POLYTECH TOURS

64 avenue Jean Portalis

37200 TOURS, FRANCE

Tél +33 (0)2 47 36 14 14

www.polytech.univ-tours.fr

Rapport de stage de fin d'études 2021-2022

Analyse de données TV

CONFIDENTIEL

Entreprise :

Laboratoire LIFAT

64 Avenue Jean Portalis,

37200 Tours - France

Etudiant :

Courné Jules

Promo 2020-2023

Tuteur Entreprise :

Rayar Frederic

Enseignant chercheur

Tuteur académique :

Bocquillon Ronan

Enseignant chercheur

Table des matières

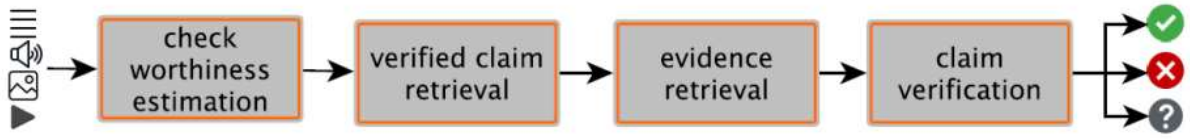
Introduction

Introduction au Fact-checking

La dernière décennie a vu accroître la désinformation et l'apparition de faits faux ou erronés au travers de différents médias : Réseaux sociaux, télévision, radio, articles, etc. Cependant lorsque cette désinformation touche la sphère politique, cela devient un enjeu majeur pour chaque citoyen vivant en démocratie. Ce régime sous lequel nous vivons est basé sur le pluralisme politique et donc sur le débat et la confrontation d'idéologies, tous deux s'appuyant sur des faits. Les systèmes permettant de valider ou non un fait énoncé sont au cœur des enjeux démocratiques.

Ainsi plusieurs organismes de vérification de faits ont émergé ces dernières années. Cependant ces organismes s'appuient prioritairement sur des moyens humains et, au vu de la quantité d'informations déferlant sur les médias, la conception d'outils automatisés capable d'aider l'Homme dans sa prise de décision semble prioritaire. Ces outils, éléments essentiels du Fact-Checking, permettent d'accomplir plusieurs tâches comme l'identification des affirmations dignes d'être vérifiées, la vérification des affirmations ou bien l'identification des rumeurs et des satire. A titre d'exemple nous avons « Fact Check Explorer » qui est un outils de Fact-checking développé par Google.

Fact-Checking réseaux sociaux



En effet, en plus des moyens humains permettant de vérifier la véracité des faits, il existe des solutions automatisées pour le Fact-Checking des réseaux sociaux. Désormais les réseaux sociaux contribuent largement à la politique. La plupart des personnalités politiques et des citoyens ont un compte Facebook ou twitter, outils de d'échanges démocratique. Le Fact-Checking des réseaux sociaux est un cas de Fact-Checking classique s'appuyant principalement sur des données textuelles à partir de sites web ou de blogs. Ce pipeline

représenté sur la figure ci-dessus nous montre les différentes étapes d'un Fact-checking classique. Premièrement, il faut trouver des faits qui valent la peine d'être vérifiés au travers de différentes sources. En second, il faut vérifier si le fait choisi a déjà été vérifié précédemment. Troisièmement, il faut rassembler des éléments de preuve pertinents pour aider à comprendre le contexte et la véracité du fait. Finalement, il faut décider si la demande est erronée, principalement fausse, imprécise, principalement vraie ou vraie.

Fact-Checking TV

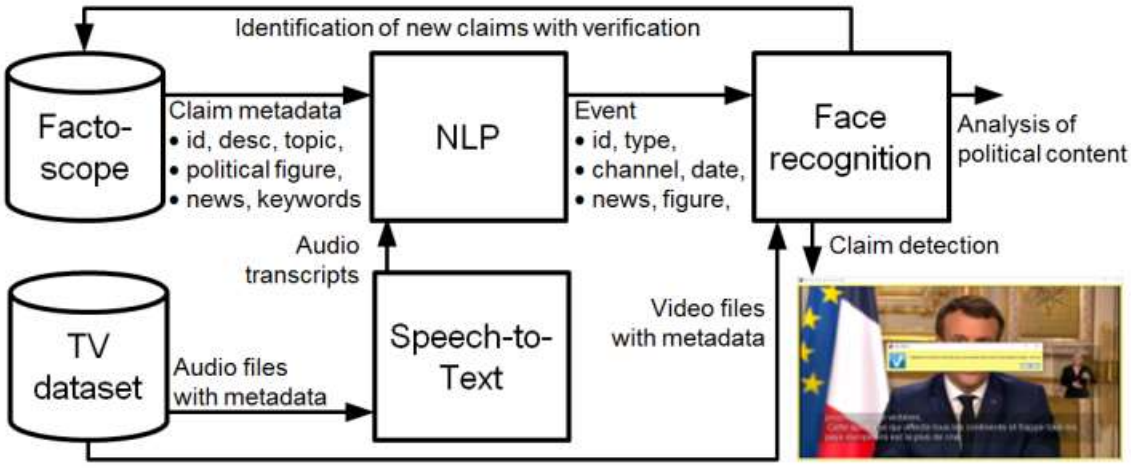
De nos jours, la majorité des technologies de Fact-Checking sont concentrés sur les données textuelles issues des réseaux sociaux. Cependant, l'information politique apparaît principalement à la télévision dans des émissions comme des débats, des interviews ou des actualités. Ce fait conduit à la nécessité de tirer parti de méthodes capables d'analyser le contenu multimédia (image, audio, vidéo) afin de réaliser vérification automatisée des faits multimédias. Cependant pour pouvoir analyser des contenus multimédias il faut avoir une base et, à ce jour, il n'existe que très peu de bases de faits politiques. Ce tableau ci-dessous ressence les bases multimédias de faits politique existantes.

Datasets / systems	Sources	Claims	Videos
Politifact (2014) [8]	Web blogs	106	No
ClaimRank (2017) [10]	US presidential debates	880	4 with transcripts
Tathya (2017) [7]	US presidential debates	967	18 with transcripts
LIAR (2017) [9]	Politifact	12, 836	No
FEVER (2018) [11]	Wikipedia	185, 445	No
ClaimBuster (2020) [4]	US presidential debates	1, 673	33 with transcripts
Full Fact (2020) [24]	UK political TV shows	1, 570	14 with annotations

Nous pouvons voir que les bases de faits politique peu nombreuses. Les bases les plus volumineuses sont à données textuelles. Les bases multimédias sont peu volumineuses. La plus importante vérifie seulement 1673 faits et ne comporte que 33 transcriptions. De plus il n'en existe pas pour la politique française. Ainsi le projet station TV a pour but de créer une large base de données multimédias (réseaux sociaux, télévision, etc.) français pour les données à faits politique.

Le projet station TV et base de capture

Mené depuis 2019 par le Laboratoire LIFAT, le projet station TV a pour objectif le développement de nouveaux services numériques autour de la télévision, du Replay et d'Internet. La station TV est une plateforme de calcul parallèle pour le traitement automatique des chaines de télévision de la TNT. En dehors du développement de la plateforme elle-même, le projet station TV recouvre aujourd'hui de multiples services pour la capture smart de vidéos et d'images, l'analyse TV, le Fact-checking et le Scraping Web de données TV. L'image ci-dessous représente l'organisation des différentes bases de données et comment elles sont utilisées pour le Fact-checking TV.



