encodage pour des sous-titres.
Soit les sous-titres sont soft coded, soit ils sont hard coded.
Des sous-titres qui sont hard coded sont intégrés directement dans l'image de la vidéo, on ne peut pas les retirer.
Des sous-titres qui sont soft coded peuvent être retirés ou extraits. Ils disposent de leur propre partie dans le fichier. Ils sont activables et désactivables, ne font pas partie de l'image.

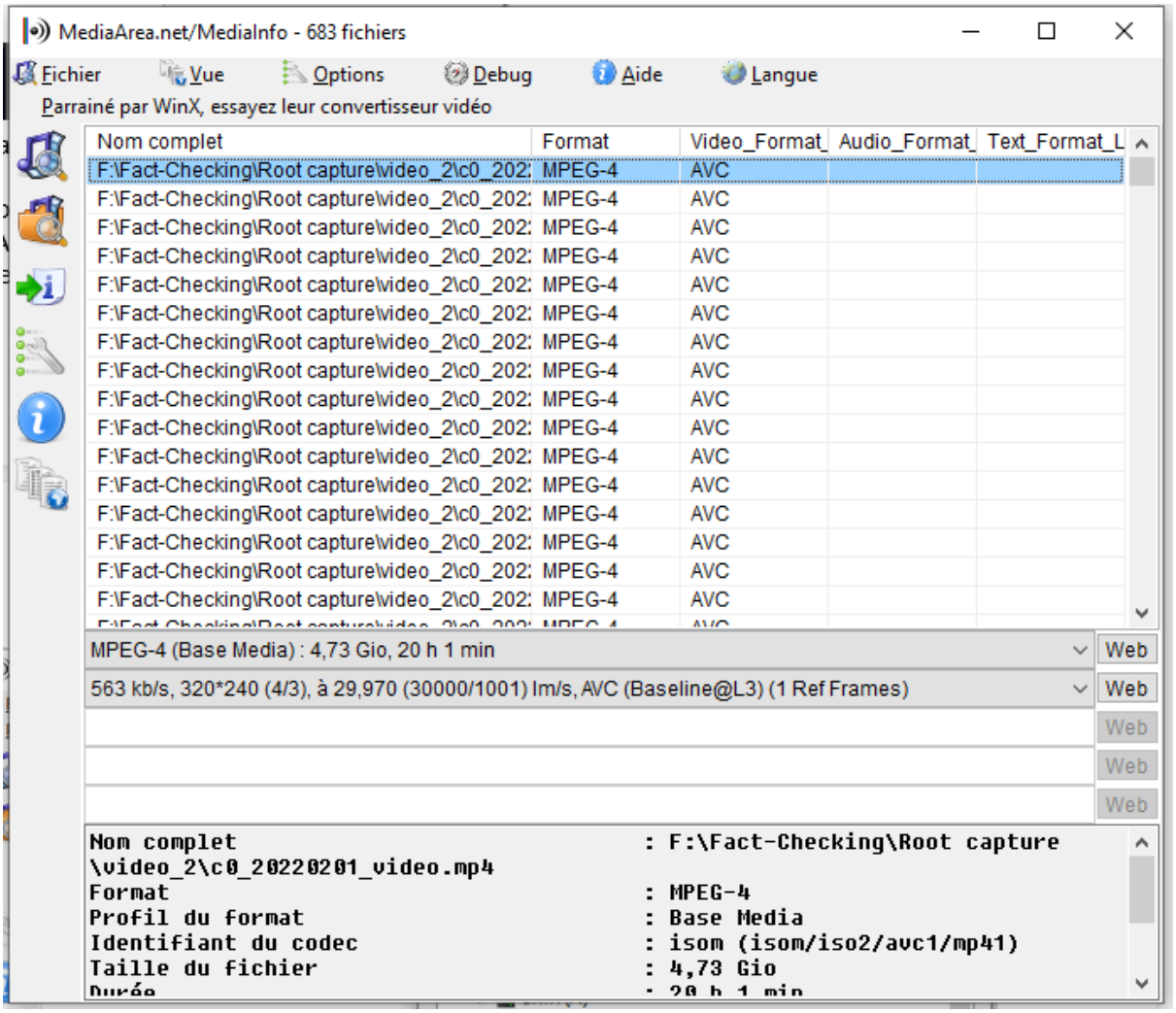
Ça nous aurait permis de produire une base avec des sous-titres dans des fichiers séparés pour d'autres genres d'exploitations par exemple pour faire des filtres de recherches de vidéos.
J'ai commencé en utilisant VLC, un lecteur vidéo qui permet d'activer ou désactiver les sous-titres dans des vidéos.
On peut voir directement sous VLC si des sous-titres sont présents dans une vidéo :
Clique-droit sur l'image de la vidéo une fois cette dernière ouverte dans VLC, sous-titres -> Piste de sous-titres.
Si l'option est grisée, la vidéo ne dispose pas de sous-titres :



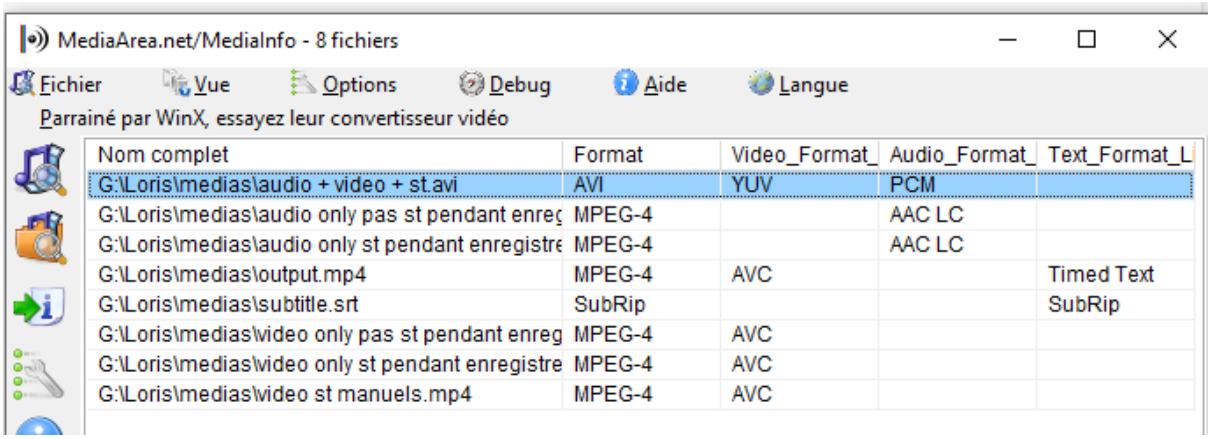
Cependant, on devait déterminer la présence de sous-titres pour une plus grosse volumétrie

de fichiers donc j'ai utilisé un autre outil qui s'appelle Media Info et qui permet de rapidement scanner l'intégralité d'un dossier.
Il permet de sélectionner un dossier et de vérifier la présence de contenu vidéo, audio et textuel dans tous les fichiers du répertoire.





Voici l'affichage dans MediaInfo lorsque les vidéos disposent de différents contenus :



On a pu vérifier très rapidement qu'aucune vidéo de la base ne disposait de contenu textuel. Ainsi, lorsque les sous-titres sont activés sur une chaîne du projet de station TV, les fichiers d'enregistrements MP4 générés ne contiennent pas de sous-titres softcoded. Ils sont hardcoded dans le flux vidéo.

Ensuite, Van Hao LE du Lifat m'a montré comment me servir de la station TV et des cartes AverMedia pour configurer différentes sortes d'extractions télé.
J'ai effectué des tests sur ces dernières, mais aucun fichier produit ne dispose de sous-titres softcoded.

J'ai travaillé depuis le poste de la station TV.
Pour lancer le programme existant, j'ai dû mettre en place Python et installer les différentes bibliothèques requises.
Le projet consiste en plusieurs fichiers Python dont les fonctions principales peuvent être appelées depuis le fichier main.
Il y a un fichier de paramétrage utilisé par plusieurs fonctions, path.py.

On a une base de fichiers XML qui répertorie pour chaque jour, la liste des émissions qui passent.
Ce sont des fichiers qui ont été récupérés avec l'API de Télérama.

Voici les fonctions principales du projet :
Premièrement, create_metadata_csv.
Dedans sont codées en dur dans une liste, des chaînes télévisées à extraire, pour créer un fichier csv.
La fonction extrait depuis les fichiers XML de la base, des informations pour créer un fichier csv à propos des émissions qui passent sur chaînes télé renseignées au début de la fonction. Chaque ligne du fichier généré correspond à la diffusion d'une émission.

La fonction create_hashcodes_csv va générer un fichier hashcodes.csv qui va contenir tous les hashcodes des émissions que l'on a sélectionnées, depuis le fichier programselection.csv.

