POLYTECH TOURS

2022-2023

Projet SI : Station TV

Etude des programmes télévisés des élections présidentielles françaises de 2022

Professeurs encadrant :

DELALANDRE Mathieu

Compte rendu présenté par :

DELOFFRE Diego

SENGER Maxime

LOIT Joris

CHOKRI Marouane

MOHAMED EWE Ely Cheikh

Introduction **3**
Contexte 3
Présentation du sujet 3
Objectifs 3

Conception et résultats **4**
Présentation générale 4
Génération des mots clés 4
Transcription audio 5
Conversion de l'audio en texte 5
Speech to Text 6
Lemmatisation et entités nommées 6
Génération du résultat JSON 7
Structure finale 7
Vérification des conversions 7
String matching 8
Matching global 8
Matching par fichier 9
Outils d'analyse statistique 9

Conclusion **11**

Introduction

Contexte

Ce projet est en lien avec la station TV développée depuis 2018 par l'école Polytech Tours et le laboratoire LIFAT. La station TV est une plateforme de calcul parallèle pour le traitement automatique des chaînes de télévision de la TNT. La station est équipée deux machines: DELL T7610 pour la capture vidéo et DELL PowerEdge T640 pour le traitement temps-réel de flux vidéo. L'objectif de la station est le déploiement d'applications de traitement d'images, de vidéos et d'intelligence artificielle du laboratoire LIFAT. En dehors du développement de la station elle-même, le projet station TV recouvre aujourd'hui de multiples services pour la capture smart de vidéos et d'images jpg, la détection de publicités et de génériques TV, le scraping Web de données TV, le fact-checking et l'annotation automatique texte/vidéo.

Présentation du sujet

Ce projet est sur la capture vidéo, la station a déjà été utilisée pour de larges campagnes de capture (e.g. plusieurs mois en continu, 8 chaînes en simultané, 20h/24h). Récemment, une base de données sur le journalisme de données a été produite à l'occasion des élections présidentielles françaises 2022. Cette base se présente comme la plus large base de données audio/vidéo à l'international sur le journalisme de données (avec plus de 6500 heures de programmes politiques collecté sur la période de février à mai 2022 et couvrant 150 émissions différentes). En complément des données audio/vidéo, la base de données est fournie avec un résultat de scraping du site Factoscope offrant différents contenus textuels et métadonnées pour le journalisme de données (liste d'hommes/femmes politiques, articles, sujets d'actualités, etc.). Cette base a été rendue publique pour les besoins de la recherche en vision par ordinateur, sur le traitement automatique du langage et en journalisme de données.

Objectifs

On se propose dans ce projet de mettre en place un outil d'analyse de cette base pour le spotting de données textuelles et audio. En effet, une analyse fine de ces données pourrait être conduite comme par exemple pour la recherche des citations des noms des personnalités politiques, l'évocation de termes en lien avec des sujets/thématiques des élections, la recherche d'idiomes et idiolectes pour la détection et personnalités et/ou groupes politiques, etc. La difficulté d'une telle analyse réside dans la reconnaissance audio. En effet, si les outils de reconnaissance vocale s'avèrent aujourd'hui assez matures pour le traitement de la majorité des données audio telles que celles de la TV (i.e. dans un contexte de parole contrôlée), ils présentent des limitations importantes en cas de noms propres et termes spécifiques. Le recours à des outils de transcription semble exclure ici compte-tenu de la volumétrie. Pour pallier

cette difficulté, nous proposons dans ce projet une approche en deux étapes "gros grain / grain fin". Plusieurs points pourront être abordés :