Nous avons deux bases servant à la vérification de faits politiques. Premièrement, la base Factoscope. Factoscope est un projet conçu par l'école publique de journalisme de Tours (EPJT) qui a pour but de vérifier les propos des personnalités politiques et de donner accès à ce travail de vérification. Cette vérification est faite via des moyens humains (Fact-checkers). Antoine Roura, un stagiaire sur le projet station TV durant l'année 2020-2021, a effectué un travail visant à créer une base de données xml constituée de tous les faits répertoriés sur le site Factoscope. Pour le moment nous avons une base de 1330 faits vérifiés. Cependant le site Factoscope est en évolution constante et le nombre de faits vérifiés varie de jour en jour.

Pour chaque faits de la base de données factoscope nous avons les informations suivantes :

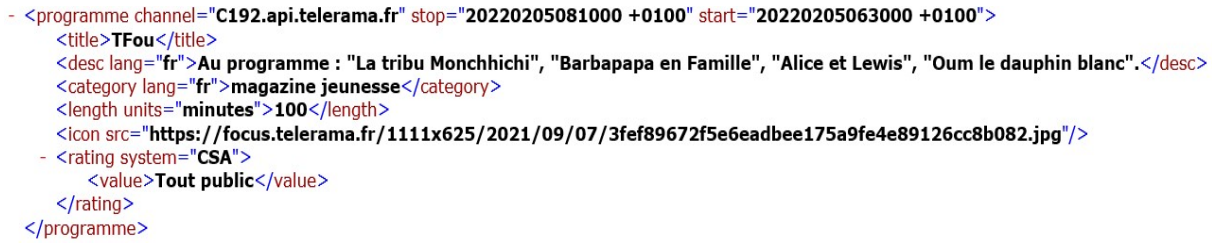
- Un identifiant
- L'url du site Factoscope sur lequel le fait est mentionné
- L'auteur
- La date d'énonciation
- Le titre
- La véracité
- Les mots-clefs
- Le texte descriptif
- L'url du site source où le fait a été repéré

La deuxième base nommée « TV dataset » est issue de la période de capture lancé par le laboratoire LIFAT dans le cadre des élections présidentiels 2022 en France. Cette période de capture s'étend du 01/02/2022 au 01/05/2022. Au cours de cette période, 8 chaînes ont été capturés 20h par jours. Ces 8 chaînes sont les plus susceptibles de contenir des diffusions à caractère politique durant les élections présidentielles. Nous avons donc 720 fichiers vidéo et 720 fichiers audio (8 chaines x 90 jours). Cependant, pour des raisons techniques, des jours n'ont pas été capturés pour certaines chaines. Le tableau ci-dessous recense les jours manquants :

		Number of files				
Channel name	Channel code	February (28 days)	March (31 days)	April + May 01 (31 days)	Missed days	Sum by channel
TF1	c0	24	29	31	18/02; 21/02;26/02; 27/02;26/03;27/03	84
France 2	c1	24	29	31	18/02;19/02;20/02;21/02;26/03;27/03	84
France 3	c2	26	29	31	18/02; 21/02;26/03;27/03	86
Arte	c3	26	29	31	18/02; 21/02;26/03;27/03	86
La chaîne parlementaire	c4	26	29	31	18/02; 21/02;26/03;27/03	86
BFMTV	c5	26	29	31	18/02; 21/02;26/03;27/03	86
CNEWS	c6	25	29	31	17/02;18/02; 21/02;26/03;27/03	85
Franceinfo	c7	26	29	31	18/02; 21/02;26/03;27/03	86
Total		**203**	**232**	**248**		**683**

Ainsi, après rectification des jours manquants, nous pouvons voir que nous avons 683 fichiers vidéo et 683 fichiers audio. Les fichiers audio et vidéo sont au format mp4. En complément des bases de captures audio et vidéo, nous avons 90 fichiers xml représentant les données XMLTV des chaînes de télévision. Ces données sont issues des programmations télévisuelle du site « Télérama » et nous communiquent des informations sur les chaines de télévisions ainsi

que sur les programmation de chaque diffusions. L'image ci-dessous nous donne un aperçu des données XMLTV pour la diffusion de TFOU sur TF1 le 5 février 2022 à 08h10.



Nous avons donc deux base de données : Une avec des données brutes (vidéo, audio, xml) et une avec des faits vérifiés. Dans le cadre du projet station TV il faudra tout d'abord passer de la base brute vers une base structurée. Puis, à partir de cette base structurée, l'objectif sera de créer des transcriptions audio pour appliquer des algorithmes de NLP (Natural Langage Processing) afin de matcher les différents mots-clés avec la base factoscope. Par la suite, nous pourrons appliquer des algorithmes de reconnaissance facial sur les vidéo associé aux diffusions et déterminer si le fait existe déjà dans la base Factoscope et ainsi vérifier la véracité des faits. Sinon, il faudra analyser la contenu politique.

Un travail allant de l'étape des transcriptions audio jusqu'à la vérification des faits a déjà été réalisé par Alexis Korek, un stagiaire de l'année 2020-2021.

Vocabulaire utilisé

Afin de bien comprendre ce dont nous parlons et de ne pas faire d'amalgame, établir un vocabulaire semble primordiale.

Une émission : (exemple : Le journal 19/20)

Diffusions : Une émission sur une chaine avec une date de commencement et d'arrêt bien précis (exemple : Le journal 19/20 sur TF1 ayant commencé le 01/02/2022 à 19h01 et ayant fini le 01/02/2022 à 20h02).

Programme : L'ensemble des diffusions d'une émission donnée. (exemple : Le journal 19/20 sur TF1).

Problématique

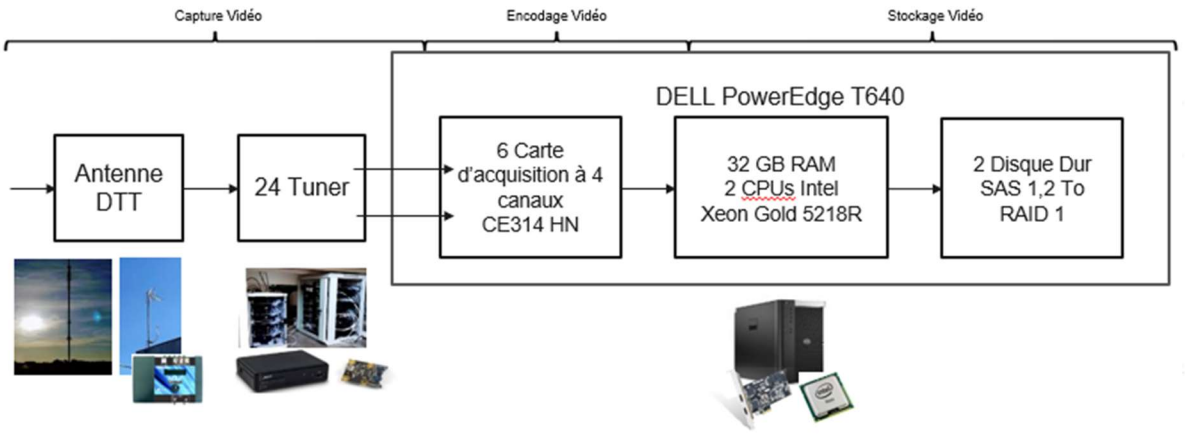
À la suite de la campagne de capture de faits politiques s'instituant dans le projet de la station TV, nous avons une base de données brute constituée de 14400h de vidéo et d'audio ainsi que de 90 fichiers xml contenant des données XMLTV. Afin de pouvoir utiliser ces données dans le cadre du Fact-Checking, nous devons analyser et organiser ces données. La problématique principale soulevée dans ce stage est donc la suivante : Comment passer d'une base brute vers une base structurée ?