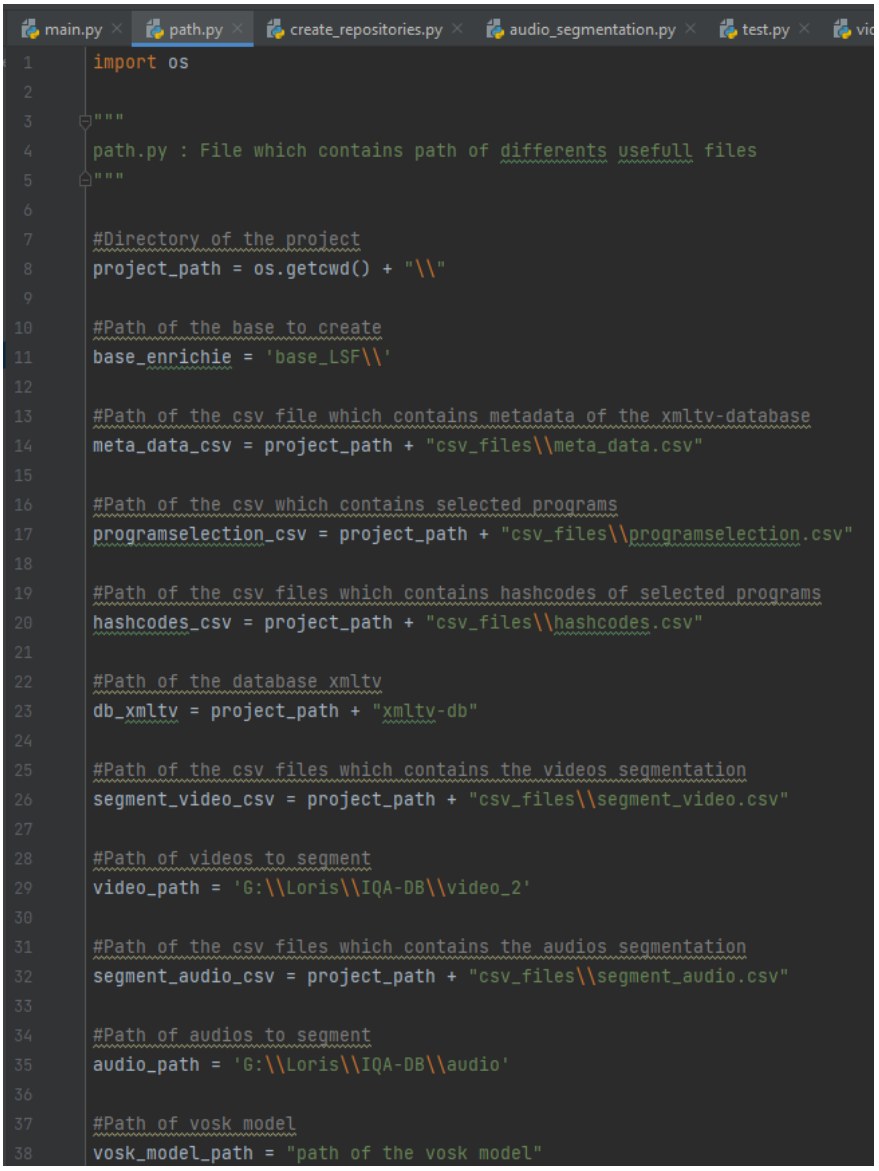
Create_repositories prend en entrée le fichier des hashcodes et s'occupe de créer les dossiers de la base vidéos. Il y a un dossier par émission.
Dans chaque dossier d'émission on retrouve un dossier par diffusion sélectionnée.

Ensuite, segment_all_audio et segment_all_video s'occupent d'extraire l'audio et la vidéo des fichiers produits par la station TV.
Donc pour chaque dossier de diffusion sélectionnée, on retrouve dedans un fichier audio et un fichier vidéo, les deux au format mp4.

Reproduire la base LSF

Installer Python et les bibliothèques nécessaires (pandas, moviepy, vosk).

Paramétrer le fichier path.py :

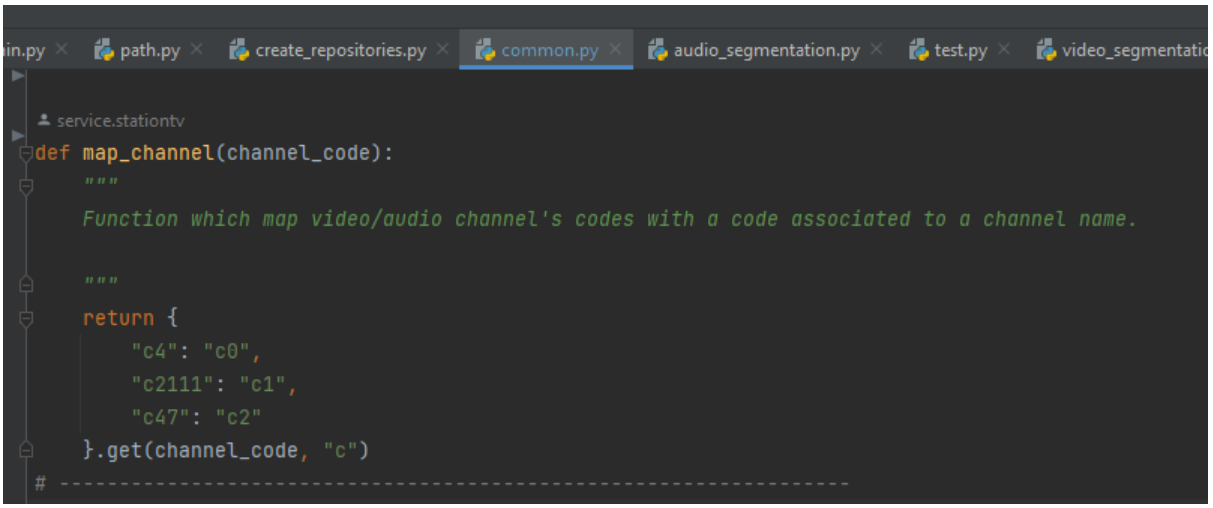


base_enrichie permet de choisir le nom de la base à créer.
On doit définir video_path, un chemin menant vers les fichiers vidéos d'une extraction de la station TV. De la même manière, audio_path est un chemin menant vers les fichiers audios d'une extraction TV.
db_xmltv est un chemin vers la base de fichiers XML correspondants aux enregistrements faits par la station TV.

meta_data_csv, programselection_csv et hashcodes_csv désignent des chemins vers des fichiers qui vont être à la fois générés et lus par le programme. Dans l'exemple ci-dessus, ils sont générés dans le dossier csv_files du projet. On peut le laisser tel quel.

segment_video_csv et segment_audio_csv ne servent pas à la génération de la base. Ils permettent de fournir des logs quant aux découpages effectués après exécution du programme.

Paramétrer common.py :
common.py contient la fonction map_channel qu'il faut configurer.



Dans l'exemple ci-dessus, les fichiers audios et vidéos extraits de la station TV commençant par c0 font référence à c4, ainsi de suite pour la suite du dictionnaire.
On a la correspondance au-dessus dans common.py, dans la fonction get_channel_name_by_code :

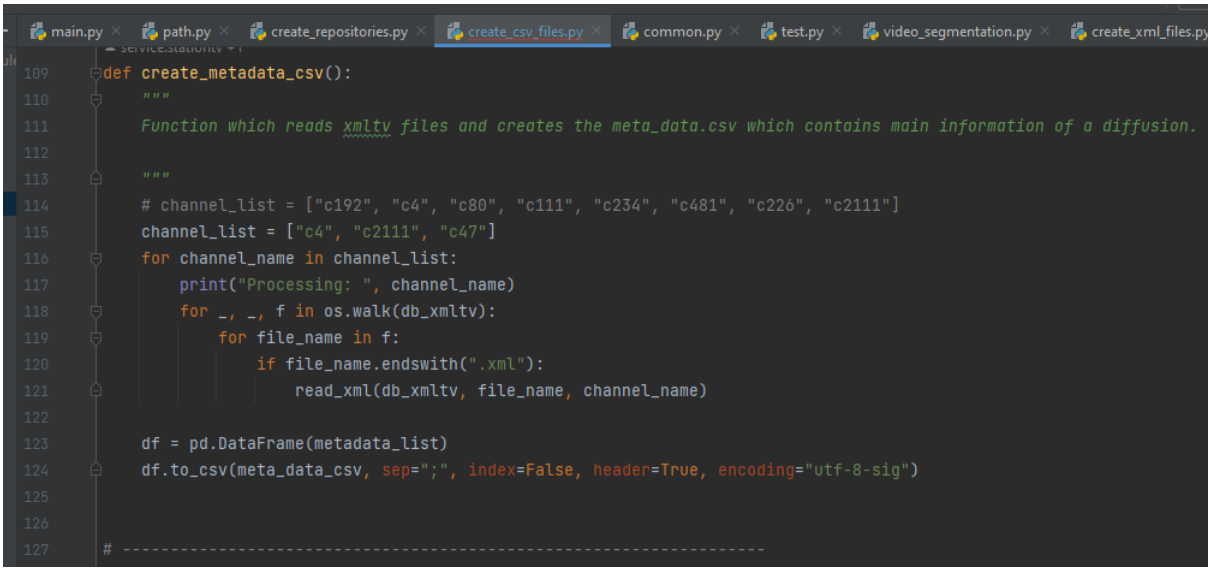
```python
def get_channel_name_by_code(channel_code):
    """
    Function which return the channel name associated to a channel code.

    :param channel_code: The channel code what we want the channel name
    :return: the channel name
    """
    return {
        "c192": "TF1",
        "c4": "France 2",
        "c80": "France 3",
        "c34": "Canal+",
        "c47": "France 5",
        "c118": "M6",
        "c111": "Arte",
        "c445": "C8",
        "c119": "W9",
        "c195": "TMC",
        "c446": "TFX",
        "c444": "NRJ 12",
        "c78": "France 4",
        "c234": "La Chaîne parlementaire",
        "c481": "BFMTV",
        "c226": "CNEWS",
        "c458": "CSTAR",
        "c482": "Gulli",
        "c1404": "TF1 Séries Films",
        "c1401": "L'Equipe",
        "c1403": "6ter",
        "c1402": "RMC Story",
        "c1400": "RMC Découverte",
        "c1399": "Chérie 25",
        "c112": "LCI",
        "c2111": "Franceinfo"
    }.get(channel_code, "c00")
```

Ainsi, on comprend que c0 dans l'exemple correspond à c4 qui correspond à France 2.