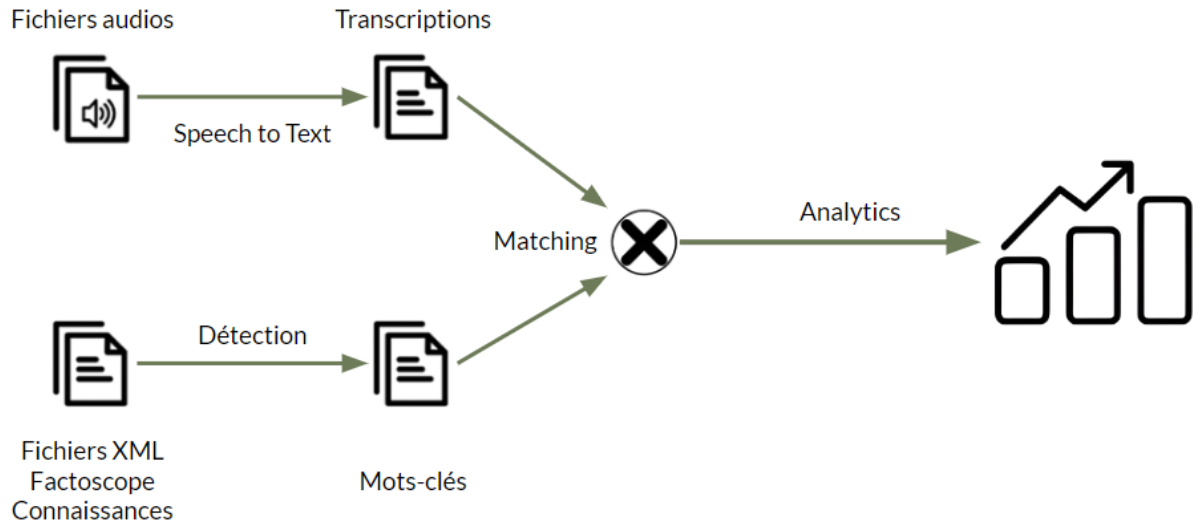
- Extraction de termes et entités nommées spécifiques de la base Factoscope
- Reconnaissance audio de la base à l'aide d'outils automatiques
- Evaluation de la qualité de transcription
- Spotting de données textuelles par "string matching"
- Construction d'outils d'analyse statistiques

Conception et résultats

Présentation générale

Nous avons choisi d'utiliser Python pour plusieurs raisons. Tout d'abord, nous sommes tous compétents et expérimentés sur ce langage. De plus, Python dispose de nombreuses librairies dédiées à l'analyse de données audio, textuelles et est propice à l'utilisation de modèles d'IA.

Le projet se divise en plusieurs étapes à réaliser définies plus tôt, voici un schéma résumant simplement l'ordre d'exécution des différents programmes.



Génération des mots clés

La première étape pour analyser cette base de données est de déterminer une liste de mots clés. Cette liste est composée de mots clés caractéristiques et pertinents concernant le sujet de la base de données, à savoir la politique, ainsi que d'entités nommées identifiant des personnes ou des lieux spécifiques comme les candidats à l'élection par exemple. Cette base de mots-clés nous servira à faire correspondre ces mots dans les données audio transcrites afin de réaliser des analyses statistiques.

Pour identifier ces mots clés, on dispose de plusieurs sources de données. Tout d'abord, chaque vidéo enregistrée dans la base de données est accompagnée d'un fichier XML contenant diverses informations sur la vidéo, dont une description de son contenu. Cette description nous servira à identifier certains mots-clés.

Ensuite, nous avons à notre disposition les données du site Factoscope sous la forme d'un fichier XML. Factoscope est un site d'agrégation de données politiques avec du fact-checking qui contient donc des éléments d'informations utiles pour identifier des mots-clés dans le cadre de notre projet. Enfin, cette base de mots-clés est à compléter et trier selon nos connaissances personnelles afin de conserver seulement les mots les plus pertinents à analyser.

Lors de l'analyse des fichiers XML associés aux vidéos et de la base Factoscope, nous parcourons tous les fichiers en comptant chaque occurrence de chaque mot. Un prétraitement est effectué afin de mettre tous les caractères en minuscules ou encore enlever les ponctuations et les stop-words. Les stop-words sont une liste de mots définis qui contiennent peu d'informations et sont donc peu pertinents (ex ; "Le", "la", etc). Ensuite, lors de l'analyse, nous utilisons également des lemmes qui composent la base lexicale de tous les mots d'une même famille.