Cette problématique soulève des questions d'arborescence, de nommage, d'analyse du contenu ainsi que de conception. Nous verrons tout au long de ce rapport comment la structure de la base a été pensée et créée.

La deuxième problématique porte sur l'extraction des mots-clés de la base et la mise en commun avec la base de données issue de Factoscope.

Environnement matériel

La station Dell PowerEdge



La station Dell PowerEdge est composée de 24 tuners reliés à l'antenne du laboratoire permettant de capter 24 chaînes de télévision en temps réel. Ces 24 tuners sont reliés en hdmi aux cartes d'acquisitions. Ces cartes d'acquisitions CE314 HN sont au nombre de 6 avec

chacune 4 canaux. Le flux vidéo et audio est ensuite récupérer grâce au logiciel SDK AVerMedia.

Afin de traiter tous les calculs d'acquisition, la station a besoin de puissance de calcul. C'est pourquoi elle possède 32Go de RAM ainsi qu'un biprocesseur : Deux CPUs Intel Xeon Gold 5218R. Gérant nativement le biprocessing, la station est équipée de Windows Server comme système d'exploitation. Avec ses deux processeurs, elle possède 80 threads et chaque CPU a une vitesse de 20*2.1GHz.

Finalement, la Dell PowerEdge possède deux disques dur SAS ayant chacun 1.2To de mémoire. Les deux disques sont en duplicata : Un disque est accessible et l'autre est en miroir. Cela permet de faire une sauvegarde en temps réel des données de la station.

Les disques de stockage

Concernant les disques externes, un point a été réalisé sur la semaine du 18/07/2022 sur les disques de stockage utilisés pour la station TV, un tag #HDD a été ouvert pour le besoin: Compte-tenu de l'usage multibase des disques, les labels ont été redéfinis en HDDXX (XX de 01 à 99). L'étiqueteuse sera empruntée auprès de V. Lasnier sur la rentrée de septembre pour relabéliser les disques. Les numéros de série S/N et d'inventaire des disques (internes à Polytech, échange M. Delalandre et S. Beaufils du 202/07/2022 ont également été relevés lorsque connus).

Label	Type	S/N	Inventaire
HDD01	disque WD XBOX 12To	5PGWHELC	NA
HDD02	disque WD XBOX 12To	5PJAMSSD	NA
HDD03	disque WD XBOX 12To	5PJAMOSF	2019006184

Les répartitions suivantes des disques HDD01 et HDD02 ont été relevées, il a été noté que les disques disque WD XBOX 12 To avaient une capacité effective de 11 TiB (capacité binaire vs. décimale).

HDD01 :	PVCD dataset (6.5 TB)		free(4.5TB)
HDD02 :	social TV dataset (1.5 TB)	Fact-Checking (8 TB)	free (1.5 TB)

Un incident s'est produit sur le disque HDD03 le 18/07/2022 (chute du véhicule de J. Courne). Après test auprès d'un technicien extérieur le 18/07/2022 (J. Courne en charge) et du service informatique (S. Beaufils) le 19/07/2022, le disque a été déclaré défectueux. En accord avec S. Beaufils, une procédure SAV a été engagé auprès LDLC. Un ticket Pal n°62952 a été ouvert le 21/07/2022 avec dépôt du bon de commande pour engagement de la demande. Il a été noté que, même si la garantie était toujours valide, compte-tenu de la responsabilité LIFAT/Polytech sur l'endommagement du disque le changement / réparation semblait peu probable. A minima, la coque pourra être ré-utilisée en prévision du retour SAV du disque Exos X16 (cf. 3, ticket Pal n°62956).

A la suite d'un échange avec D. Conte le 20/7, il a été évoqué que du budget investissement restait au Di. Ce budget peut être utilisé sous contrainte d'un achat d'un minimum de 500€ (TTC ou HT, à vérifier) par équipement. Le disque WD My Book 18 To [1] a été identifié (échange M. Delalandre, H. Le du 21/07/2022) comme répondant à la contrainte (premier > 500€ TTC, connectique USB 3.0) chez le fournisseur LDLC pro. Un comparatif des disques WD My Book [1] et WD_Black D10 [2] est donné ci-dessous. En cas d'achat, il a été précisé que ces disques (non robustes aux chocs) ne pourraient faire l'objet de prêt étudiant. Un ticket Pal n°62955 a été ouvert le 21/072022 pour demande de devis.

WD My Book	18To	USB 3.0
sans cache	sans choc	600€ TTC

WD Black D10	12To	USB 3.0
sans cache	avec choc	400€ TTC

Conception de la base STVD Fact-checking

Sélection des programmes

Un fichier csv relatant tous les programmes diffusés sur les 8 chaînes de télévision pendant les 3 mois de captures nommé « programselection.csv » est créer à partir de la base XMLTV. Dans ce fichier les informations suivantes sont données pour chaque programmes :

- Le nom de la chaîne de diffusion
- Le hash code associé au programme
- Le titre du programme
- La durée moyenne en minutes des diffusions du programme sur les 3 mois de capture
- Le nombre de diffusion du programme sur les 3 mois de capture
- La durée totale de toutes les diffusions du programme en minutes
- La durée totale de toutes les diffusions du programme en heures
- La sélection ou non du programme
- Le nombre de diffusion du programme s'il est sélectionné
- La durée en heures du total des diffusions du programme s'il est sélectionné
- La durée en minute du total des diffusions du programme s'il est sélectionné avec les durées de latences

Les différents programmes sont classés par ordre décroissant de leurs durées.

Pour le moment, nous n'avons aucun scripts permettant de faire la sélection d'un programme en fonction de si son contenu est à caractère politique ou non. Ainsi la labélisation de chaque programmes s'est faite par l'expertise de l'utilisateur (1 si sélectionné, 0 sinon). Pour cela plusieurs méthodes ont été mise en œuvre :

- Recherche web d'un programme
- Visionnage des diffusions du programmes
- Lecture des avis utilisateurs

Une fois tous les programmes labélisés, plusieurs statistiques pertinentes ont été calculés :

- En tout 1226 programmes différents ont été capturés, soit 17290 diffusions.
- Le total en heure de tous les programmes capturés (politique ou non) est de 14440h.
- 151 programmes différents (considéré à fait politique) ont été retenu.
- 6730 diffusions ont été retenu.

- La durée de tous les programmes sélectionnés est de 4297 heures, soit 29.84% du total.
- La durée de tous les programmes sélectionnés avec le phénomène de latence est de 6540 heures, soit 45.4% du total.

Ces valeurs ne prennent pas en compte les jours de captures manquants. Cependant elles restent de bonnes références pour contrôler les résultats obtenus à la création de la base.

Structure de la base

Convention de nommage

Afin de caractériser les programmes, les diffusions et les différents fichiers associés aux diffusions, nous avons mis en place une convention de nommage.

Premièrement, les différents répertoires associés aux programmes sont nommés en fonction de leurs hash codes. La méthode ci-dessous montre comment le hash code d'un programme se construit.

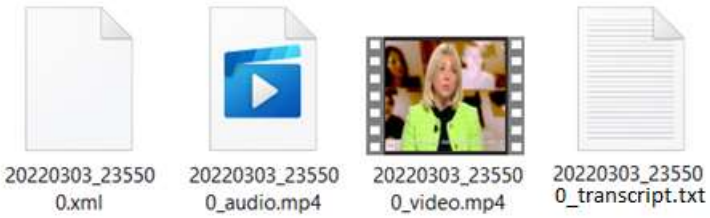
```
def get_hashcode(prog_title, channel_code):

    normalize_title = normalizeText(prog_title)
    concat_str = str(channel_code) + normalize_title
    hashcode = hashlib.sha1(concat_str.encode()).hexdigest()
    return hashcode
```

Le titre du programme normalisé est concaténé avec le code de la chaine de diffusion de programme. Le tout est encodé en sha1 et exprimé en hexadécimal, codé sur 160 bits, 40 caractères (ex : 0a83db5c5d885ef7dbddeb5bc6ef09e640a26b1e).