Il faut modifier map_channel de manière à associer chaque chaîne enregistrée par la station TV à sa correspondance par rapport à get_channel_name_by_code.

Paramétrer create_csv_files.py :
create_csv_files.py contient la fonction create_metadata_csv dans laquelle il faut renseigner une variable channel_list contenant la liste des chaînes extraites par la station TV :



Ici, l'extraction de la station TV contient France 2, France Info et France 5.

Une fois ces paramétrages terminés, on peut lancer depuis le main la fonction create_metadata_csv, pour générer meta_data.csv, listant toutes les émissions contenues dans les fichiers XML de xmltv-db.

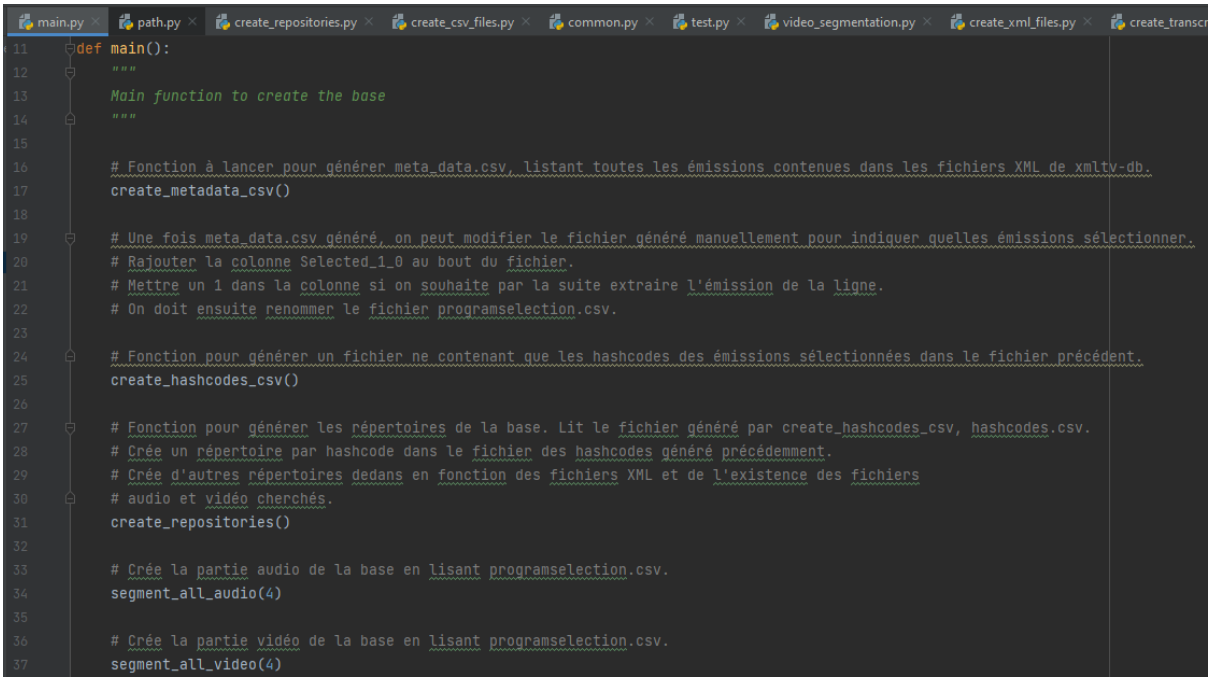
Une fois meta_data.csv généré, on peut modifier le fichier généré manuellement pour indiquer quelles émissions sélectionner.
Rajouter la colonne Selected_1_0 au bout du fichier.
Mettre un 1 dans la colonne si on souhaite par la suite extraire l'émission de la ligne.
On doit ensuite renommer le fichier programselection.csv.

On appelle ensuite la fonction create_hashcodes_csv pour générer un fichier ne contenant que les hashcodes des émissions sélectionnées dans le fichier précédent.

create_repositories génère les répertoires de la base. Lit le fichier généré par create_hashcodes_csv, hashcodes.csv.
Crée un répertoire par hashcode dans le fichier des hashcodes généré précédemment.
Crée d'autres répertoires dedans en fonction des fichiers XML et de l'existence des fichiers audio et vidéo recherchés.

segment_all_audio et segment_all_video génèrent les parties audios et vidéos de la base en lisant programselection.csv. Le paramètre donné à la fonction est un entier indiquant le nombre de cœurs du processeur à utiliser.

Tout est indiqué en commentaires dans le main :


main.py
path.py
create_repositories.py
create_csv_files.py
common.py
test.py
video_segmentation.py
create_xml_files.py
create_transcr
def main():
"""
Main function to create the base
"""
# Fonction à lancer pour générer meta_data.csv, listant toutes les émissions contenues dans les fichiers XML de xmltv-db.
create_metadata_csv()
# Une fois meta_data.csv généré, on peut modifier le fichier généré manuellement pour indiquer quelles émissions sélectionner.
# Rajouter la colonne Selected_1_0 au bout du fichier.
# Mettre un 1 dans la colonne si on souhaite par la suite extraire l'émission de la ligne.
# On doit ensuite renommer le fichier programselection.csv.
# Fonction pour générer un fichier ne contenant que les hashcodes des émissions sélectionnées dans le fichier précédent.
create_hashcodes_csv()
# Fonction pour générer les répertoires de la base. Lit le fichier généré par create_hashcodes_csv, hashcodes.csv.
# Crée un répertoire par hashcode dans le fichier des hashcodes généré précédemment.
# Crée d'autres répertoires dedans en fonction des fichiers XML et de l'existence des fichiers
# audio et vidéo cherchés.
create_repositories()
# Crée la partie audio de la base en lisant programselection.csv.
segment_all_audio(4)
# Crée la partie vidéo de la base en lisant programselection.csv.
segment_all_video(4)

Générer les encarts LSF

Pour générer les encarts LSF d'une émission, j'ai fait un script PowerShell qui utilise une commande FFmpeg.
FFmpeg est un ensemble de bibliothèques et d'outils open-source pour la manipulation de fichiers audio et vidéo.
Il permet de lire, d'encoder, de décoder, des fichiers audio et vidéo dans différents formats.
Il est largement utilisé par des développeurs et des utilisateurs pour des tâches de conversion de médias.
Les scripts Python s'en servent notamment pour extraire des passages d'une vidéo.
Dans les scripts PowerShell que j'ai produits, je m'en sers pour cet effet et aussi pour rogner l'image de la vidéo.
On peut utiliser FFMPEG à partir de lignes de commandes CMD de Windows par exemple.

J'ai fait un script PowerShell qui permet pour chaque diffusion de l'émission parcourue et qui commence à une certaine heure d'exécuter une commande FFmpeg pour extraire le passage désiré :

```
Get-ChildItem -Directory | Where-Object {
  $_.Name -match "0630|0629"
} | ForEach-Object {
  Set-Location $_.FullName
  Get-ChildItem -Filter "*.mp4" | ForEach-Object {
    $inFile = $_.FullName
    $outFile = "$($_.BaseName)-trimmed.mp4"
    ffmpeg -i $inFile -ss 00:01:00 -to 00:11:00 -c copy $outFile
  }
  Set-Location ".."
}
```

J'exécute cette commande en me plaçant dans le dossier du hashcode de Télématin (d476417f43fdfe2681be7fd84d6e3444e6ed698d).
cd G:\Loris\internship2022-julescourne\JulesCourne\Base_LSF\base_LSF\d476417f43fdfe2681be7fd84d6e3444e6ed698d

Pour chaque dossier de l'émission, si le dossier contient 0630 ou 0629 (donc si l'émission commence à 6H30 ou 6H29), j'exécute dans le dossier une commande FFmpeg permettant d'extraire, à la fois pour le fichier audio et pour le fichier vidéo, la section qui m'intéresse, de 1 minute à 11 minutes.

Un autre script qui permet pour chaque extraction du script précédent de copier la vidéo en rognant l'encart LSF :

```
Get-ChildItem -Directory | Where-Object {
  $_.Name -match "0630|0629"
```

```
} | ForEach-Object {
  Set-Location $_.FullName
  Get-ChildItem -Filter "*video*trimmed*.mp4" | ForEach-Object {
        $inFile = $_.FullName
        $outFile = "$($_.BaseName)-cropped.mp4"
        ffmpeg -i $inFile -vf "crop=237:389:2:85" -c:v libx264 -crf 18 -preset veryfast $outFile
  }
  Set-Location ".."
}
```

J’exécute cette commande en étant toujours situé dans le dossier du hashcode de Télématin.
Après vérification, dans le JT de ce créneau sur Télématin, l'encart LSF se situe dans un rectangle ayant pour point en haut à gauche : (2, 85) et en bas à droite : (238, 474).
En me basant sur ces informations, je peux faire une commande FFmpeg pour rogner la vidéo et alors extraire l’encart.
Ainsi, pour chaque fichier vidéo que j'ai déjà réduit avec la commande précédente, je crée un nouveau fichier vidéo ne conservant que l'encart LSF.

Organisation

Je n’ai pas réalisé de diagramme de Gantt pour m’organiser dans ce projet SI.

