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 2021-2022

Enrichissement des métadonnées

Projet station TV

Confidentiel

Entreprise :
Laboratoire LIFAT
60 Rue du plat d'Etain, 37000, Tours, France

Etudiant :
ROUSSEL Angèle
Promo 2020-2023

Tuteur Entreprise :
FRIBURGER Natalie
Maître de conférences

Tuteur académique :
GAUCHER Pierre

Table des matières

1. Le LIFAT

Le LIFAT est le laboratoire d'informatique fondamentale et appliqué de l'université de Tours. Leur objectif est de créer des modèles, des méthodes et des algorithmes, et de fournir des ressources et des logiciels d'extraction de données, d'extraire des connaissances à partir de données par le biais d'interaction homme-machine, et de résoudre des problèmes d'optimisations combinatoire avec pour objectif d'atteindre de bons résultats avec un temps correcte d'exécution.

Le laboratoire est composé de trois groupes de recherches :

- Bases de données et traitement du langage naturel (BdTln)
- Recherche opérationnelle, planification et transport (ROOT)
- Reconnaissance de formes et analyse d'images (RFAI)

Depuis le 1e janvier 2012, le groupe ROOT est associé au CNRS en tant qu'Equipe Recherche Labellisée du CNRS.

Les trois principaux domaines d'activités du laboratoire sont :

- La santé et le handicape avec plusieurs partenariats avec le CHU de Tours et l'INSERM pour une aide technique pour le handicape physique, l'autisme, l'analyse d'image pour une aide au diagnostic, …
- Le Big Data et le calcul intensif avec des problématiques d'infrastructure pour le stockage de données, le cloud, l'extraction, analyse et structuration des données, …
- Les Sciences humaines numérique avec des problématiques liées aux structures de base de données, le scan 3D ou encore la reconnaissance de paternes. Plusieurs partenariats sont en cours avec le CESR et le CITERES.

Le LIFAT a déjà fait plusieurs collaborations académiques et industrielles, que ce soit au niveau national ou international. Avec le Laboratoire d'Informatique Fondamentale d'Orléans (LIFO), il fait parti de la fédération ICVL.

2. Le projet station TV

Le projet station TV est une collaboration entre les différents groupes du laboratoire. La base du projet consiste à capturer l'audio et la vidéo de plusieurs chaînes télé afin de pouvoir faire diverses analyses. Pour le moment, deux ordinateurs sont assignés à ces enregistrements, mais un seul permet de capturer l'audio, et seulement sur 8 chaînes. Le deuxième ordinateur capture sur 24 chaînes. Les problématiques de stockage et de parallèlisation sont donc importantes pour ce projet.

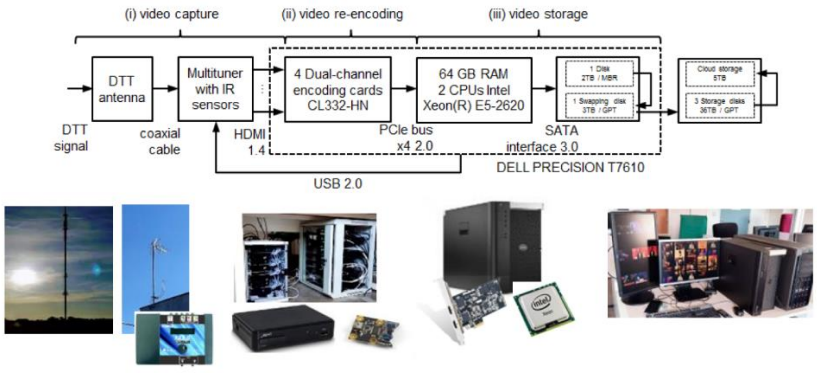

Figure 1

Avec ces enregistrements, plusieurs autres projets en ont découlé. Ainsi, une campagne de capture a été lancée pendant la campagne présidentielle afin de pouvoir faire du fact-checking sur les différentes informations énoncées par les candidats, ou encore un projet sur l'influence de la télévision sur le tourisme par le biais des émissions sur le patrimoine local.