Deuxièmement, les différents répertoires associés aux diffusions sont nommés en fonction de la date de commencement de la diffusion arrondie à la minute près. Par exemple, le nom du répertoire associé à une diffusion qui commence le 21 août à 23h59 sera « 20220821_235900 ».

Troisièmement, les différents fichiers associés aux diffusions auront comme nommage le timestamp de la diffusion suivi du type de fichier. Par exemple, une vidéo associé à une diffusion qui commence le 21 août à 23h59 aura comme nommage

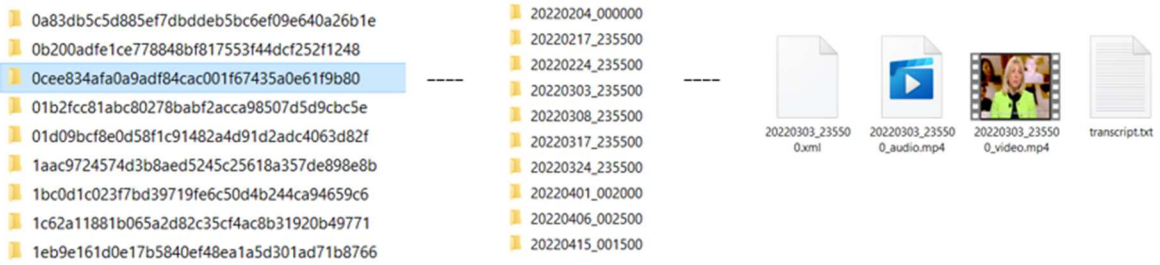
« 20220821_235900_video.mp4 ». La transcription de l'audio aura comme nommage « 20220821_235900_transcript.txt ».



Arborescence

En collaboration avec Mr Delalandre, nous avons pensés la structure suivante :

La base est composée de tous les programmes sélectionnés représentés par des répertoires ayant pour nom les hash codes associés. Dans chacun de ses répertoires sont listées toutes les diffusions représentées par des répertoires ayant pour nom les timestamps associés. Dans chacun de ses répertoires nous pourrons trouver la vidéo, l'audio, le fichier xml et la transcription de l'audio associé à la diffusion. L'image ci-dessous illustre l'arborescence de la base STVD Fact-checking.



Pour le moment nous avons pensé à remplir la base avec ces 4 fichiers pour chaque diffusions. Cependant, il se peut que par la suite nous rajoutions des fichiers pour la reconnaissance facial, pour le Matching de faits avec factoscope etc. Il se peut également que nous enlevions des fichiers, notamment les transcriptions sur lesquels des problèmes de qualité et de création persiste.

Les avantages de cette structure :

Ce type de structure privilégie un nombre de dossiers plus important en contrepartie d'une taille de fichiers plus réduite. Il est fort probable que dans la suite du projet de Fact-checking nous recueillons un grand nombre de données. Il sera donc plus facile de rechercher des informations dans des fichiers de taille réduite (notamment pour le fichier xml).

Dans l'éventualité où une partie du support deviendrait défectueuse, si nous avons qu'un seul gros fichier dessus alors il y a de fortes chances que l'on perde la possibilité d'exploiter tout son contenu. Alors qu'avec de multiples fichiers, on pourra probablement en récupérer une partie (plus ou moins significative).

Extraction des métadonnées

L'extraction des métadonnées est une étape cruciale dans la création de la base. Le fichier csv qui contient ces métadonnées sert à la création de la structure de la base, au parsing xml ainsi qu'à la segmentation des vidéo et des audio.

L'extraction des métadonnées se fait en parcourant la base XMLTV. Tout d'abord, nous recherchons les données XMLTV des chaines qui nous intéressent à savoir TF1, France 2, France 3, Arte, La chaine parlementaire, BFMTV, CNEWS et Franceinfo. De plus, il y a plusieurs condition pour extraire les métadonnées associés à une diffusion.

Premièrement, le programme lié à la diffusion doit être sélectionné dans le fichier « programselection.csv ». Pour le vérifier, nous construisons le hash codes associé à la diffusion présente dans la base XMLTV et comparons avec les hash codes présent dans le fichier « hashcodes.csv ».

Deuxièmement, nous retenons seulement les diffusions qui ont la même date que le fichier xml de la base XMLTV. En effet, chaque fichier xml contient les diffusions du jour mais aussi les diffusions des 15 prochains jours en prévision. En revanche, plus les données XMLTV de la diffusion sont proche du fichier ayant la même date, plus les données sont précises.

Troisièmement, les diffusions retenu durent au moins 5 minutes. Il parait peu intéressant de garder des diffusions avec très peu de contenu.

Quatrièmement, nous prenons seulement les diffusions commençant au plus tôt à 6h08 ou au plus tard à 1h43. En effet, avec le phénomène de latence, que nous expliquerons

dans la section de segmentation des vidéo, nous devons ajouter 8 minutes avant la date de commencement de la diffusion et 12 minutes après la date de fin.

Cinquièmement, nous prenons seulement les diffusions finissant au plus tôt à 6h13 ou au plus tard à 1h48 en raison du même phénomène de latence.

Finalement, pour chaque diffusions, nous prenons le code de la chaine concerné, le hash code, le titre de la diffusion, la date de début, la date de fin et la durée en minute. Nous insérons ces métadonnées dans un fichier csv. L'image ci-dessous illustre une partie des résultats obtenus.

Channel_Code	Hashcode	Title	Start	Stop	Length
c192	833ce1f5c0498df44d8d3e242991e26ff966886e	Journal	20220205200000	20220205204000	40
c192	833ce1f5c0498df44d8d3e242991e26ff966886e	Journal	20220206130000	20220206134000	40
c192	aa161dd20e9c1ec69b7e6c8da62597aa482c9f2d	Sept à huit	20220206181500	20220206194500	90
c192	b972ada14f97fd736cdb57e59e5c54e9e297cfa1	La bataille de l'Elysée	20220207231000	20220208001500	65
c192	aa161dd20e9c1ec69b7e6c8da62597aa482c9f2d	Sept à huit	20220213181500	20220213194500	90
c192	833ce1f5c0498df44d8d3e242991e26ff966886e	Journal	20220213200000	20220213204000	40
c192	b972ada14f97fd736cdb57e59e5c54e9e297cfa1	La bataille de l'Elysée	20220307230000	20220308001500	75
c192	833ce1f5c0498df44d8d3e242991e26ff966886e	Journal	20220308130000	20220308134000	40

Extraction XMLTV

- Pour chaque hash codes de la base STVD Fact-checking nous parcourons les fichiers de la base de données XMLTV.
- Si un élément (diffusion) contenu dans un fichier de la base XMLTV a le même hash code que le répertoire parcouru
- Si la date de début de la diffusion correspond à un répertoire (timestamp) contenu dans le répertoire du hash code
- Prendre toutes l'élément du fichier XMLTV contenant les informations relative à cette diffusion et créer un fichier xml dans le répertoire associé à la bonne diffusion.