J’ai contacté en autonomie les différentes personnes qui gravitaient autour de ce projet, Guillaume Mekrouda pour parler des transcriptions audio.
Jules Courne et Van Hao LE pour les scripts Python.
J’ai aussi échangé avec Arno Balois et Matteo Doyen qui travaillaient sur un autre projet en parallèle avec la station TV.

On a organisé plusieurs réunions avec M. Delalandre, nous permettant de faire le point.
Aussi, j’ai tenu à jour quotidiennement un journal de bord que j’ai communiqué avec M. Delalandre, faisant part de mes avancées et de mes difficultés.
Il m’a permis aussi de bien voir rétrospectivement ce que j’ai fait et de plus facilement en parler, il m’a permis de faire la présentation du projet et ce rapport plus facilement.
Il est joint à ce rapport en annexes.

Conclusion

Pour conclure, j'ai rencontré plusieurs difficultés au cours de ce projet. Il a été très difficile d'établir une liste d'émissions LSF. Il y a très peu de documentation sur internet à ce sujet.
J'avais commencé à regarder individuellement des émissions extraites de la station TV pour en trouver sans succès.
Ma base n'est donc constituée que d'extraits de télématin.
Même dans cette émission, toute l'émission n'est pas traduite en LSF, il a fallu extraire les passages qui nous intéressaient.

A propos de quelques difficultés que j'ai pu avoir, il y avait une interdépendance entre les fonctions create_metadata_csv et create_hashcodes_csv.
En fait, chacune produisait un csv dont l'autre avait besoin.
J'ai pensé que pour répondre à ce besoin je devais me servir du programme de Van Hao LE pour modifier les csv.
J'avais commencé à l'utiliser et même à le modifier alors que nous n'avons en réalité pas besoin ici.
Pour corriger ça, j'ai retiré la dépendance dans create_metadata_csv en modifiant un peu le programme.

Aussi, en plus des autres paramétrages, il y a aussi une fonction map_channel(),
qui renvoie un dictionnaire de correspondance entre la liste des chaînes télé des fichiers XML et les noms de nos fichiers extraits de la station TV.
Je ne l'avais pas paramétrée dans un premier temps et ne comprenais pas pourquoi je retrouvais des fichiers audios et vidéos d'émissions, dans des dossiers qui ne leur correspondaient pas pendant mes tests.

Malgré ces difficultés, je pense avoir pu produire ce qui était attendu de moi.
J'espère avoir l'opportunité de poursuivre un PRD2 dans la continuité de ce projet.

Pour les perspectives d'amélioration du projet, on pourrait faire en sorte que l'extraction des encarts LSF soient faites par Python.
On mettrait dans le csv de sélection d'émissions deux colonnes supplémentaires, la position du début du rectangle d'encart LSF et la fin de ce rectangle.
On pourrait ensuite lancer la commande FFMPEG associée pour rogner l'image, directement depuis le Python pendant l'extraction.

Je pense aussi que ce serait intéressant maintenant qu'un proof of concept est fait d'agrandir la base en trouvant plus d'émissions en LSF.

Annexes

Voici les mails que j'ai envoyés à M. Delalandre quotidiennement, faisant office de journal de bord :

28 Novembre 2022 :
Bonjour monsieur,

Voici les quelques notes que j'ai prises pendant cette journée. Elles me servent de journal de bord.

1) Détecter la présence de sous-titres de manière automatisée dans un pool de fichiers vidéos à l'emplacement suivant :
F:\Fact-Checking

On peut voir directement sous VLC si des sous-titres sont présents dans une vidéo :
Clique-droit sur l'image de la vidéo une fois cette dernière ouverte dans VLC, sous-titres --> Piste de sous-titres.
Si l'option est grisée, la vidéo ne dispose pas de sous-titres.
Maintenant, j'aimerais automatiser cette action pour un grand nombre de fichiers.

https://stackoverflow.com/questions/43005432/check-if-a-video-file-has-subtitles
C'est pour du Linux, je dois trouver la correspondance Windows :
ffmpeg -i video -c copy -map 0:s:0 -frames:s 1 -f null - -v 0 -hide_banner; echo $?

Je vais copier une de ces vidéos dans un répertoire et faire des essais de commande avec le CMD et FFMPEG dessus.
Inutile d'aller plus loin pour le moment, les vidéos enregistrées avec sous-titres activés ne contiennent pas de sous-titres dans leurs métadonnées. Les essais ont été faits avec des vidéos complètes (audio + vidéo), mais aussi avec des fichiers MP4 contenant uniquement de l'audio ou uniquement de la vidéo. Le résultat est toujours le même : les métadonnées des fichiers ne contiennent pas de sous-titres. Ces derniers sont uniquement présents dans l'image de la vidéo. Je contacte alors Frédéric et Guillaume pour en apprendre davantage sur l'outil permettant de générer des sous-titres à partir d'un fichier audio.

2) Capture "Hello World"

Van Hao m'a déjà montré comment effectuer des enregistrements avec les cartes AverMedia, que ce soit via l'interface AVerMedia capture du bureau ou depuis le SDK avec des exemples de code source. Il m'a montré les fichiers de configuration que je pouvais modifier pour changer les productions réalisées et des portions modifiables de code source pour modifier les enregistrements.

Code source C# / WPF :

C:\Program Files (x86)\AVerMedia
C:\Program Files (x86)\AVerMedia\AVM SDK\SDK Pro\Sample\Src\C#\AVerCapSDKDemo_WPF_C#

Il m'a aussi montré comment changer de channels sur les cartes à l'aide du programme de télécommande :
E:\Applications
E:\Applications\Channel-Tune\Telecommande

J'essayerai d'en réaliser une par moi-même avec les outils de base. Ça n'a pas l'air difficile et Van Hao m'a bien montré les choses.
--> Je l'ai fait, sur un même programme avec les sous-titres activés et désactivés. Les sous-titres ne sont pas présents dans les métadonnées ! C'est l'image qui est directement modifiée. En tout cas, via AVerMedia Capture Studio, je ne peux pas enregistrer des fichiers avec des sous-titres dans les métadonnées.

J'ai aussi pu rencontrer les deux élèves en charge de la télécommande.

Pour enregistrer l'audio et/ou la vidéo, lancer :
C:\Program Files (x86)\AVerMedia\AVM SDK\SDK Pro\Sample\Bin\AVerCapSDKDemo.exe
Et modifier le fichier de configuration lu par le logiciel.

La prochaine étape maintenant est de faire de la veille web. Je dois trouver une liste d'émissions contenant des traductions en langue des signes pour pouvoir lancer une capture durant le reste de la semaine.

Merci encore pour l'aide que vous m'avez apportée et pour la clarté de vos explications.
Bonne soirée.

Cordialement,
Loris PERDEREAU

29 Novembre 2022 :
Bonsoir monsieur,

Voici la deuxième partie de mon journal de bord.
Tout n'est pas destiné à être lu, ce serait assez fastidieux.

1)

Tests confirmés dans les 4 cas de figure :
a) audio ou vidéo
b) avec ou sans st

Dans les 4 cas de figure : pas de st dans les métadonnées du média.

2)

Ajout de st manuellement dans une vidéo via VLC (option Media --> Diffuser (Ctrl+S))
https://www.movavi.com/learning-portal/how-to-add-subtitles-in-vlc-permanently-new22.html

Je ne parviens pas à ajouter des sous-titres dans un fichier MP4 sans les incruster directement dans la vidéo.
J'ai pourtant déjà manipulé des fichiers avec lesquels il était possible d'activer ou de désactiver les st. Je vais essayer d'enregistrer la vidéo et l'audio au format AVI. J'ai créé une nouvelle configuration de AVerMedia Capture SDK demo pour ça.
--> Idem

Ajout de sous-titres dans une vidéo avec FFMPEG :
https://en.wikibooks.org/wiki/FFMPEG_An_Intermediate_Guide/subtitle_options
ffmpeg -i "video st manuels.mp4" -sub_charenc CP1252 -i subtitle.srt -map 0:v -map 0:a -c copy -map 1 -c:s:0 mov_text -metadata:s:s:0 language=fre output.mp4
--> Je n'arrive pas à produire quelque chose de concluant, même en jouant avec les paramètres. Je dois me tromper quelque part.

ffmpeg -i "video st manuels.mp4" -sub_charenc CP1252 -i subtitle.srt -vcodec copy -acodec copy -scodec mov_text -metadata:s:s:0 language=fra output.mp4
--> Voilà qui fonctionne directement ! J'ai bien généré une vidéo MP4 contenant directement des sous-titres activables et désactivables à partir d'une vidéo MP4 et d'un fichier de sous-titre au format srt.

Maintenant que j'ai une vidéo avec st (softcoded) et des vidéos sans st, je veux être capable de lancer un utilitaire pour m'indiquer quelles vidéos contiennent des st.
Si on ouvre le fichier MP4 contenant les sous-titres avec un lecteur de texte comme Notepad, on peut retrouver les st inscrits en dur.

J'ai trouvé l'utilitaire MediaInfo :
https://mediaarea.net/en/MediaInfo/Download/Windows
Il permet directement d'ouvrir un dossier et d'afficher les détails des médias contenus sous forme d'un tableau.
Ainsi, pour chaque fichier, je vois rapidement s'il dispose de vidéo, d'audio ou de st.
Voir capture d'écran mediainfo.bmp en pièce-jointe.