3. Mon objectif pendant ce stage

Pendant ce stage, mon objectif était d'enrichir les métadonnées liées au différents programmes capturés par les stations. Une métadonnée est une donnée servant à décrire une autre donnée. Dans le cas présent, les métadonnées servent à décrire un programme tv, elles peuvent être donc du texte (titre, description, …) ou une image (tiré du programme en lui-même ou utilisé pour décrire le programme par une source extérieur).

Mon objectif était donc de croiser ces différentes métadonnées afin de les enrichir, par exemple, faire en sorte qu'une image d'un volcan dans un programme consacré à la Réunion soit reconnu comme étant le Piton de la Fournaise.

4. XMLTV

Le XMLTV est un fichier XML contenant des informations sur les différents programmes télé de la journée, ainsi qu'une prévision pour la semaine à venir. Il se présente sous la forme suivante :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE tv SYSTEM "xmltv.dtd">
<tv generator-info-url="http://mythtv-fr.org/" generator-info-name="XMLTV" source-data-url="https://api.telerama.fr" source-info-url="https://api.telerama.fr">
    <channel id="C192.api.telerama.fr">
        <display-name>TF1</display-name>
        <icon
    src="https://television.telerama.fr/sites/tr_master/files/sheet_media/tv/500x500/192.png"/>
    </channel>
    …

    <programme channel="C78.api.telerama.fr" stop="20211218165000 +0100" start="20211218164500 +0100">
        <sub-title>La coquette</sub-title>
        <desc lang="fr">Saison:1 - Episode:15 - Un programme court qui lutte contre les stéréotypes sexistes dès le plus jeune âge</desc>
        <date>2021</date>
        <category lang="fr">série d'animation</category>
        <length units="minutes">5</length>
        <icon
    src="https://focus.telerama.fr/1111x625/2021/07/08/a59561fa841bea84dadf1a8d33ce1507c9a9d36f.jpg"/>
        <episode-num system="xmltv_ns">0.14.</episode-num>
        <rating system="CSA">
```