Lors d'un run pour créer tous les fichiers xml, je me suis aperçu qu'il manquait des fichiers dans certains dossiers. Ce bug vient du fait qu'un fichier XMLTV a une date. Chaque élément de ce fichier contient les informations des diffusions de cette date mais aussi des 15 jours qui

suivent cette date. Ainsi lorsque les métadonnées étaient créées, le programme prenait quelques fois les informations dans un fichiers XMLTV d'une date précédente. J'avais créé une méthode permettant de rechercher dans les données XMLTV des fichiers avec une date inférieur. Cette méthode fonctionnait bien, cependant il y avait une erreur dans le raisonnement : Pourquoi prendre, pour certaines diffusions, les données XMLTV d'un fichier ayant la même date de début et pour d'autre une date de début différente ? J'ai donc repensé le code en modifiant directement la création du fichier contenant les métadonnées. Ces métadonnées ne prennent que les informations des diffusions ayant la même date de début que le fichier xml concerné.

Extraction audio / vidéo

Pour chaque programmes, l'objectif était de prendre une partie des vidéos et des audios contenant les informations relatives à une diffusion et d'enregistrer la vidéo / l'audio segmenté dans le répertoire associé. Pour segmenter les différents fichiers vidéo / audio, l'outil « ffmpeg » a été utilisé. Cette outils est une collection de logiciels libres destinés au traitement de flux audio ou vidéo (enregistrement, lecture ou conversion d'un format à un autre, segmentation). Il peut être compilé sur la plupart des systèmes.

L'image ci-dessous représente la méthode utilisé pour la segmentation des fichiers vidéo / audio.

```
def crop(start, end, input, output):
    """
    crop a video file

    :param start: The time of the segmentation's start
    :param end: The time of the segmentation's end
    :param input: The video path to segment
    :param output: The path of the segmented video
    """
    cmd = "ffmpeg -hwaccel qsv -c:v h264_qsv -i " + input + " -ss  " + start + " -to " + end + " -c copy " + output
    runbash(cmd)
```

- L'option « -i » permet de donner un fichier en entrée. Dans notre cas, le fichier donnée en entrée est un fichier vidéo / audio issue de 20h issue de la base de capture.

- L'option « -ss » permet de donnée l'heure de début de segmentation du fichier en entrée. L'heure donnée doit être de la forme « hh:mm:ss ».

- L'option « -to » permet de donnée l'heure de fin de segmentation du fichier en entrée. L'heure donnée doit être de la forme « hh:mm:ss ».

- L'option « -c copy » permet d'indiquer que le fichier de sortie est une copie du fichier en entrée. L'argument qui suit est le chemin vers le fichier de sortie qui doit être créer.

Afin d'optimiser le temps d'encodage et de décodage, il existe des accélérateurs. Il existe de nombreux accélérateurs fonctionnant avec diverses cartes graphiques. J'ai choisi l'accélérateur « Intel QuickSync » car cet accélérateur fonctionne correctement avec les processeurs Intel de la DELL PowerEdge.

Les segmentations sont faites sur les captures vidéo et audio de 20h et le choix des heures de début et de fin de segmentations ont été décidé avec le fichier « meta_data.csv » qui contient les informations essentielles sur les diffusions.

De plus, nous devons prendre en compte le phénomène de latence. Ce phénomène vise à éliminer les faux négatifs. Ainsi ne voulant surtout pas perdre de contenu, nous rajoutons, pour chaque segmentations, quelques minutes avant l'heure de commencement et quelques minutes après l'heure de fin de la diffusion. La latence a été calculé par Van Hao Le sur des milliers d'heures de diffusions et il en ressort qu'il faut rajouter 8 minutes avant le début et rajouter 12 minutes après la fin de la diffusion pour être sûre de ne pas perdre de contenu.

Un point important dans la segmentation des vidéo / audio est d'identifier la date qui composera la nom du fichier vidéo / audio de sortie. En effet un fichier vidéo / audio pris en entrée commence à 6h00 à la date du nom du fichier et fini à 2h00 le lendemain. Ainsi plusieurs cas ont été identifiés :

- Si la segmentation vidéo / audio commence avant 0h08, alors le fichier de sortie aura la même date que le fichier d'entrée.
- Si la segmentation vidéo / audio commence à 0h08 ou après alors le fichier de sortie aura la date du jour d'après.

La segmentation des fichiers audio et vidéo se fait via une parallélisation. Pour ce faire, la fonction « segment_all_video » utilise la librairie « multiprocessing ». La méthode « Pool » permet de définir le nombre de processus travaillant en parallèle. La méthode « starmap » permet de découper l'itérable en un nombre de morceaux qu'elle envoie au pool de processus comme des tâches séparées. Dans notre cas, nous lui passons la méthode « crop » en première argument (voir annexe « Segmentation avec l'outil ffmpeg »). En deuxième argument, nous lui passons une liste de paramètre représentant les différentes variables de segmentations (input, output, start, end). Cette liste est une variable global qui est construite via la fonction « segment vidéo » et est vidée à chaque nouveaux programme traité. L'image ci-dessous illustre la parallélisation de la méthode de segmentation des fichiers audio / vidéo.

```
def segment_all_video(nb_process):
    """
    Segments videos and puts them in the associated timestamp's directory

    :param nb_process: The number of process you want
    """
    for new_list in group_elements(os.listdir(base_enrichie), 1):
        for f in new_list:
            segment_video(f)
        with multiprocessing.Pool(nb_process) as pool:
            pool.starmap(crop, [(i[0], i[1], i[2], i[3]) for i in cmd_list])
        cmd_list.clear()
```

Un fichier csv nommé « segment_video.csv » / « segment_audio.csv » est créé et rempli avec différentes informations sur la segmentation :

- Le hash code du programme associé à la vidéo segmentée.
- Le chemin de la vidéo à segmenter.
- Le chemin de la vidéo segmentée.
- L'heure de début de la segmentation.
- L'heure de fin de la segmentation.

Transcription audio

Une fois l'audio segmenté et placé dans le répertoire associé à la bonne diffusion, la transcription peut être faite. Alexis Korek, un stagiaire ayant travaillé sur la transcription audio, mentionnait deux librairie : « Speech_recognition » et « Vosk ».

Pour utiliser la librairie « Speech_recognition », nous devons avoir une connexion internet. En effet cette librairie se connecte à des serveurs distant comme google ou bien IBM pour utiliser leurs modèles. Ainsi une coupure réseau pourrait entraîner un bug dans la transcription audio. C'est pourquoi la librairie « Vosk » semblait la plus pertinente. La transcription avec cette librairie s'effectue hors ligne. Nous pouvons télécharger un modèle réentraîné sur la page

https://alphacephei.com/vosk/models . J'ai choisi le modèle « vosk-model-fr-0.22 ». Ce modèle est adaptable aux besoins de l'utilisateur et l'équipe « cephei » nous donne la marche à suivre sur la page https://alphacephei.com/vosk/adaptation.

Pour utiliser cette librairie nous devons tout d'abord convertir le fichier audio segmenté en un fichier wav. Un bug apparait si nous convertissons directement le fichier audio mp4 en wav. La solution trouvé est de recréer un sous clip avec les mêmes date de début et de fin que nous enregistrons avec l'extension wav. L'image ci-dessous illustre les dits ci-dessus.

```
def convert_to_wav(input, output):
    """
    Convert a mp4 audio to a wav audio

    :param input: path of the audio.mp4
    :param output: path of the audio.wav
    """
    if not os.path.exists(output):
        clip = moviepy.editor.AudioFileClip(input)
        time_end = time.strftime('%H:%M:%S', time.gmtime(clip.duration))
        new_clip = clip.subclip("00:00:00", time_end)
        new_clip.write_audiofile(output, ffmpeg_params=["-ac", "1"])
```

Nous faisons ensuite la transcription audio. Pour cela deux sous librairie de Vosk sont utilisées : « Kaldirecognizer » et « Model ». La librairie « Kaldirecognizer » est une boite à outils pour traiter les données vocales. Elle sert principalement à la reconnaissance vocale. A

partir de cette librairie nous pouvons charger un modèle pré entrainé via la librairie « Model ». Le résultat de la reconnaissance vocal est stocké dans une chaine de caractère puis enregistrer dans un fichier texte dans le répertoire associé à la bonne diffusion. Finalement, nous supprimons le fichier wav créé temporairement.