Il pourrait être intéressant d'automatiser ce procédé, de générer une liste de fichiers contenant des st plutôt que d'ouvrir cet utilitaire manuellement.
J'ai essayé d'ajouter le répertoire d'installation de MediaInfo dans la variable d'environnement "Path" du système pour pouvoir le lancer via un cmd. Je pensais ensuite pouvoir utiliser la commande mediainfo sur cmd en précisant des paramètres mais je n'ai pour le moment réussi que à lancer le programme depuis l'invite de commandes.
--> Après concertation avec M. Delalandre, nous avons conclu que d'automatiser ce processus n'était pas utile. Je peux me contenter d'ouvrir les dossiers de la base Fact-Checking avec MediaInfo et vérifier s'ils contiennent des st.
--> Aucune vidéo de F:\Fact-Checking\Root capture\video_2\ et aucun fichier audio de F:\Fact-Checking\Root capture\audio\ ne contient de st. Les st sont hardcoded. Il faut donc utiliser un outil d'OCR pour les détecter. De nombreux outils existent déjà, open-source ou non. Exemple :
https://github.com/YaoFANGUK/video-subtitle-extractor/blob/main/README_en.md

3)

On m'a communiqué la base dont je vais devoir extraire des vidéos en LSF, emplacement : H:\IQA-DB

Je la copie au fur et à mesure sur un disque local pour que les traitements soient plus rapides (je ne suis pas convaincu).
J'ai pu copier l'intégralité du contenu de la base sur le disque G de la machine, dans un emplacement à mon nom.

Je dispose du code de Julien. J'essaye un peu de le prendre en main dans mon coin avant de demander de l'aide à Van-Hao.
Questions à poser et remarques par rapport aux codes :
- Qu'est-ce qu'un modèle VOSK ?
- Visiblement, la fonction create_metadata_csv de create_csv_files.py ne prend pas de paramètre mais une liste codée en dur au début du code.
- programselection_csv doit être le nom du csv généré.

1- Trouver comment générer le CSV indiquant quels programmes de la base extraire.
2- Trouver comment utiliser ce CSV avec le programme.
3- Extraire les médias concernés.

Van-Hao m'a présenté le programme. Il m'a montré quelques fonctions contenues dans create_csv_files_py ainsi que la structure des fichiers XML du dossier xmltv-db de Base_Video_Tagging. Nous avons essayé de le lancer sans succès : la machine est nouvelle, Python et les bibliothèques nécessaires ne sont pas installées.

J'installe Python.
Bibliothèques à installer :
- pandas

Visiblement, les fichiers Python qui m'intéressent sont plutôt ceux contenus dans Base_STVD_FC.
Je ne comprends pas tout à fait les liens entre les différents répertoires de sources. En consultant path.py de Base_STVD_FC je remarque que une base de données xmltv-db est attendue à la racine du projet. Je vais probablement devoir changer son chemin ou la déplacer.

Si on regarde le fichier main.py, la première fonction appelée (create_hashcodes_csv()) utilise déjà un fichier csv créé. Ce dernier n'a pas l'air d'avoir été créé à partir d'un des programmes.
1227 émissions sont répertoriées.
Je dois comprendre lesquelles contiennent une traduction LSF et comment les marquer comme activées ou désactivées dans le fichier csv pour extraction. C'est une question de 0 ou de 1. Un champ s'appelle "Selected_1_0", c'est certainement celui-ci.
On retrouve aussi d'autres champs plus ou moins clairs :
Selected_frequency;Selected_Duration;Duration_with_latency;;Total_h_all;Total_h_scheduled;Total_h_hidden;Total_program;Total_hascodes;selected_hashcode;selected_programs;selected_duration;duration_with_latency

Je n'aurai malheureusement pas assez avancé pour lancer une extraction...

Parler dans mon rapport des différentes tâches effectuées, du format MP4 et des tests que j'ai faits avec VLC pour y ajouter des sous-titres (que ce soit l'option diffuser ou l'option convertir, en quoi ça a abouti...).

Merci et bonne soirée.

Cordialement,
Loris PERDEREAU

5 Décembre 2022 :
Bonjour,

J'espère que vous allez bien.
Voici ce que j'ai écrit aujourd'hui et qui constitue la 3ème partie de mon journal de bord.

A propos Lecture de la partie transcription audio du rapport de Jules.
Jules écrit "La qualité des transcriptions ne semble pas convenir : elle ne reconnaît pas les noms d'hommes politiques où lors d'une interview le bruit joue sur la qualité".
Il mentionne aussi la lenteur du traitement de transcription.

Deux objectifs en parallèle :
- Essayer de s'approprier le code de manière à lancer une session d'extraction.

- Noter les émissions en LSF.
Je ne trouve pas grand chose sur internet à ce propos.
https://www.csa.fr/Proteger/Garantie-des-droits-et-libertes/Les-droits-des-personnes-handicapees/La-Langue-des-Signes
Tout comme M. Delalandre l'avait dit, il y a peu de documentation à propos des émissions disponibles en LSF.

Je vais plutôt travailler sur le premier point en premier : c'est le plus technique, je pense faire l'autre point de manière manuelle, en regardant des fragments de chaque émission. Ça risque d'être assez long.

Suivons l'ordre d'exécution des fonctions du main :

1) create_hashcodes_csv()
Cette fonction permet de créer une liste des hashcodes pour les émissions sélectionnées d'un fichier CSV (ayant le champ Selected_1_0 à 1).

2) create_metadata_csv()
Pour une liste de channels écrites en dur, pour chaque fichier xml, on lit les détails de l'émission concernée dans le fichier xml que l'on réécrit dans un autre fichier csv ?

3) create_repositories()
On crée le dossier de la base s'il n'existe pas, puis pour chaque hashcode précédent, on crée son dossier dans le dossier de la base.

4) segment_audio(4) et 5) segment_video(4)
Ici, le paramètre laissé dans l'appel de cette fonction dans le main est 4, il est censé désigner un nom de dossier.
--> Après concertation avec Jules, il s'avère être une coquille. Les fonctions qui doivent être appelées ici sont segment_all_audio et segment_all_video. Le nombre reçu en paramètre permet de définir le nombre de threads utilisés par les fonctions.

Les fonctions mettent respectivement l'audio et la vidéo dans les bons dossiers, correspondants aux hashcodes déjà définis.

6) create_all_xml()
Crée des fichiers XML pour tous les dossiers de la base, en ne lisant les diffusions que de la même date ?

7) create_all_transcripts
Crée les transcriptions à partir de l'audio des vidéos (voir section transcription audio du rapport de Jules).
Le traitement devrait être excessivement long. Je ne sais pas encore si je vais l'utiliser.
Si l'utilisation est choisie je vais devoir télécharger le modèle.

A priori je ne devrais donc modifier que le CSV de sélection des émissions.

RDV avec Jules vers 15H : explications sur des fonctions utilisées.

A propos des attributs du CSV sur lesquels j'avais un doute, on retrouve la liste dans le rapport de Jules :
Selected_frequency : Nombre de diffusions du programme s'il est sélectionné.
Selected_Duration : Durée de l'émission.
Duration_with_latency : Durée de l'émission en prenant en compte une marge supplémentaire de sécurité.

Je copie le fichier csv existant et me demande comment je vais procéder pour trier les 1226 émissions.
Est-ce que la solution la plus envisageable serait vraiment de regarder manuellement des bouts de chaque émission ?

Points à mettre au clair :
- Quelle est la base de vidéos avec laquelle je dois travailler ?
Celle que j'ai copiée depuis le disque externe à l'emplacement suivant :
"H:\IQA-DB\video_2\" ne comporte que trois chaînes.
Etait-ce à des buts de tests ? Je ne devrais pas exploiter la base vidéos de juin à juillet 2022 qui a été utilisée pour créer la base Fact-Checking ?
- Est-ce que je devrais commencer à regarder des émissions pour déterminer si elles contiennent une traduction LSF ? C'est probablement ce que je vais commencer demain en parallèle de mon rapport.

Bonne journée.

Cordialement,

6 Décembre 2022 :
Bonjour monsieur,

Voici la partie 4 de mon journal de bord.

Début rédaction rapport.

Réunion avec M. Delalandre :
La petite base sur laquelle je travaille suffit.
Le plus important est que je fasse un POC.
Je peux commencer une première extraction et affiner en ne sélectionnant que les émissions LSF ensuite.
Communication avec M. Rayar pour le PRD2.

- Trouver comment le fichier CSV de sélection des programmes est généré.
Je n'ai pas l'impression que ce soit inclus dans le programme.
Je vois une fonction create_hashcodes_csv, mais cette dernière permet simplement de ne garder que la colonne hashcode du fichier programselection.csv.
Il y a aussi une fonction create_metadata_csv, qui sert à générer un fichier csv à partir de fichiers xml de la base xmltv-db, avec les en-têtes suivantes :
Channel_Code;Hashcode;Title;Start;Stop;Length
Dans ce fichier, un même hash peut être retrouvé plusieurs fois. En effet, chaque diffusion est inscrite.
Il diffère du fichier de sélection de programmes.