```
                <value>Tout public</value>
            </rating>
        </programme>
        …
</tv>
```

Ces fichiers permettent d'obtenir diverses informations sur les programmes et les chaînes comme le titre, la description, le début et la fin du programme et des images d'illustrations. Ils sont répartis en plusieurs bases : la basique avec 30 chaînes qui a servi à la majorité des expérimentations, celle avec 300 chaînes pour avoir une plus grande volumétrie de données, et la multi-sources qui n'a pas été utilisée. La basique a enregistré des données du 17 décembre 2021 au 8 juillet 2022, et celle avec 300 chaînes du 7 avril 2022 au 27 juin 2022.

La première étape a donc été de parser ces fichiers afin de pouvoir utiliser toutes leurs informations pertinentes dans mon programme. Pour ce faire, j'ai choisi d'utiliser Java car j'avais déjà réalisé auparavant des parseur d'XML dans ce langage. Ce parser SAX a par la suite continué à être modifié pour répondre aux nouvelles problématiques, par exemple, j'avais besoin à un moment d'avoir la liste des programmes diffusés pour une semaine donnée, donc j'ai dû créer un nouveau booléen qui était vrai quand le programme répondait au critère.

Un autre point important du parser était la création d'un hash pour identifier de manière unique un programme. Il est composé du hash titre concaténé au hash de la chaîne afin de pouvoir regrouper ensemble les émissions d'une même chaîne ayant le même titre, tout en distinguant celles des autres chaînes, comme la météo. Ces hash sont obtenus en utilisant la fonction hashCode de Java sur les noms simplifié du programme et de la chaîne pour éviter les problèmes d'erreur