Les méthodes permettant la transcription de l'audio fonctionne, cependant, le temps estimé pour faire la transcription de toute la base de Fact-Checking TV est de 22 jours, soit 528h. Les délais de mon stage ne permettaient pas de lancer un run de cette ampleur.

J'ai tenté de paralléliser les méthodes de conversions de l'audio et de transcriptions. La conversion en multithreading s'effectue sans problème. Cependant un problème persiste au niveaux de la transcription. Je n'ai pas réussi à corriger ce problème et, à ce jour, la base de Fact-checking TV ne comporte donc pas les transcriptions.

Finalement nous avons convenu avec Monsieur Delalandre de laisser en suspend la transcription de l'audio. En effet à terme nous voulons pouvoir matcher les différents mots-clés de cette base avec celle de factoscope. Or la qualité des transcriptions ne semble pas convenir : Elle ne reconnait pas les noms d'Homme politiques ou lors d'une interview le bruit joue sur la qualité. Une alternative trouvé à la transcription audio serait de créer l'audio à partir du texte factoscope via la librairie « pyttsx3 ». A partir des fichiers textes et des fichiers audio associés, nous pourrons générer des modèles audio d'entités nommées afin d'entrainer des modèles pour le « audio keyword spotting » et l'appliquer sur les audio de la base STVD Fact-checking.

Archivage

L'objectif de l'archivage est de mettre la base de Fact-Checking TV en ligne et de partager cette ressource. Un point important sur l'archivage est que nous voulons pouvoir télécharger la base par lot de 16Go. Cette taille de lot permet à un utilisateur avec une connexion faible de pouvoir télécharger la base lot par lot.

Nous savons que notre base fait environ 2.1To pour 6540h de vidéo/audio. Nous pouvons estimer qu'après la compression de la base, celle-ci aura une taille d'environ 1.95To. La cible de 16Go, cela nous fera environ 123 fichiers. Nous avons donc convenu avec Mr

Delalandre de faire 8 paquets qui contiendront 15 fichiers. Ces paquets devaient avoir une taille à peu près équivalente : 262Go pour chaque paquets. Le but était donc de répartir les différents programmes dans 8 paquets pour que chaque paquets ait une taille d'environ 262Go.

Pour cela j'ai implémenté trois principales méthodes :

- La première récupère à partir du fichier « meta_data.csv » l'ensemble des hash codes associés aux programmes et leurs nombres total d'heures de diffusions sur la période de capture.

- La deuxième méthode renvoie la combinaison des programmes la plus proche des 262Go pour les 7 premiers paquets et met le reste des programmes dans le dernier paquet.

- Le dernier paquet dépassant grandement les 262Go , la dernière méthode prend les programmes en trop dans le dernier paquet et regarde s'il peut les rajouter dans un des 7 premiers paquets sans trop dépasser les 262Go.

Avec ses trois méthode chaque paquets s'approchaient des 262Go (entre 800 et 900h) mais ce n'était pas optimal. J'ai donc déplacé les derniers paquets à la main pour obtenir une solution convenable et j'ai obtenu le résultat suivant :

	Paquet 1	Paquet 2	Paquet 3	Paquet 4	Paquet 5	Paquet 6	Paquet 7	Paquet 8
Nombre programmes	28	19	27	21	14	17	12	13
volume horaire / h	855,05	854,69	855,69	854,95	854,72	855,64	855,3	856,33

Ces chiffres restent théoriques car ils ne prennent pas en compte les jours manquants mais restent un bon indicateur pour la répartition de la base en différents paquets.

Quant aux nommage des différents paquets, nous avons convenus de la nomination

« stvdfc-partX-fileYY », avec X allant de 1 à 8 représentant les paquets et YY allant de 1 à 15 représentant les fichiers contenus dans les paquets.

Expérimentations

Bilan des données extraites

Nous avons extrait 151 programmes et 6690 diffusions. Le tableau ci-dessous met en évidence les différents volumes extrait de la base.

Avant compression, nous avons obtenu les résultats suivants :

Part1 :	267 597 001 651	bytes	28	hashcodes
Part2 :	267 987 271 519	bytes	19	hashcodes
Part3 :	267 324 456 032	bytes	27	hashcodes
Part4 :	266 426 236 842	bytes	21	hashcodes
Part5 :	263 790 420 068	bytes	14	hashcodes
Part6 :	271 441 260 448	bytes	17	hashcodes
Part7 :	262 217 352 001	bytes	12	hashcodes
part8 :	265 476 999 815	bytes	13	hashcodes
Total:	2 132 260 998 376	bytes	151	hashcodes

Après compression nous obtenons ses résultats :

Part	Duration (h)	Hashcodes	Files	Size (GB)	Link
1	855.0 h	28	16	245.8 GB	download
2	854.7 h	19	16	246.0 GB	download
3	855.7 h	27	16	242.6 GB	download
4	854.9 h	21	16	244.7 GB	download
5	854.7 h	14	16	242.4 GB	download
6	855.6 h	17	16	249.5 GB	download
7	855.3 h	12	16	241.0 GB	download
8	856.3 h	13	16	243.8 GB	download
	6842.2 h	**151**	**128**	**1,956 GB**	

Ecart avec les métadonnées

Selon les statistiques calculés nous devrions avoir une base d'une taille d'environ 1.98To avant compression. Or nous avons une base de l'ordre de 2.1To. Il s'avère que nous avons plus de contenu que prévu. Cela se confirme avec les données de la base compressé. Selon le fichier de sélection des programmes nous devions avoir environ 6500h de vidéos et d'audio. Cependant lorsque nous faisons la sommes des heures de diffusions contenu dans

chaque paquets nous obtenons 6842h. Ces calculs ont été effectué grâce aux données XMLTV. Nous avons donc un écart d'environ 300h inexpliqué. Ces valeurs seront recalculées à partir de la base pour avoir le nombre d'heures exact.

Plusieurs explication sont possibles :

- Mauvaise estimation de la taille de la base (variable non prise en compte)
- Oublie d'un cas particulier dans la segmentation des vidéo / audio

Finalement, nous avions prévu d'obtenir 6730 diffusions, cependant nous en avons obtenu 6690. Il y a donc un écart de 40 diffusions.

Guide d'utilisation des scripts de génération de la base de Fact-checking TV

Renseigner les chemins : « path.py »

Avant de commencer à lancer les scripts, vous devez tout d'abord renseigner les différents chemin dans le fichier « path.py » :

- « base_enrichie » : Le chemin vers la base que vous voulez créer.
- « video_path » : Le chemin de la base contenant les fichiers vidéo à segmenter
- « audio_path » : Le chemin de la base contenant les fichiers audio à segmenter.
- « vosk_model_path » : Le chemin du modèle Vosk. (A télécharger)

Les autres variables ont des chemins déjà attribués. Vous pouvez toujours changer leurs chemins en fonction de vos besoin :

- « project_path » : Le chemin vers le projet de création de la base.
- « meta_data_csv » : Le chemin vers le fichier « meta_data.csv ».
- « programselection_csv » : Le chemin vers le fichier « programselection.csv ».
- « hashcodes_csv » : Le chemin vers le fichier « hashcodes.csv ».
- « db_xmltv » : Le chemin vers la base XMLTV.
- « segment_video_csv » : Le chemin vers le fichier « segment_video.csv ».
- « segment_audio_csv » : Le chemin vers le fichier « segment_audio.csv ».