Dans le rapport de Jules, il est indiqué que le fichier programselection.csv est créé à partir de la base XMLTV.
J'envoie un message à Jules à ce sujet.
--> Il me répond que ce fichier lui a été fourni par M. Delalandre et qu'il n'a pas connaissance du code qui a permis de le générer.
Je demande à Van Hao s'il a une idée.
--> Il m'explique que ce fichier est créé depuis le fichier meta_data.csv. En effet, on dispose dedans de toutes les informations nécessaires à la consitution de programselection.csv.
Il n'y a pas vraiment de programme qui puisse générer un fichier tel qu'il est attendu dans le code de Jules mais Van Hao m'a montré le code source d'un programme qu'il a fait permettant d'automatiser en partie ce travail.
Il m'a aussi expliqué que l'on appliquait des filtres supplémentaires pour la création de meta_data.csv, il a mentionné que l'on ne prenait pas les émissions trop courtes.
Mon prochain objectif est donc de créer un fichier meta_data.csv que je donnerai au programme de Van Hao et que je retoucherai de manière à correspondre à ce qu'attend le code de Jules.
J'ai rencontré une erreur de profondeur de récursion maximale. A suivre.

Merci.

Cordialement,

12 Décembre 2022 :
Bonjour,

Voici la 5ème partie de mon journal de bord.

1) Produire un bon fichier metadata.csv.
Je dois lancer la fonction create_metadata_csv en complétant correctement la variable au début de cette dernière : channel_list.
Elle contient la liste des chaînes dont on souhaite extraire des passages.
Les fichiers XML de xmltv-db contiennent des informations sur des chaînes dont.

On peut lire les fichiers vidéos pour comprendre les chaînes dont on a extrait des vidéos :
c0 : France 2 / C4
c1 : Franceinfo / C2111
c2 : France 5 / C47
Et on peut donc composer channel_list ainsi : ["c4", "c2111", "c47"].

Avant de lancer create_metadata_csv, il faut déjà générer les hashcodes.
create_hashcodes_csv...
C'est là que je ne comprends pas. J'ai besoin de générer le fichier des metadata pour utiliser le programme de Van Hao pour générer le fichier programselection.csv. Pour ce faire, je dois utiliser create_metadata_csv. Cette fonction utilise la fonction read_xml qui elle-même utilise le fichier hashcodes.csv. Elle vérifie si les hashcodes des émissions sont dedans avant de les mettre dans le fichier csv de metadata. Je rencontre une inter-dépendance entre hashcodes.csv et programselection.csv.

Je suppose que je n'ai pas besoin de hashcodes.csv. Je peux enlever la vérification concernant les hashcodes de read_xml().
En exécutant ainsi la fonction en debug, je rencontre après un certain temps d'exécution une erreur :
Process finished with exit code -1073741819 (0xC0000005)
Après recherches, cette erreur serait liée à une interdiction d'accès.
Une des pistes pour la résoudre serait d'exécuter le programme sans le mode debug.
--> En effet, l'exécution se déroule maintenant sans erreur... Mais le fichier généré est vide.
C'est normal, j'ai mal enlevé la vérification sur les hashcodes.
Je suis parvenu à générer le fichier meta_data.csv correctement !

2) L'exploiter avec le programme de Van Hao.
Je le donne au programme de Van Hao. Je rencontre une erreur à la lecture du csv.
Le délimiteur choisi n'est pas le bon. Je modifie la ligne en ajoutant le paramètre delimiter pour passer de la virgule par défaut au point-virgule :
df = pd.read_csv(r"meta_data.csv", delimiter=';')
Le fichier data.csv est bien généré.

3) Le modifier et compléter manuellement de manière à produire programselection.csv.
Je ne sais pas exactement comment remplir certains champs. Ils ne sont de toute manière pas tous utilisés.
Je pense avoir établi la relation suivante :
Duration_with_latency = Selected_duration + (Selected_frequency / 3)
--> Montrer juste dans le rapport le fichier d'origine et le fichier une fois modifié, pas besoin de rentrer dans le détail.
Inspecter l'encodage du fichier : certains caractères spéciaux sont mal affichés. --> réglé

4) L'exploiter pour réaliser une première extraction avec du contenu pas forcément LSF.
Je tente de donner le fichier programselection.csv en sélectionnant tous les contenus pour faire une première extraction.
Je dois d'abord générer hashcodes.csv. Je lance create_hashcodes_csv.
Encore une erreur d'encodage : UnicodeDecodeError: 'utf-8' codec can't decode byte 0xe9 in position 231: invalid continuation byte
Je dois d'abord convertir le fichier en UTF-8. Après conversion avec Excel (Enregistrer-sous, Outils, Web, Encodage) je rencontre toujours la même erreur. J'ai mal converti le fichier. Je passe par Notepad++ (Encoding, Convert to UTF-8). La conversion a l'air mieux. Le fichier hashcodes.csv est généré très rapidement. J'aurais d'ailleurs rapidement pu le produire moi-même en extrayant uniquement la colonne des hashcodes, mais je sais au moins que le fichier est lisible maintenant.

J'essaye maintenant de générer les répertoires de la base, mais la création des dossiers ne se passe pas comme prévue.
C'est simple, il suffit d'ajouter deux anti-slash dans la variable base_enrichie de path.py.
--> Tous les dossiers sont correctement créés.
J'installe VOSK et moviepy avec pip pour faire fonctionner correctement toutes les fonctions du programme.

Segmentation de l'audio : rien n'est généré lorsque je lance segment_all_audio. Forcément puisque je n'ai pas indiqué dans path.py le chemin des fichiers sons. Après l'emplacement renseigné, je relance le programme et rencontre des erreurs.
[NULL @ 00000215bbc48280] Unable to find a suitable output format for 'base'
base: Invalid argument
Peut-être que ce serait parce que le nom de mon dossier de base contient un espace ?
Il est peut-être donné en dur à la commande FFMPEG, sans guillemet autour.
Je modifie ça et réessaye.
Les erreurs rencontrées sont maintenant différentes, exemple :
base_LSF\003d6d9cb8d579f2f31488573d164c1f4077cade\20220618_203000\20220618_203000_audio.mp4: No such file or directory
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'G:\Loris\IQA-DB\audio\\c1_20220702_audio.mp4':
On peut voir que dans le chemin du fichier source, il y a deux antislashs avant le nom de fichier.
On peut donc modifier dans path.py le répertoire, en enlevant les antislashs au bout.
Toujours une erreur :

Input #0, mov,mp4,m4a,3gp,3g2,mj2, from
'G:\Loris\IQA-DB\audio\c1_20220623_audio.mp4':
.
.
.
base_LSF\0cee834afa0a9adf84cac001f67435a0e61f9b80\20220624_001500\20220624_00
1500_audio.mp4: No such file or directory
Les paramètres d'input_file et d'output_file ont l'air pourtant bons.
Je suis l'exécution en débug. La commande donnée à ffmpeg n'a pas l'air mauvaise.
Je la copie et tente de la lancer en CMD : je tombe sur la même erreur.
Je vais essayer de créer le répertoire de destination de l'audio manuellement.
Peut-être que le fait qu'il n'est pas complet le dérange.
C'était ça, la commande fonctionne maintenant. Je ne comprends pas pourquoi toute l'arborescence n'est pas bien créée par create_repositories(). A creuser.

Prochaines étapes :

5) Ajuster la sélection manuelle pour extraire uniquement le contenu LSF.
On peut se contenter de seulement quelques émissions.

6) Utiliser FFMPEG pour extraire les encadrés LSF des programmes.

Bonne journée.

Cordialement,
Loris PERDEREAU

15 Décembre 2022 :
Bonjour,

Voici la 6ème partie de mon journal de bord.

Je reprends à la création des dossiers des dates dans les répertoires d'émission.
La fonction create_timestamp_repositories est appelée dans create_repositories.
Son nom a l'air de correspondre à ce que je veux faire.
Son code aussi a l'air fonctionnel.
Juste en relançant la fonction create_repositories, j'obtiens bien les bons sous-répertoires.
Je n'ai pas dû les obtenir dès le départ à cause de certains mauvais paramétrages de ma part.
Beaucoup de dossiers de hash sont quand même vides.
En lançant maintenant la fonction segment_all_audio, des fichiers audios sont bien générés.

J'aurais peut-être dû commencer avec une sélection déjà plus réduite. Le temps de traitement semble assez long, et ce n'est que l'audio.
Cette extraction complète va me permettre de lire plus facilement les émissions une à une et déterminer lesquelles contiennent une traduction LSF.

Je pourrai ensuite, soit relancer une extraction en paramétrant correctement le fichier programselection.csv, soit supprimer directement les dossiers des hashcodes des émissions non LSF.
Je pense que le mieux serait de paramétrer le fichier programselection.csv. Je pourrais ainsi ajouter des champs dedans pour gérer l'extraction des encarts LSF et ajouter une fonction dans le Python qui appellerait FFMPEG pour ça.

Extraction de l'audio terminée.
La base de laquelle on a extrait l'audio contient 224 Go d'audio. Celle contenant l'extraction en a 102.
La réduction de poids ne s'explique pas par rapport à l'encodage, il est identique.
C'est plutôt que les émissions courtes ont été filtrées, ainsi que les publicités.
On a seulement gardé les émissions notées dans les fichiers xml.