Par exemple, les mots "politique", "politisé", "politiques" vont être comptés ensemble et regroupés sous le mot-clé "politique". Cela permet de mieux repérer les thèmes généraux de la base de données. Grâce à cela, nous pouvons réduire le nombre de mots-clés et ainsi accélérer les traitements ultérieurs.

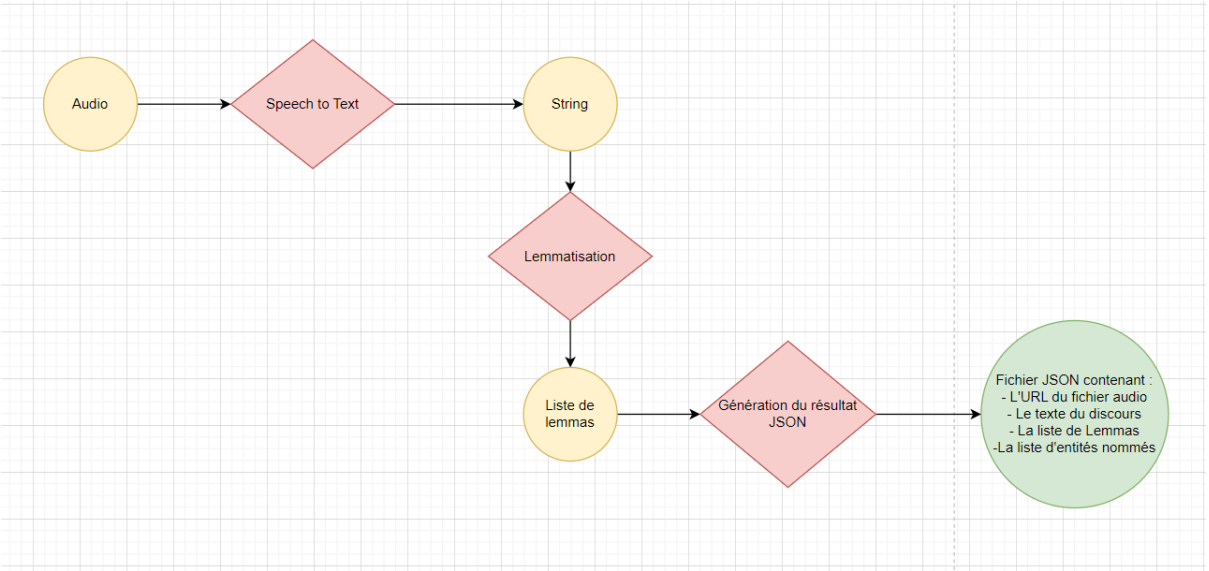
Une fois tous ces traitements effectués, nous pouvons donc constituer une liste de mots-clés en collectant toutes les chaînes de caractères récupérées.

Transcription audio

Conversion de l'audio en texte

La conversion de l'audio en texte est l'une des phases les plus importantes pour pouvoir étudier les chaînes de caractères d'un audio et éventuellement générer des mots clés qui nous faciliteront l'analyse du contenu lexical d'un discours politique.

Pour réussir une telle tâche, nous allons suivre le schéma suivant :



Speech to Text

La première étape consiste à développer un script permettant de convertir un fichier audio en entrée en un fichier textuel en sortie correspondant au script du fichier audio. Une telle opération peut paraître compliquée au début, mais Python propose de nombreuses solutions pour réaliser une telle démarche. Dans notre cas, nous avons choisi d'utiliser Whisper qui est un modèle de reconnaissance vocale à usage général, développé par la société OpenAI. Il est formé sur un vaste ensemble de données audio diverses et est également un modèle multitâche qui peut effectuer une reconnaissance vocale multilingue ainsi que de la traduction vocale et de l'identification de langue.

Il suffit seulement d'avoir une version de Python 3 et d'installer quelques packages comme ffmpeg-python pour la lecture des fichiers audio. Ainsi, on peut facilement obtenir le script d'un fichier audio sous la forme d'une chaîne de caractères.