Le projet nommé Base_STVD_FC comporte un répertoire nommé « csv_files » où tous les fichiers csv seront stockés. Ainsi qu'un répertoire « xmltv-db » contenant la base XMLTV. Enfin un répertoire documentation qui contient les documentation des différents scripts du projet.

Créer les fichiers csv : « create_csv_files.py »

Au début de la création de la base, nous n'avons que le fichier « programselection.csv ». Ce fichier, comportant la sélection des diffusions, est rempli par un utilisateur. Premièrement, vous devez créer un fichier contenant tous les hash codes retenus.

Pour cela vous devez appeler la méthode « create_hashcode_csv() ». Cette méthode créer un fichier csv nommé « hashcodes.csv » et le stocke dans le répertoire « csv_files ».

Deuxièmement vous devez créer le fichier contenant les métadonnées. La méthode « create_metadata_csv() » créer le fichier « meta_data.csv » à partir de la base XMLTV et du fichier « hashcodes.csv » et le stocke dans le répertoire « csv_files ».

Créer la structure de la base : « create_repositories.py »

Une fois les fichiers csv crées, vous devez créer la structure de la base. Pour cela vous appelez la méthode « create_repositories() ». Cette méthode créera les répertoires associés aux programmes et nommés par des hash codes, ainsi que les sous répertoires associés aux diffusions et nommés par des timestamps. Attention pour appeler cette méthode vous devez avoir accès aux fichiers « meta_data.csv » et « hashcodes.csv ». Vous devez aussi avoir accès à la base vidéo et la base audio car cette méthode vérifie si les fichiers audio et vidéo existent avant de créer les répertoires.

Créer les fichiers xml : « create_xml_files.py »

Votre structure de base est maintenant créée. Vous devez désormais la remplir. La méthode « create_all_xml() » crée les fichiers xml associés à chaque diffusions et les stockent dans les différents répertoires de la base.

Segmenter les fichiers vidéo : « video_segmentation.py » / « audio_segmentation.py »

Pour créer les différentes segmentations vidéo, vous devez appeler la méthode « segment_all_video(nb_process) ». Cette méthode fonctionne en mutliprocess et l'argument « nb_process » représente le nombre de processus que vous voulez créer. De plus, cette méthode parcourt le fichier « meta_data.csv » , les fichiers vidéo de la base vidéo pour créer les segmentations et les stocker dans le répertoire associé. Un fichier nommé « segment_video.csv » relatant toutes les segmentations effectuées est stocké dans le dossier « csv_files ».

Le processus de segmentation des fichiers audio est le même. Un fichier nommé « segment_audio.csv » est créé et stocké dans le dossier « csv_files ».

Créer les transcriptions audio (à revoir) : « create_transcripts.py »

Pour créer les transcriptions de l'audio vous devez appeler la méthode « create_all_transcripts() ». Cette méthode va, pour chaque diffusions de la base, créer un fichier wav à partir du fichier audio, créer et stocker le fichier de transcription dans le répertoire associé, puis supprimer le fichier wav. Pour créer la transcription audio, la librairie Vosk est utilisé et vous devez télécharger le modèle dans la langue que vous voulez sur le site « https://alphacephei.com/vosk/models ». Pour le moment, la transcription ne se fait pas en multithreading donc le temps de calcul est important. De plus la qualité des transcriptions ne semble pas convenir pour réussir à matcher avec la base factoscope (non-reconnaissance des noms d'Hommes politiques). Ainsi cette partie est à revoir.

Travaux annexes

Extraction des entités nommées

Le but de la création de la base STVD Fact-checking est en autre de permettre d'associé une diffusion à un ou plusieurs faits de la base factoscope. Pour cela nous devons passer par l'extraction des mots clés sur les deux bases. C'est ce que permet « Unitex/GramLab ». « Unitex/GramLab » est une suite de logiciels libre fondées sur des dictionnaires et des grammaires pour l'analyse de corpus. Le projet « Entity_extraction » contient des méthodes permettant l'extraction des entités nommés pour la base Factoscope et pour la base STVD Fact-checking.

Entités nommées de la base STVD Fact-checking

Les méthodes d'extraction des entités nommées de la base STVD Fact-checking se trouvent dans le sous répertoire « stvd » du projet « Entity_extraction ». Les méthodes d'extraction de la base STVD Fact-checking sont les suivantes :

« run_unitex(transcript_path, unitex_path) ». Cette méthode permet de lancer unitex sur un fichier texte. Dans ce cas nous le lançons sur les transcriptions de la base STVD Fact-checking. Le fichier de sortie contient le texte de la transcription en entrée avec des balises qui décrivent les entités nommées reconnues. Par exemple une organisation reconnu aura comme balise « <org> », une date aura la balise « <date> », une géolocalisation aura la balise « <geoFeat> », etc. L'image ci-dessous nous donne un aperçu d'une sortie unitex sur une transcription.

<p><s>Invité sur <persName><surname>RTL</surname></persName> <date>le 2 avril 2022</date>, le <ro
<measure type="duration"><unit>années</unit></measure> à gérer les demandes d'<org type="local">as
ont le fait d'<nationality>étrangers</nationality>, 63 % des agressions sexuelles, 66 % des vols avec violen
lles contre les femmes sont commises par une personne connue de la victime, son conjoint ou son ex-con
és dans le scandale des Pandora Papers <date>en 2021</date>.</s> <s>Volodymyr Zelensky détenait une p
ersName> a vérifié cette affirmation.</s> <s>Les <roleName type="office">journalistes</roleName> s'app
andidat</roleName> à la présidentielle, invité de l'émission " <orgName>Elysée</orgName> 2022 " <date
i évoque le chiffre de " 300 000 personnes privées de domicile fixe ".</s> <s>Les propos de <persName><fo
de surveillance de la qualité de l'air en <placeName>Ile-de-France</placeName> et dans le Grand Est.</s>
/persName>, a comparé les tarifs affichés du litre d'essence en <placeName>Espagne</placeName> avec
me> des <roleName type="office">constructeurs</roleName> <placeName>européens</placeName> d'au
<persName><roleName type="office">journalistes</roleName> <nameLink>de</nameLink> <surname>
/roleName> <date>en 2019</date>.</s> <s>En <persName><surname>France</surname></persName>, o
".</s> <s><persName><forename>France</forename> <surname>Info</surname></persName> analyse l
surname>État</surname></persName>.</s> <s>Ainsi, le droit des <nationality>étrangers</nationality>
raites (COR).</s> <s>Pour évaluer la soutenabilité du système à moyen et long terme, le COR observe l'év
ne</surname></persName> de 2014, qui ne sera pleinement mise en œuvre qu'<date>en 2035</date>.</s>

L'argument « transcript_path » correspond au chemin du fichier de transcription de la base STVD Fact-checking et « unitex_path » le chemin du fichier unitex en sortie. Les différents chemin ne doivent comporter ni d'espaces, ni d'accent ou de symboles pour des raisons techniques liés à unitex. De plus le fichier de transcription donné en entrée ne doit pas comporter de ponctuation.

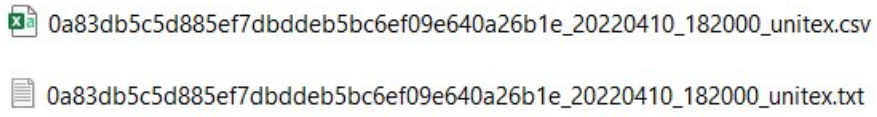
« normalize_transcript(input, output) » . Cette méthode prend un fichier texte en entrée (input), supprime les ponctuation du texte et écrit le résultat dans un autre fichier texte (output).