Cependant, je n'ai pas inversé les t+ et t- de marge, les variables permettant de décider quand commencer et terminer l'enregistrement en prévoyant un delta de manière à ne louper aucune seconde d'une émission. Ceci ne nous intéresse pas ici, au contraire, on souhaiterait même condenser le volume de données. Avant l'extraction vidéo, je choisis donc d'inverser ces temps de marge.
Dans les fonctions segment_video et segment_audio :

```
        time_element = str2datetime(timestamp)
        time_start = time_element - timedelta(minutes=before_delta)
        time_stop = str2datetime(stop)
        time_end = time_stop + timedelta(minutes=after_delta)
-->
        time_element = str2datetime(timestamp)
```

```
time_start = time_element + timedelta(minutes=before_delta)
time_stop = str2datetime(stop)
time_end = time_stop - timedelta(minutes=after_delta)
```

Maintenant que c'est fait, je relance le programme en essayant de segmenter les fichiers vidéos.
Je choisis de ne pas limiter la sélection de manière à ne pas louper par erreur des émissions LSF.
Je n'ai pas besoin de l'intégralité d'entre-elles. Entre 4 et 10 devrait suffire pour un POC.
Malgré tout, je ne sais pas combien d'émissions sont traduites en LSF et je ne peux pas me permettre d'en effacer par erreur quelques unes s'il y en a en réalité très peu.

Extraction en cours.
Poids de la base en Go (audio + vidéo) aux temps suivants :
114 à 11H56
199 à 14H44
214 à 15H39
227 à 16H50

Pendant l'extraction, je peux chercher la commande FFMPEG que je devrais utiliser pour rogner les vidéos et ne garder que les encarts LSF. Ça n'a pas l'air compliqué :
ffmpeg -i in.mp4 -filter:v "crop=80:60:200:100" -c:a copy out.mp4
Cette commande permet de garder 80 pixels de large et 60 de hauteur à partir de la position 200;100 de la vidéo.
On extrait donc de la vidéo un encart formant un rectangle de la position (200;100) à la position (280;160) de la vidéo originale.

Je tente de m'en servir avec mes propres fichiers de test.
ffmpeg -i "video st manuels.mp4" -filter:v "crop=80:60:200:100" -c:a copy "video st manuels cropped.mp4"
Ça fonctionne parfaitement.
J'essaye d'être plus précis en extrayant uniquement un logo de la vidéo.
Dans le menu vidéo de VLC, on peut prendre une capture d'écran de la vidéo en cours. Cela nous permet de l'ouvrir avec Paint et d'avoir ainsi accès à la position des pixels recherchés.
ffmpeg -i "video st manuels.mp4" -filter:v "crop=150:154:104:874" -c:a copy "video st manuels cropped logo.mp4"
Je suis parvenu à extraire uniquement le logo.

Je dois voir comment créer une fonction python qui prendrait en compte des paramètres et lancerait cette commande.

Il serait intéressant d'être capable de déterminer un pourcentage de la largeur et de la longueur de ces positions afin que les paramètres n'aient pas à être changés en fonction de la résolution de la capture.

Pendant l'extraction vidéo, je remarque qu'un fichier audio de 1H15 est associé à une vidéo de 35 minutes. Un fichier audio de 50 minutes est associé à une vidéo de 10 minutes. On a 40 minutes d'écart. t+ et t- sont fixes et valent donc chacun 10 minutes.
J'espère que l'on ne rencontrera pas de problème pour les fichiers audio de 40 minutes ou moins.

Mail à M. Rayar pour lui expliquer mon projet actuel.
--> Réunion avec M. Rayar à 16H00 pour discuter du potentiel projet de PRD2.
Envoyer un mail à M. Rayar après noël pour lui communiquer mes nouvelles idées à propos du projet et le relancer.

travail de visualisation.
indexation sur les transcripts.
Attention au décalage audio et vidéo suite à l'enregistrement asynchrone.
...

Je peux déjà commencer à visualiser les bouts de vidéos extraits et déterminer ainsi si les émissions contiennent de la LSF ou pas.
Je démarre en supprimant tout le contenu de la colonne Selected_1_0 du fichier csv. Pour chaque dossier d'émission que je regarde, s'il est vide ou ne contient pas d'encart LSF, je mets 0. Sinon, je mets 1. J'aimerais trouver une émission LSF en maximum 60 émissions consultées de manière à pouvoir avoir au moins 5 émissions LSF dans le pool de 300 émissions.

Pendant ce procédé, j'avoue être assez inquiet. Je me rends compte que le dossier de l'émission "Drag Race France" n'a pas l'air de contenir des fichiers de l'émission en question. Plus encore, 4 émission sont enregistrées dans ce dossier, et elles n'ont pas de lien les unes avec les autres. Je regarde les fichiers XML de référence pour comprendre comment cela a pu arriver.
20220625_232500 : L'émission est bien censée avoir lieu. L'enregistrement respecte la durée prévue. L'émission enregistrée l'a été sur Franceinfo : Centre-Val de Loire alors que le XML indique une émission sur France2... L'émission enregistrée n'a rien à voir.
Je ne comprends pas cette erreur.

En regardant le code, je vois que les fonctions de segmentation attendent à la base un fichier metadata.csv, contenant une colonne channel code. De mon côté je lui ai donné le fichier programselection.csv avec les mêmes attributs que le fichier de Jules, donc sans ce code. Les extractions sont donc forcément mauvaises. Je dois laisser tomber le format de données de Jules qui ne me concerne pas, reprendre metadata.csv et ajouter un champ indiquant si le programme est sélectionné ou pas. Ce sera simple à changer.
J'ai dû libérer la salle avant de pouvoir lancer une nouvelle extraction, la base ne sera donc pas prête avant janvier...

Bonne journée.

Cordialement,

3 Janvier 2023 :
Bonjour,

Voici la 7ème partie de mon journal de bord.

Je devais donner au programme Python un bon fichier metadata.csv pour que l'extraction se fasse correctement.
Visiblement, prendre le fichier en sortie du programme de Van-Hao devrait suffire, celui que j'ai produit avant de le modifier de manière à ce qu'il corresponde au format de celui de Jules.

Modification du script de Hao pour qu'il copie aussi les colonnes Start et Stop du fichier en entrée, qui seront utilisés pour créer les dossiers de timestamps. Cela permet de créer les dossiers après filtrage des émissions sélectionnées et donc de créer beaucoup moins de dossiers. (--> Au final, le même nombre de dossiers a été généré, je ne pensais pas.)

Reformater les cellules des csv générés de exponentiels vers nombres simples pour que le fichier soit bien lu.
Je commence à générer la base vidéos avant midi. Je commence par la vidéo plutôt que l'audio pour reconnaître plus facilement si la génération se passe bien ou pas. Je pourrais ainsi confirmer que les vidéos générées correspondent bien aux émissions mentionnées.

Le dossier 0b2e65d60feafea6a9d1b3a5382ff0509ca31e49 contient 20220626_151000.
Lorsque j'ouvre la vidéo, c'est un enregistrement de France 3 qui apparaît. Or, dans les fichiers CSV, cela devrait correspondre à "Vivement dimanche", une émission de France 2...
--> Après vérifications, "Vivement dimanche" est bien une émission de France 3, ce n'est pas ce qui est marqué dans les fichiers xml ??

Je laisse poursuivre l'extraction à midi, mais je pense qu'il va falloir réparer quelque chose...
En effet, l'extraction ne correspond pas. Les erreurs sont identiques aux précédentes.
Je vais débugger le programme et comprendre comment cette erreur a lieu.

Déjà, est-ce que les fichiers XML sont bons ? Peut-on retrouver réellement les bonnes émissions en se focalisant uniquement sur les XML et l'enregistrement initial ?
--> Pour dimanche prochain

Pour la ligne suivante :
c4 France 2 baa5711f6efabb5eb8576e68e2d6bc7354d1d708 Vivement dimanche prochain
20220626160500
20220626165900

C'est la vidéo suivante qui a été choisie :
c1_20220626_video.mp4 Extrait vers 9H18 et quelques.

Si je regarde pour l'équivalent en c0 à 9H18... On tombe en effet sur "Vivement dimanche" !
C'est une bonne nouvelle, au moins les timecodes ont l'air de correspondre. Il faut maintenant que ce soit la bonne vidéo qui soit sélectionnée. Qu'est-ce qui détermine quel fichier est sélectionné ?
Il suffisait de renseigner correctement la fonction map_channel...

```
def map_channel(channel_code):
    """
    Function which map video/audio channel's codes with a code associated to a channel name.

    """
    return {
        "c4": "c0",
        "c2111": "c1",
        "c47": "c2"
    }.get(channel_code, "c")
```

Ça devrait mieux fonctionner.
Erreur 0xC0000005 : erreur de violation d'accès. Il ne faut pas être en train de modifier le fichier lu.

Je relance une extraction et vérifie que les vidéos correspondent bien aux bonnes émissions, je vérifie une émission pour chaque chaîne pour m'assurer que le mapping est correct.
--> OK !

Maintenant, je regarde les émissions une à une et met à jour le fichier csv de sélection de manière à sélectionner quelques émissions LSF.

Je peux déjà noter les émissions qui m'ont été communiquées par M. Delalandre :
L'oeil et la main / France 5 :
On ne retrouve pas l'émission dans les fichiers XML.
Après vérifications, malheureusement, cette émission n'a pas eu lieu pendant la période d'enregistrement de la station TV.
--> https://www.replay.fr/l-oeil-et-la-main.html

Télématin / France 5 :
L'émission n'est pas sur France 5 mais sur France 2 (c'est un détail).
Aucun dossier de timestamps n'a été généré pour cette émission ??
Pourtant, le programme dure 90 minutes, ce devrait être assez pour extraire des vidéos.

Une émission a eu lieu le 16/06 : nous n'avons pas l'enregistrement.
Une autre devrait avoir eu lieu le 19/06 : je retrouve bien l'émission à partir de 2H17 dans le fichier vidéo c0_20220621_video.mp4.
Je ne comprends pas pourquoi le bon dossier de timestamps ne s'est pas créé.
L'émission n'est toutefois pas traduite en LSF !