La plus grande difficulté de cette étape réside dans le temps de calcul. En effet, Whisper est un des outils gratuits les plus performants du marché actuel. Il est un des rares à proposer une exécution locale plutôt que via des API, ce qui nous permet donc de profiter pleinement de la puissance de calcul de la machine à notre disposition. Même avec une machine puissante, un outil performant et du multi-threading, la transcription des 6500 heures de données nous a pris environ 3 semaines. En effet, nous avons effectué un premier relevé au bout de 4 jours, nous avions parcouru 1100 fichiers sur 6200. Les fichiers n'étant pas tous de la même durée et la machine étant utilisée à d'autres fins durant l'exécution du programme, il est difficile d'effectuer un calcul plus précis.

Lemmatisation et entités nommées

La lemmatisation, déjà définie plus tôt dans ce rapport, sera réalisée grâce à la librairie Spacy. Cette librairie nous a permis d'obtenir une liste de lemmes à partir des données

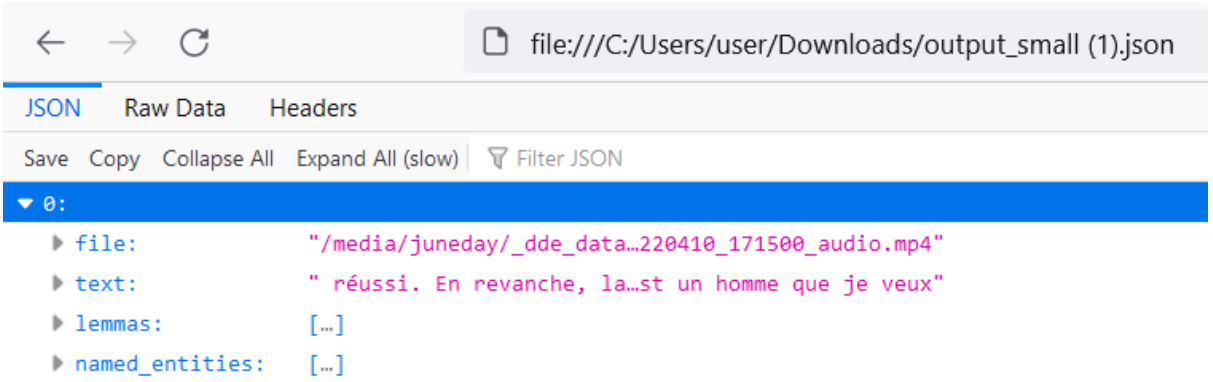
transcrites qui nous sera utile dans des phases ultérieures. En effet, il sera donc possible de rechercher certains mots-clés de manière exacte dans le texte de la transcription, ou bien dans une liste de lemmes générés à partir de la transcription. Cette deuxième méthode permet d'identifier des thèmes plutôt que des mots précis.

En NLP (Traitement Automatique du Langage Naturel), une entité nommée est souvent utilisée pour décrire un nom propre, tel que le nom d'une personne, d'une organisation, d'une ville, d'un pays, etc. Nous parcourons donc le texte transcrit à la recherche de toutes les entités nommées qui nous seront également utiles dans d'autres phases du projet.

Génération du résultat JSON

Une fois toutes ces étapes réalisées, nous disposons de différentes données que nous stockons dans un objet JSON contenant :

- L'URL de l'audio
- Le texte correspondant en format de chaîne de caractère
- La liste de lemmes correspondante
- Les entitées nommées rencontrées



Structure finale

Initialement, nous avons stocké tous les objets JSON dans un même fichier. Cependant, après la demande de M. Delalandre, nous avons séparé chaque objet dans un fichier à part. Ce fichier sera placé dans une architecture similaire à la base de données sur laquelle nous avons travaillé avec pour convention de nommage *'timestamp_transcript.json'* Cette seconde base ne contiendra pas de fichiers audio ou vidéo et sera donc beaucoup plus légère pour les futures personnes travaillant sur ces données textuelles.