de saisie sur des accents par exemple. Pour cela, j'ai utilisé une classe ASCII donné par M.Delalandre. Celle-ci permet de transformer un string en son équivalent avec uniquement des minuscules et des chiffres. Néanmoins, certains caractères n'apparaissaient pas dans cette classe, donc j'ai dû les gérer moi-même, ainsi pour certains caractères comme le '+' qui posaient des soucis car ils peuvent avoir un rôle particulier pour les strings. Après avoir obtenus ces deux hashs, leur concaténation a aussi posé un problème car tous deux peuvent avoir 10 chiffres, le maximum pour un int, donc rassemblés, ils pouvaient faire jusqu'à 20 chiffres, or un long ne peut faire que jusqu'à 19 chiffres, donc j'ai dû utiliser un bigInteger pour tous les hashs des programmes.

5. Eliminer les doublons

Après avoir parser tous les fichiers, on se retrouve avec une liste de plusieurs milliers de programmes, mais certains sont identiques donc on peut les regrouper. Si leur hash est identique, on regroupe ces programmes en actualisant le nombre d'occurrence de ce programme et en concaténant les descriptions si elles sont différentes. En effet, certains programmes possèdent plusieurs fois la même description (même émission, description générique, …), mais cela rend le programme moins intéressant à analyser si la description finale est composée de 80% de descriptions redondantes. Néanmoins, je n'ai pas traité le cas des descriptions presque identiques (une description générique avec le numéro d'épisode qui change par exemple) car on a considéré ceci comme étant négligeable dans mon étude.

6. Statistiques sur ces programmes

Après avoir les doublons, on se retrouve avec 7082 programmes uniques dont 6632 avec une description non nulle.