Le journal de 20h / France 2 :

De la même façon, aucun dossier de timestamps n'a été généré.

Laisser l'extraction se faire.
Trouver pourquoi certains fichiers timestamps ne sont pas générés.
Et surtout trouver quelques émissions LSF à extraire correctement !

Bonne journée.

Cordialement,
Loris PERDEREAU

5 Janvier 2023 :
Bonjour,

Voici la 8ème partie de mon journal de bord.

J'ai déjà vérifié chez moi le comportement de la fonction de création des dossiers de timestamps.
Le comportement a l'air bon. Pour Télématin par exemple, si les fichiers c0_20220619_video.mp4 et c0_20220619_audio.mp4 existent dans la base, les dossiers devraient être générés. Je vérifie leur existence. Les fichiers existent bien, mais les dossiers ne sont pas générés.

Faire attention entre chaque modification manuelle de fichiers CSV à bien formater les colonnes Start et Stop de manière à ne pas laisser Excel utiliser la notation scientifique. Cela empêche le bon fonctionnement du programme.

Le comportement observé sur le PC du bureau des doctorants et le mien n'est pas identique.
Les fichiers CSV utilisés sont différents. Sur mon PC, on vérifie bien l'existence des fichiers audio et vidéo pour CHAQUE diffusion de l'émission Télématin. Sur le PC du bureau des doctorants, uniquement la date du 18 juillet est vérifiée, date pour laquelle nous n'avons pas de fichier. Je creuse le problème.
--> C'est normal, le fichier que j'utilise est le fichier réduit à 320 lignes, ne contenant pas toutes les émissions, c'est le fichier une fois réduit par le programme de Van Hao. Pour générer les dossiers complets j'ai besoin de la liste de toutes les émissions, donc avant sélection des programmes. Cela m'embête car je vais créer trop de dossiers par rapport à ce que je vais utiliser. Ce n'est pas si grave, je pourrai supprimer les dossiers vides par la suite.
Je régénère un fichier csv correct et relance la création de dossier.
--> Les dossiers de timestamps sont bien générés.
Supprimer les dossiers vides :
for /f %d in ('dir /ad /b /s') do rd "%d" 2>nul

Maintenant je dois déterminer quelles émissions contiennent du langage LSF.
Je lance une extraction comme la dernière fois, mais avec tous les dossiers et donc la génération de toutes les vidéos cette fois.
Le traitement était déjà assez long sans extraire toutes les émissions, ça le sera donc encore plus maintenant.
Ce n'est pas si grave, je peux déjà vérifier les émissions extraites lors des précédentes extractions.

En lisant le programme, je comprends que le comportement que j'avais interprété n'était pas la réalité.
Les fonctions de générations vidéo et audio doivent elles aussi prendre en entrée le fichier meta_data.csv, contenant les dates de début et de fin de chaque émission. C'est donc bien dans ce fichier que je dois ajouter pour chaque diffusion d'émission si on la sélectionne ou pas. Je pensais n'avoir besoin de donner que un fichier csv réduit, contenant une ligne par

émission, et si elle est sélectionnée ou pas. A mon sens le reste était lu depuis les fichiers XML. Ce n'est pas le cas. Ce n'est pas grave non plus. Mais en fait, le programme de Van-Hao n'a pas besoin d'être utilisé dans ce cas.

Télématin est en fait bien en partie en LSF, mais la majorité de l'émission ne l'est pas.
https://interpretelsf.blog/2011/09/11/rendre-linformation-televisee-accessible-aux-sourds-4/
Deux journaux d'informations d'une durée d'environ cinq minutes, sont interprétés dans l'émission à 6h30 et à 8h55, du lundi au vendredi. Les journaux du samedi à 7h00 et à 8h40 seraient également interprétés.

Réunion avec M. Delalandre :
J'annule l'extraction en cours.
Parler de la difficulté supplémentaire d'extraire des émissions comme Télématin, ne contenant qu'une partie de l'émission en LSF.
Je ne sélectionne que Télématin pour extraction; Je modifie meta_data.csv de manière à ajouter un champ Selected_1_0 et je mets un 1 devant toutes les émissions de Télématin (très rapide avec des filtres).
Je modifie les fonctions appelées par segment_all_video et segment_all_audio de sorte à ne traiter depuis meta_data.csv que les lignes possédant 1 dans Selected_1_0.
Je réinverse les t- et t+, pour éviter de rogner du contenu LSF présent en début ou fin d'émission.
--> Finalement je les mets en commentaire, je ne veux garder que l'émission sans marge dans un premier temps. On peut les décommenter si on le désire par la suite.
Je relance le programme en refaisant la création de dossiers, et en lançant segment_all_video et segment_all_audio.
--> Après correctifs, le programme se lance et enregistre correctement et uniquement les émissions de Télématin.
Les émissions extraites font bien 90 minutes, le contenu est cohérent.

Je supprime tous les dossiers vides de la base avec la commande cmd au-dessus (il faut lancer la commande deux fois, une fois pour supprimer les dossiers vides des timestamps contenus dans les dossiers des hashcodes, puis une autre fois pour supprimer les dossiers des hashcodes qui ne contiennent maintenant plus aucun dossier).

Je regarde si le contenu LSF est toujours présent au même moment. Il faudra dans tous les cas faire l'extraction au moins en partie manuellement :
L'émission commence parfois à 6h29, parfois à 6h30. Elle reprend comme une autre émission, parfois à 8h10 parfois à 8h15, elle est interrompue par le journal à 8h00.

20220617_063000 :
contenu LSF de 1 minute à 10 minutes 35. Je pense que cela dure globalement 10 minutes, de 1 minute à 11 minutes.
J'ai aussi retrouvé du LSF à un endroit non prévu, probablement car contenu politique : De 1 heure et 11 minutes à 1h et 21 minutes.

Le 17 juin n'était pas un samedi.

20220617_081500 :
La vidéo commence directement avec du contenu LSF, qui dure jusqu'à 5 minutes.
LSF de 44:30 à à 50:00.
Les encarts LSF ne sont pas systématiquement présents à la même position, même dans une même émission : le premier encart était à gauche et le second à droite.

20220618_063000 :
LSF de 1 minute à 10 minutes.
Cette fois, pas de LSF de 1:11:00 à 1:21:00.
Le 18 juin était pourtant un samedi, on devrait retrouver du LSF au jt de 7h00 (donc à partir de 30 minutes). Ce n'est pas le cas et donc pas systématique.

20220618_081500 :
On ne commence par directement avec du LSF cette fois.
Pas de LSF non plus à 8h40 malgré le samedi.
Pas de LSF à 8H55 (40 minutes) non plus.

J'ai vérifié d'autres émissions, dont certaines démarrant à 6h29. Le JT en LSF commence ici également après une minute d'émission.
Je choisis donc d'extraire de toutes les émissions démarrant à 6h29 ou 6h30 la vidéo et l'audio de la première minute à la 11ème.

J'utilise la commande Powershell suivante, avec FFMPEG dans chaque dossier d'émission commençant à 6H29 ou 6H30 :

```
Get-ChildItem -Directory | Where-Object {
  $_.Name -match "0630|0629"
} | ForEach-Object {
  Set-Location $_.FullName
  Get-ChildItem -Filter "*.mp4" | ForEach-Object {
    $inFile = $_.FullName
    $outFile = "$($_.BaseName)-trimmed.mp4"
    ffmpeg -i $inFile -ss 00:01:00 -to 00:11:00 -c copy $outFile
  }
  Set-Location ".."
}
```

J'exécute cette commande en me plaçant dans le dossier du hashcode de Télématin (d476417f43fdfe2681be7fd84d6e3444e6ed698d).
cd G:\Loris\internship2022-julescourne\JulesCourne\Base_LSF\base_LSF\d476417f43fdfe2681be7fd84d6e3444e6ed698d
Pour chaque dossier de l'émission, si le dossier contient 0630 ou 0629 (donc si l'émission commence à 6H30 ou 6H29), j'exécute dans le dossier une commande FFMPEG permettant

d'extraire, à la fois pour le fichier audio et pour le fichier vidéo, la section qui m'intéresse, de 1 minute à 11 minutes.

Après vérification, dans le JT de ce créneau sur Télématin, l'encart LSF se situe dans un rectangle ayant pour point en haut à gauche : (2, 85) et en bas à droite : (238, 474).
En me basant sur ces informations, je peux faire une commande FFMPEG pour n'extraire
Ainsi, toujours en étant situé dans le dossier du hashcode de Télématin, j'exécute la commande Powershell suivante :

```
Get-ChildItem -Directory | Where-Object {
  $_.Name -match "0630|0629"
} | ForEach-Object {
  Set-Location $_.FullName
  Get-ChildItem -Filter "*video*trimmed*.mp4" | ForEach-Object {
    $inFile = $_.FullName
    $outFile = "$($_.BaseName)-cropped.mp4"
    ffmpeg -i $inFile -vf "crop=237:389:2:85" -c:v libx264 -crf 18 -preset veryfast $outFile
  }
  Set-Location ".."
}
```

Ainsi, pour chaque fichier vidéo que j'ai déjà réduit avec la commande précédente, je crée un nouveau fichier vidéo ne conservant que l'encart LSF.
Cela fonctionne, tous les fichiers sont générés. L'extraction est complète pour les 30 émissions de Télématin commençant à 6h29 ou 6h30.
Le résultat est, selon moi, satisfaisant.

Merci et bonne journée.

Cordialement,
Loris PERDEREAU