Vérification des conversions

Pour vérifier que la partie speech to text était précise, nous avons transcrit à la main différents passages des fichiers audio afin de pouvoir calculer un taux de précision. Whisper propose plusieurs niveaux de taille de modèle pour réaliser la transcription. Plus le modèle est grand et plus le résultat est précis, en revanche la vitesse augmente

également en fonction de la taille du modèle utilisé. Nous avons donc utilisé le modèle Tiny de Whisper, à savoir le plus petit.

Nous avons fait ce choix car c'était probablement le meilleur compromis entre la précision et la rapidité. En effet, le taux de précision était de 88% pour le modèle Tiny. A titre d'exemple, le modèle Base nous permettait d'obtenir un taux de précision de 90%, mais une exécution deux fois plus longue comme le montre ce tableau présent dans la documentation de Whisper.

Size	Parameters	English-only model	Multilingual model	Required VRAM	Relative speed
tiny	39 M	`tiny.en`	`tiny`	~1 GB	~32x
base	74 M	`base.en`	`base`	~1 GB	~16x
small	244 M	`small.en`	`small`	~2 GB	~6x
medium	769 M	`medium.en`	`medium`	~5 GB	~2x
large	1550 M	N/A	`large`	~10 GB	1x

Le calcul du taux de similarité a été effectué avec la bibliothèque FuzzyWuzzy en utilisant la fonction fuzz.ratio() qui prend en argument deux chaînes de caractères et qui utilise l'algorithme de Levenshtein pour calculer la similarité entre ces chaînes. L'algorithme de Levenshtein mesure la distance entre deux chaînes de caractères en comptant le nombre minimum d'opérations (insertions, suppressions et substitutions de caractères) nécessaires pour transformer une chaîne en une autre.

String matching

Le string matching est une technique utilisée pour trouver toutes les occurrences d'une chaîne de caractères dans une autre chaîne de caractères plus grande.

Nous avons utilisé le string matching afin de compter les occurrences des mots clés dans toutes les transcriptions audio récupérées via l'étape de la transcription audio. Pour ce faire, nous avons utilisé la librairie Python *FuzzyWuzzy* qui fournit des fonctions pour effectuer des opérations de matching de chaînes approximatives. Elle utilise la distance de Levenshtein pour calculer la similarité entre deux chaînes de caractères.

Matching global

Nous avons d'abord réalisé un fichier global contenant pour toutes les transcriptions, le nombre d'occurrences de chaque mot clé. Pour cela, nous avons utilisé le fichier *keywords.json* qui contient la liste des mots clés définie précédemment. Nous utilisons également le fichier *transcripts.json* qui contient les transcriptions de tous les fichiers audio.

Ensuite, nous avons utilisé une boucle pour parcourir tous les fichiers de transcription et pour chaque fichier, nous avons créé un dictionnaire qui va contenir le nom du fichier et les occurrences des mots clés. Pour cela, nous avons utilisé la fonction *extractBests* de

FuzzyWuzzy avec une correspondance minimal de 80%. Cette fonction retourne une liste des occurrences dans l'ordre d'apparition. La longueur de cette liste correspond donc au nombre de fois où le mot apparaît.

Une fois le traitement de données terminé, nous enregistrons les résultats suivants au format *json* :