Voici quelques autres statistiques tirées de cette liste de programmes :

- Moyenne de mot par description : 327,646286
- Moyenne d'émissions par programme : 23,723948
- Nombre de programmes sans description : 450
- Nombre de programmes avec plus de 300 mots de description : 772
- Moyenne de mots par description non nulle : 349,878016
- Nombre de mots total : 2320391
- Nombre d'émissions total : 168013
- Moyenne de mot par émissions : 13,8107825
- Équivalent en nombre de romans : 29,0048875
- Moyenne de mots par description de plus de 300 mots : 2336,24741

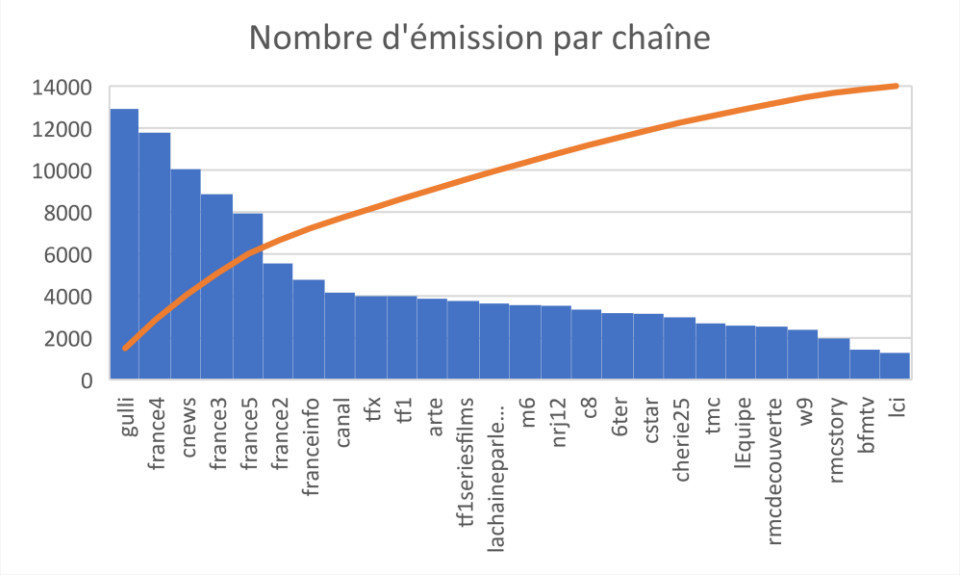

Figure 2

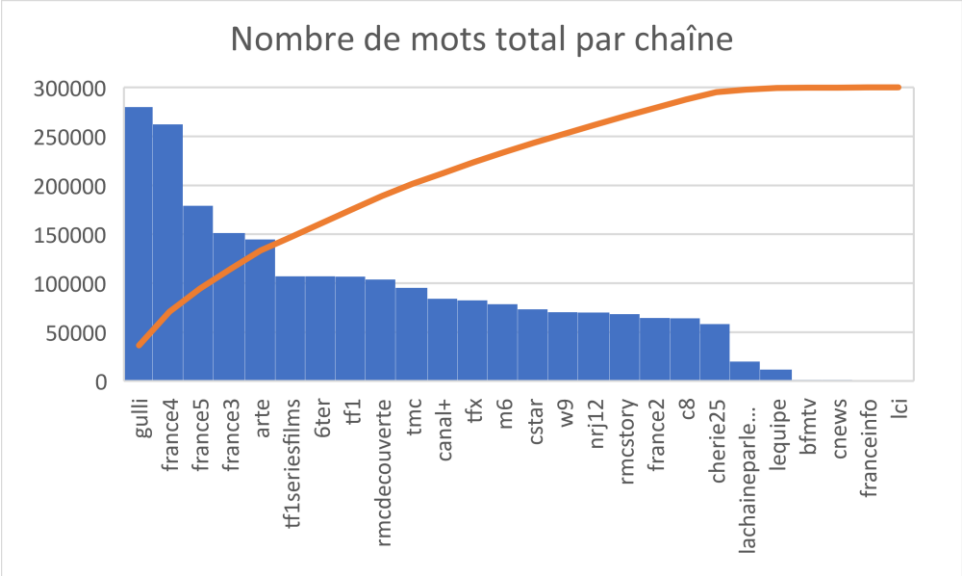

Figure 3

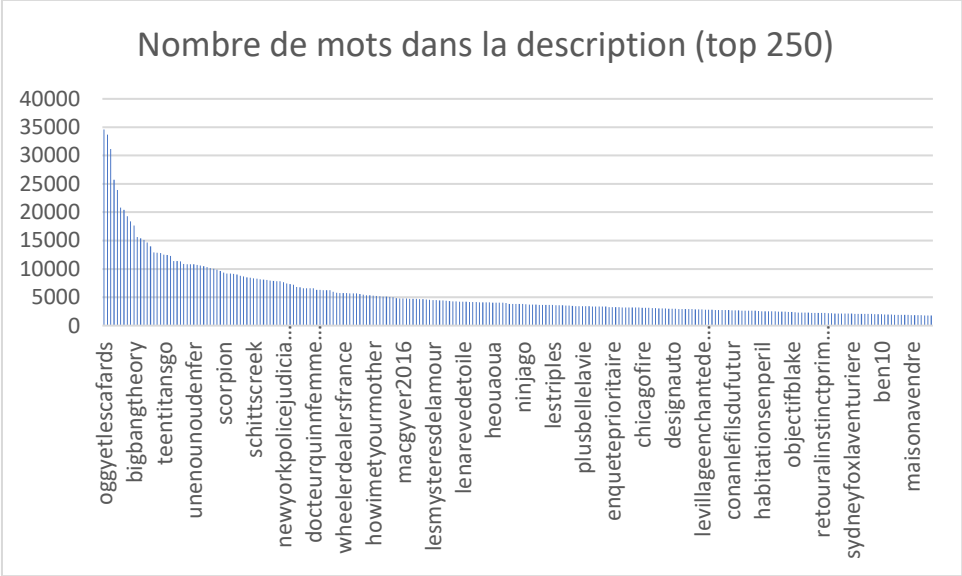

Figure 4

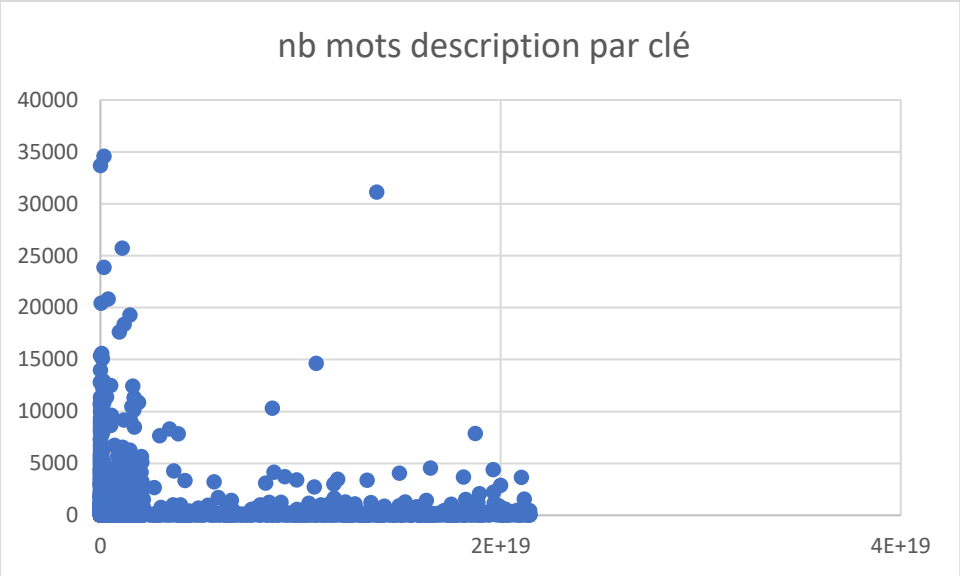

Figure 5

Avec ces statistiques, on peut voir qu'il y a une grande disparité entre les programmes : 450 n'ont pas de description, tandis que certains ont plus de 10 000 mots de description. De plus, les figure 2 et 3 permettent de voir qu'il existe certainement une corrélation entre le nombre d'émission et la taille de la description, ce qui semble logique puisque plus d'émission veut dire potentiellement plus de descriptions uniques. On peut également voir que surprenamment, Gulli est la chaîne avec les descriptions les plus fournies, même si les enfants ne sont pas forcément ceux à qui s'adresse ces descriptions. De plus, Gulli met souvent une description correspondant à l'épisode en question, ce qui permet d'accroitre la volumétrie de donnée. A l'inverse, les chaînes d'informations ont peu d'émissions car celles-ci sont plus longues, mais on ne peut pas savoir à l'avance les actualités du jour, ce qui empêche les descriptions uniques au profit de celle plus générique.