« unitex_to_csv(unitex_text_path, unitex_csv_path) ». Le fichier texte brute en sortie d'unitex ne permet pas de pouvoir exploiter les entités nommées repérées. Ainsi lister toutes ces entités nommées dans un fichier csv permet d'exploiter ces données pour pouvoir, par la suite, les matcher avec la base Factoscope. Cette méthode permet donc d'écrire dans un fichier csv toutes les entités nommées repérées à partir d'un fichier texte unitex. L'argument « unitex_text_path » est le chemin du fichier texte unitex d'entrée dont nous voulons lister les entités nommées. L'argument « unitex_csv_path » est le chemin fichier csv de sortie listant toutes les entités nommées reconnu dans le fichier d'entrée.

nom	valeur	
date	le 2 avril 2022	
roleName	candidat	
forename	Éric	
placeName	Japon	
org	asile	
org	équipe	
orgName	ministère	
date	depuis 2010	
date	2020	
orgName	autorités	
unit	années	
roleName	journalistes	
orgName	Commissariat aux Réfugiés	
geogFeat	île	
placeName	Syrie	
placeName	Libye	
forename	Marine	
roleName	candidate	
orgName	Elysée	

L'image ci-contre illustre du fichier de sortie csv. Nous avons deux colonnes : « nom » représente le nom de l'entité nommée reconnu et « valeur » représente la valeur de l'entité nommée reconnu.

« create_all_unitex() ». Cette méthode crée, pour chaque fichier de transcription de la base STVD Fact-checking , un fichier texte unitex et le fichier csv associé. Le fichier texte unitex est stocker dans le dossier « unitex_text » et le fichier csv associé est stocker dans le dossier « unitex_csv ». Afin de pouvoir reconnaitre à quel diffusion sont associé les différents fichiers une convention de nommage a été mise en place. Les fichier unitex texte et csv contiennent les hash codes et les timestamps de la diffusion associé.



Entités nommées de la base Factoscope

Tout d'abord nous devons extraire les données de la base Factoscope. La méthode « extract_factoscope(factoscope_database, attribut) » permet d'extraire les données associé à un attribut de la base Factoscope. « factoscope_database » représente le chemin de la base factoscope et « attribut » le nom de l'attribut dont nous voulons extraire les données. Ce dernier argument peut prendre les valeurs « id », « factoURL », « auteur », « date », « titre », « veracite », « mots-clefs », « texteDescriptif » et « sourcesURL ». Les données extraites sont stockées dans un fichier texte. Ce fichier texte a comme nom l'attribut extrait (auteur.txt, titre.txt, etc.) et est stocker dans le dossier « text_files ».

Les méthodes « run_unitex(facto_path) » et « unitex_to_csv(unitex_text_path) » sont les mêmes que pour l'extraction de la base STVD Fact-checking à l'exception des fichiers de sorties qui sont respectivement « factoscope_unitex.txt » et « factoscope_unitex.csv ». Le fichier texte est stocker dans le dossier « unitex_text » et le fichier csv dans le dossier « unitex_csv ».

La méthode « factoscope_unitex_csv() » utilise pour toutes les méthodes précédentes pour créer le fichier csv contenant les entités nommées à partir de la base factoscope. Premièrement, cette méthode extrait le texte descriptif de la base factoscope via la méthode « extract_factoscope ». Extraire seulement cette attribut semble pertinent car il contient déjà les informations de dates, d'auteurs, de titre, etc. Ensuite un fichier texte temporaire est créer. Ce fichier contient les informations du texte descriptif sans les sauts de ligne et la ponctuation. Finalement les fichiers texte et csv unitex sont créer via les méthodes « run_unitex » et « unitex_to_csv » et le fichier temporaire est supprimé.

Création de la base Vidéo Tagging

La base vidéo tagging a pour objectif de permettre l'extraction de méta données sur des vidéos grâce à l'analyse d'images. Ainsi cette base ne comporte que des vidéos. Ainsi la base vidéo tagging a la même structure que la base STVD Fact-checking. Seul les fichiers qu'elle comporte divergent. Le fait d'extraire des méta données supplémentaire sur des vidéo peut notamment servir pour compléter la base STVD Fact-checking mais aussi pour créer une base dans le domaine du patrimoine. Le code de génération de la base Vidéo Tagging s'apparente fortement au code de génération de la base STVD Fact-Checking. Les quelques points divergeant seront expliqués dans la suite. Le projet python de création de cette base se nomme « Base_Video_Tagging » et se trouve dans les livrables du stage. La création de la base n'a pas pu aboutir pour des contraintes de temps, cependant les codes de génération de cette base sont achevés.

Données d'entrées et base de capture

La sélection des programmes est constituée de 3 fichiers csv nommés « CSVVideo1.csv », « CSVVideo2.csv » et « CSVVideo3.csv ». Un fichier correspond à la sélection des programmes pour une semaine.

17/06 - 24/06 semaine 1
24/06 - 01/07 semaine 2
01/07 - 08/07 semaine 3
08/07 - 15/07 semaine 4

La dernière semaine n'a pas été traité pour des contraintes de temps. De plus, la base xmltv s'étend du 17/06/2022 au 15/07/2022. Finalement, la base vidéo comprend les captures vidéo sur 3 chaines de télévision : France 2, France 3 et France 5.

Création des méta données

La création des méta données se fait de la même façon que celle pour la base STVD Fact-checking. Le seul point qui diverge est que plus qu'enlever les faux négatifs, nous voulons enlever les faux positifs. Ainsi nous enlevons 8 minutes au début de chaque diffusions et 12 minutes à la fin. De ce fait, chaque diffusion doit retenu dure au moins 25 minutes pour avoir au moins 5 minutes de contenu (25 – 12 – 8 = 5).

De plus nous prenons seulement les diffusions commençant au plus tôt à 5h52 ou au plus tard à 1h47. Finalement, nous prenons seulement les diffusions finissant au plus tôt à 6h17 ou au plus tard à 2h12 en raison du même phénomène de latence.

Références

- Alexis KOREK : Fact Checking : Détection de fake news à partir de flux vidéo: Projet Recherche & Développement, Ecole Polytechnique de l'Université de Tours, Tours, France,2021-2022.
- Frederic Rayar, Mathieu Delalandre, Van-Hao Le : A large-scale TV video and metadata database for French political content analysis and fact-checking
- Mathieu Delalandre, Van-Hao Le, Donatello Conte : Une large base de données pour la détection de segments de vidéos TV
- Mathieu Delalandre : A Workstation for Real-Time Processing of Multi-Channel TV

Stage d'analyse de données TV

Résumé :

L'objectif principal de ce stage est de créer une base de données pour des données multimédias à caractère politique collectées par une station TV. Pour cela je devais analyser et segmenter des fichiers audio/vidéo à partir de méta données. De plus je devais parser des données XMLTV à partir de ces méta données. Je devais aussi faire la transcription audio afin de détecter des mots-clés et de les mettre en relation avec une base de données de faits vérifiés. Finalement je devais étendre mon travail produit pour la création d'une base de faits politique pour générer une base de données sur le domaine du patrimoine.

Mots-clés :

Données multimédias, fact-checking, faits politiques, domaine du patrimoine, XMLTV

Abstract :

The main goal of this internship is to create a multimedia database for political data collected on the TV station. In this way, I had to analyse and segment audio / video files from metadata. More, I had to parse XMLTV data from the same source. An other task was to create audio transcripts in order to detect keywords et match them with a verified facts database. Finally, I had to extend my work in order to create a patrimony database.