#idkeyword; keyword; position; precision; occurrency_number

Matching par fichier

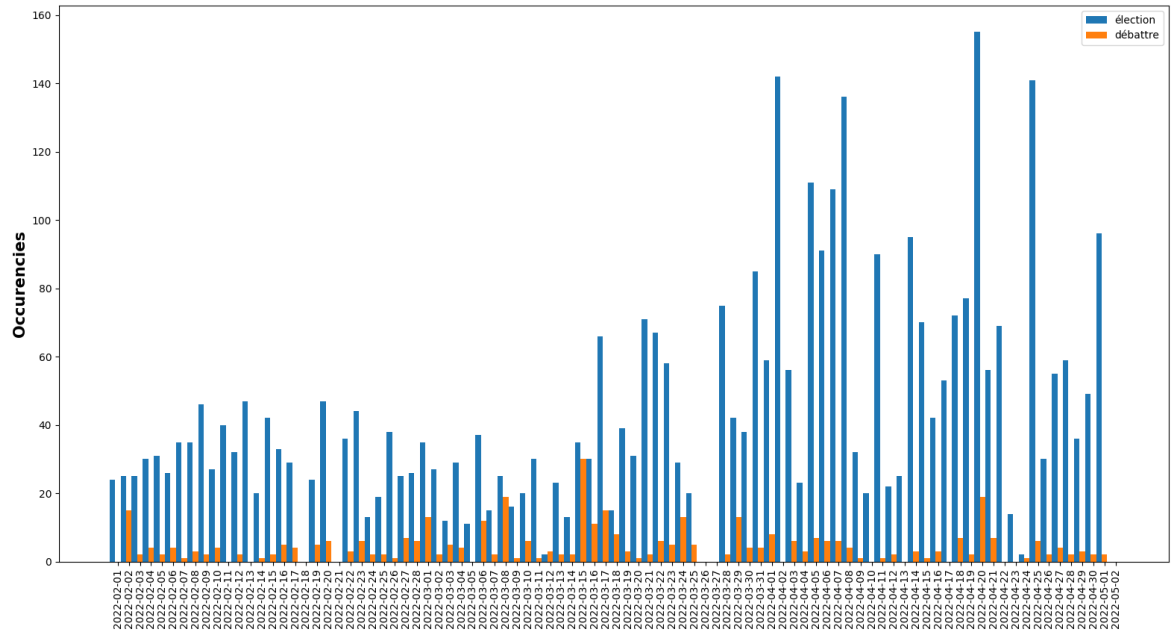
Nous avons également réalisé un fichier de string matching pour chaque transcription unique. Ce fichier sera placé dans la nouvelle architecture créée pour les fichiers de transcriptions avec pour convention de nommage *'timestamp_spot.json'*. Les mêmes outils et techniques ont été utilisés pour réaliser ces fichiers au format json et les résultats sont enregistrés sous la même forme.

Outils d'analyse statistique

L'objectif final de ce projet est de pouvoir analyser le contenu de cette base de données. Une fois les données audio transcrites en texte, la base de mots clés définies et les fichiers de string matching réalisés, il ne reste donc plus qu'à développer des outils d'analyse.

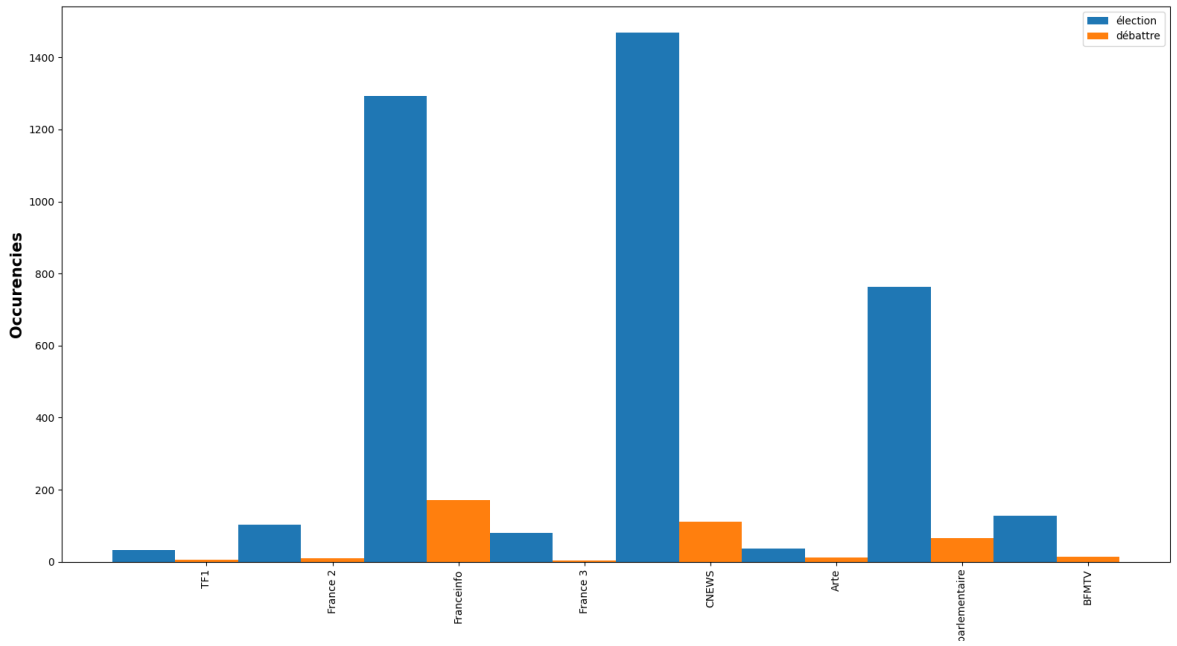
Cette étape commence par une phase de réflexion. En effet, il faut réfléchir à ce qu'il serait intéressant d'analyser à partir des données actuelles.