7. Les images tirées du web

Comme indiquer dans la partie 4, le XMLTV associe une image web à un programme. Ces images ont donc été ma base d'expérimentation pour étudier les liens entre les images et les descriptions. Celle-ci sont généralement des images génériques qui peuvent être tirées ou non du programme en question.

La première étape de leur étude a été de rassembler tous les doublons (déterminé par l'url), afin d'avoir leurs nombres d'occurrences et le nombre de programmes auxquels ils sont associés. En effet, une image apparaissant dans plusieurs dizaines de programmes, comme le logo déconseillé au moins de 10 ans, n'est peut-être pas la plus adaptée pour une étude avec un programme en particulier. Nous avons donc uniquement étudié les programmes ayant au moins trois images apparaissant uniquement dans ce programme et ayant au moins 300 mots de description pour avoir un volume de mots suffisant à étudier.

Exemples d'images n'ayant pas été retenues à cause de leur généricité :

Figure 6: 912 apparitions dans 164 programmes

Figure 7: 994 apparitions dans 974 programmes

8. Statistiques avec ces images

Avec cette nouvelle sélection de programmes, nous pouvons refaire quelques statistiques :

- Nombre total de programmes : 708
- Moyenne de mots par description : 2487,76271
- Nombre de mots total : 1761336
- Équivalent en nombre de romans : 22,0167
- Moyenne d'émissions par programme : 114,418079
- Nombre d'émissions total : 81008
- Moyenne de mots par émission : 21,7427415
- Nombre d'images total : 13747
- Moyenne d'images par programme : 19,4166667