Premièrement, nous avons simplement pensé à produire un graphique comparant le nombre d'occurrences par jour de chaque mot-clé afin d'avoir une vue d'ensemble de l'évolution de chaque terme au cours des élections. Nous avons donc développé une fonction permettant de générer ce graphique à partir de n mots clés.



Histogramme du nombre d'occurrences de 2 termes en fonction de la date

Ensuite, disposant de diverses informations sur les émissions comme la chaîne de diffusion, la catégorie ou l'heure de diffusion. Nous avons décidé de produire des graphiques en fonction de chaque catégorie. Ainsi, nous sommes capables, pour N mots sélectionnés, de déterminer à quelle heure, sur quelle chaîne ou encore dans quelle catégorie d'émission ils ont le plus été évoqués. Cela peut permettre d'identifier des tendances en fonction des critères définis.



Histogramme du nombre d'occurrences entre 2 termes en fonction de la chaîne de télévision entre février et mai 2022

Conclusion

Ce projet fut très intéressant à de nombreux niveaux. Tout d'abord, travailler sur une telle base de données, unique au monde de par sa taille, est un privilège. Ensuite, travailler avec de tels volumes de données nous a permis d'élargir notre manière de penser et nous a forcés à prendre en compte des paramètres supplémentaires comme le temps de calcul ou la scalabilité (mise à l'échelle). Ces problématiques sont très actuelles et ne feront que se renforcer dans le futur donc il est utile d'y être confronté dès maintenant. Enfin, les outils et algorithmes de traitement de données audio sont nouveaux pour nous et il a également été intéressant de les découvrir.

Globalement, nous sommes satisfaits du travail produit puisque chaque étape a été réalisée et il est donc possible d'obtenir une analyse statistique des données, ce qui était l'objectif final.

Cependant, certains aspects sont améliorables. Premièrement, nous n'avons pas forcément les compétences ou simplement une appétence pour la politique suffisante pour assurer la pertinence du choix des mots-clés. Nous avons choisi selon ce que nous pensions cohérent mais il se peut que la base de mots-clés soit à retravailler. Deuxièmement, il en est de même pour les analyses finales. Les outils construits ont été faits selon ce que nous pensions qu'il serait intéressant d'analyser mais il se peut qu'une personne trouve ces outils peu utiles. Il serait donc intéressant de demander à un client des idées précises d'outils afin d'augmenter la pertinence des analyses.

Toutefois, au-delà de la pertinence par rapport au sujet. Tous les outils ont été développés dans le but d'être adaptables et améliorables. L'outil de transcription est fonctionnel et a un bon taux de réussite, l'outil de string matching est performant et pertinent et la base de mots clés est toujours améliorable. Quant aux outils d'analyses statistiques permettant la génération de graphes, nous pensons tout de même qu'ils sont utiles et peuvent servir à démontrer le potentiel d'une telle analyse et donc à encourager des projets ultérieurs avec des besoins analytiques plus définis. Ce projet est donc une base solide et stable pour accueillir de nouveaux projets.