- Moyenne de mots par image : 128,125118

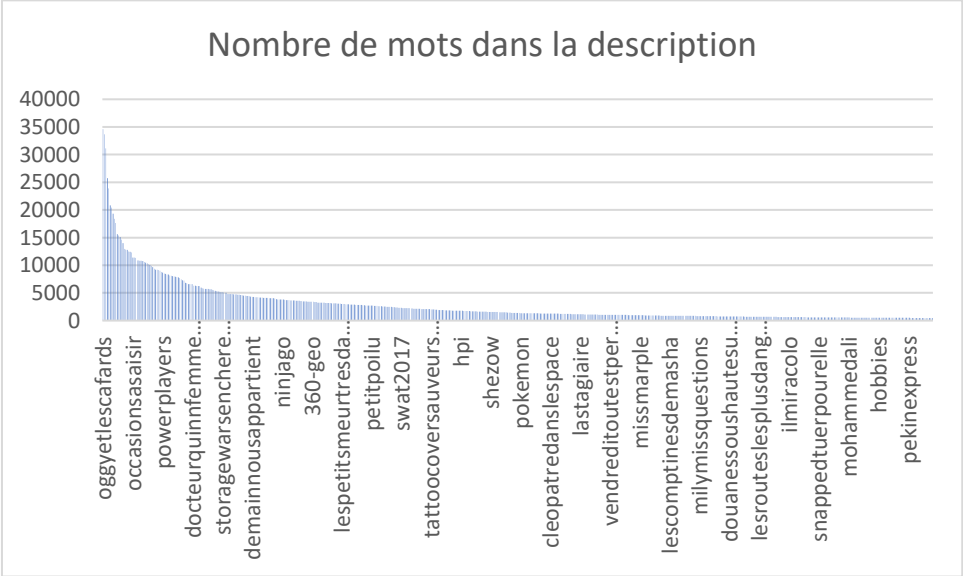

Figure 8

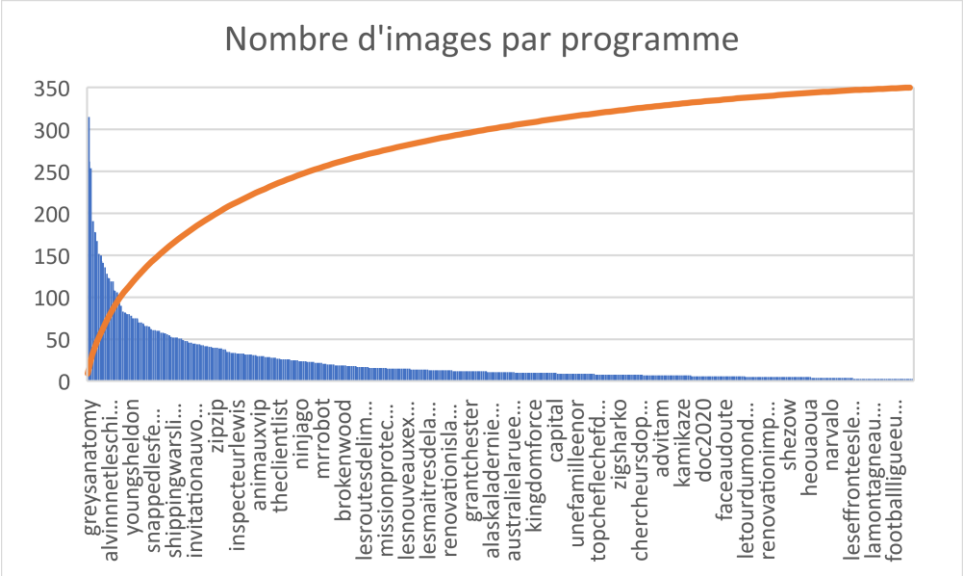

Figure 9

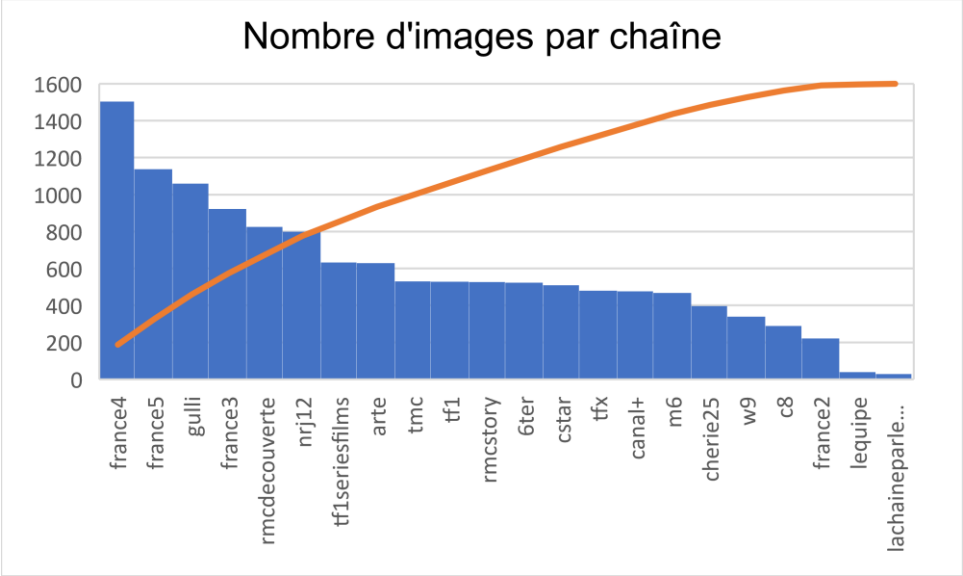

Figure 10

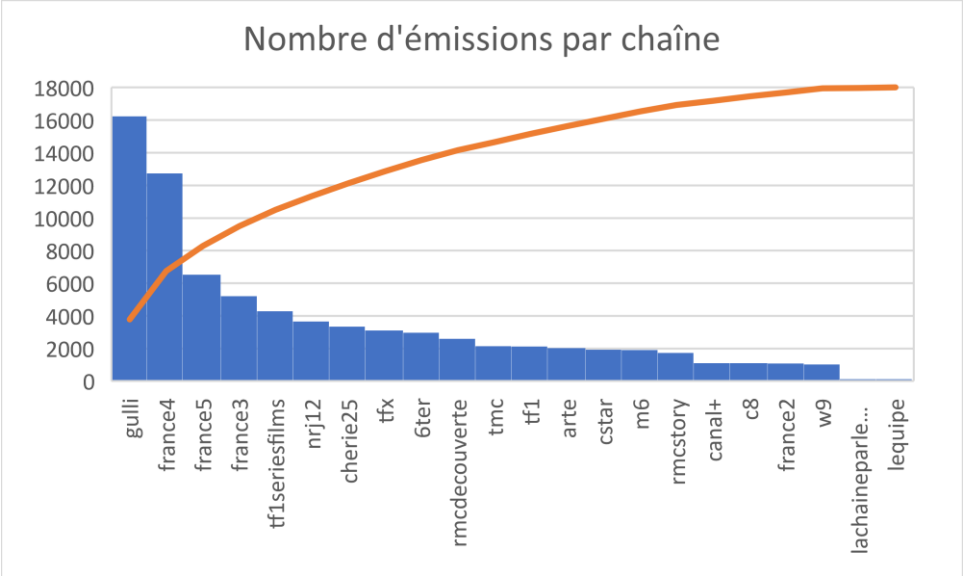

Figure 11

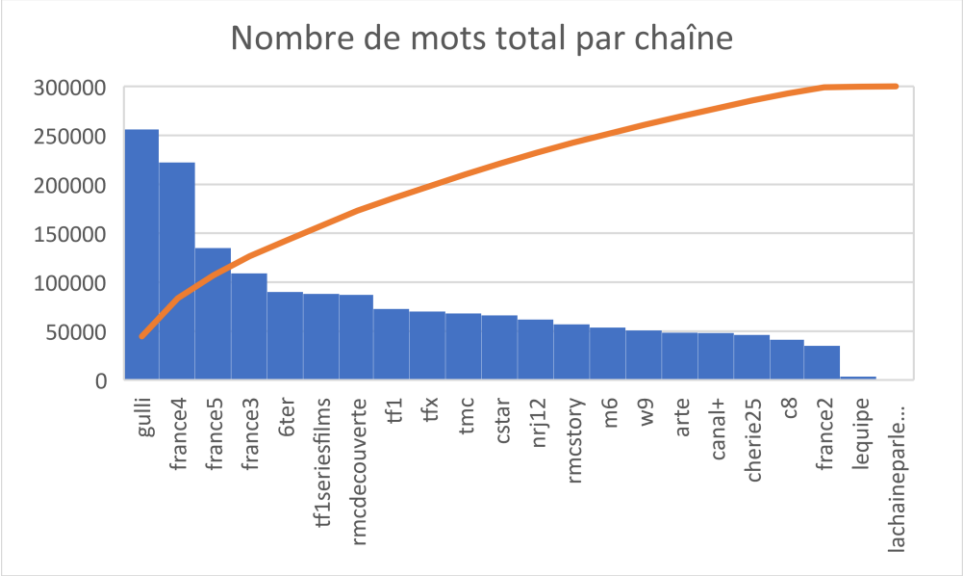

Figure 12

En comparant les figures 10, 11 et 12 aux figures 2 et 3, on s'aperçoit que Bfmtv, Cnews, France info et LCI ne sont plus représentées. Ces quatre chaînes ne possèdent donc aucun programme répondant à nos critères et ne sont donc plus considérées. Avec ces mêmes figures, on peut voir que ce sont généralement les chaînes avec le plus d'émissions qui ont le plus d'images, même si les écarts sont moins importants.

On peut également constater que bien que le nombre de programme maintenant considéré (708) soit bien inférieur à celui initiale (7082), il reste néanmoins globalement stable par rapport à celui avec seulement le nombre de mots comme critère (772).

9. Tags des images

Une fois les images à étudier identifiées, je les ai passées à Léo dans un fichier CSV afin qu'il puisse les étudier pour me donner leurs tags et la précision de ces tags selon son logiciel (imageNet). Il m'a renvoyé les résultats dans un fichier CSV.

Les tags trouvés ne sont pas toujours exacts (une image de girafe est détectée comme un guépard), la précision permet donc de limiter ces erreurs. La valeur classique qui sera par la suite utilisé sera 80%, mais l'utilisateur peut décider de la valeur à utiliser.

Figure 13: malinois à 37% et German_shepherd à 34%

Figure 14: Les dessins animés ne sont pas très bien gérés par le logiciel : comic_book à 95%

Figure 15: king_penguin à 99,99%

10. Unitex : Un logiciel pour trouver les entités nommées

10.1. Qu'est-ce que Unitex ?

Unitex est un logiciel libre fondé sur des dictionnaires et des grammaires pour l'analyse de corpus. Il se base sur des automates du traitement automatique des langues (TAL) pour gérer les dictionnaires et les grammaires afin de les appliquer à un texte pour l'analyser. Les dictionnaires utilisés sont construits par les membres du réseau RELEX, un réseau international de laboratoires spécialisés en Linguistique Informatique. Les grammaires permettent de décrire des règles syntaxiques ou sémantiques. Elles sont fondées sur des automates à états finis faisant appel à des dictionnaires afin d'analyser automatiquement des données textuelles. Unitex permet de créer ses propres grammaires afin de répondre à nos besoins.

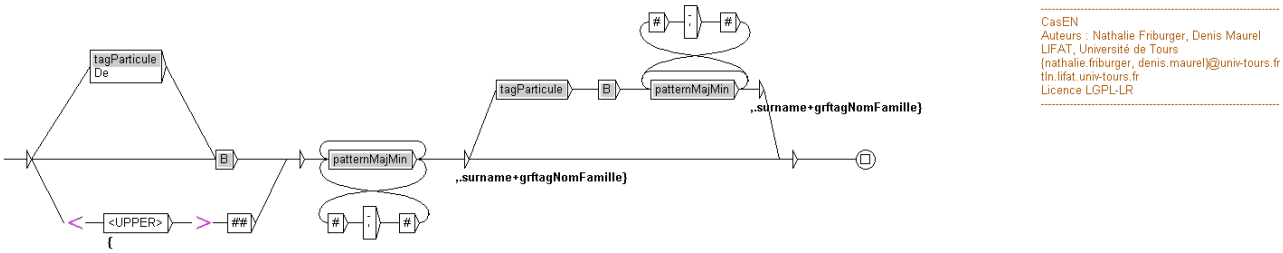

Figure 16: exemple d'automate pour trouver un nom de famille

Ces automates sont bien souvent connectés les uns aux autres afin de pouvoir définir le plus précisément possible le texte. Ainsi, si on trouve un nom de personne, celui-ci peut être

composé d'un prénom et d'un nom de famille, ce qui utilise au moins trois automates. Malgré tout, tous les automates ne sont pas connectés les uns aux autres, mais on peut dans ce cas les lancer en cascade pour analyser d'une traite le texte.

Pour analyser un texte avec Unitex, la première étape est de le prétraiter : le logiciel détermine le nombre de mots, le nombre de mots unique, le nombre de caractères uniques… et délimite les phrases du texte avec {S}. La deuxième étape est de lancer les automates pour trouver les mots reconnus par notre grammaire.

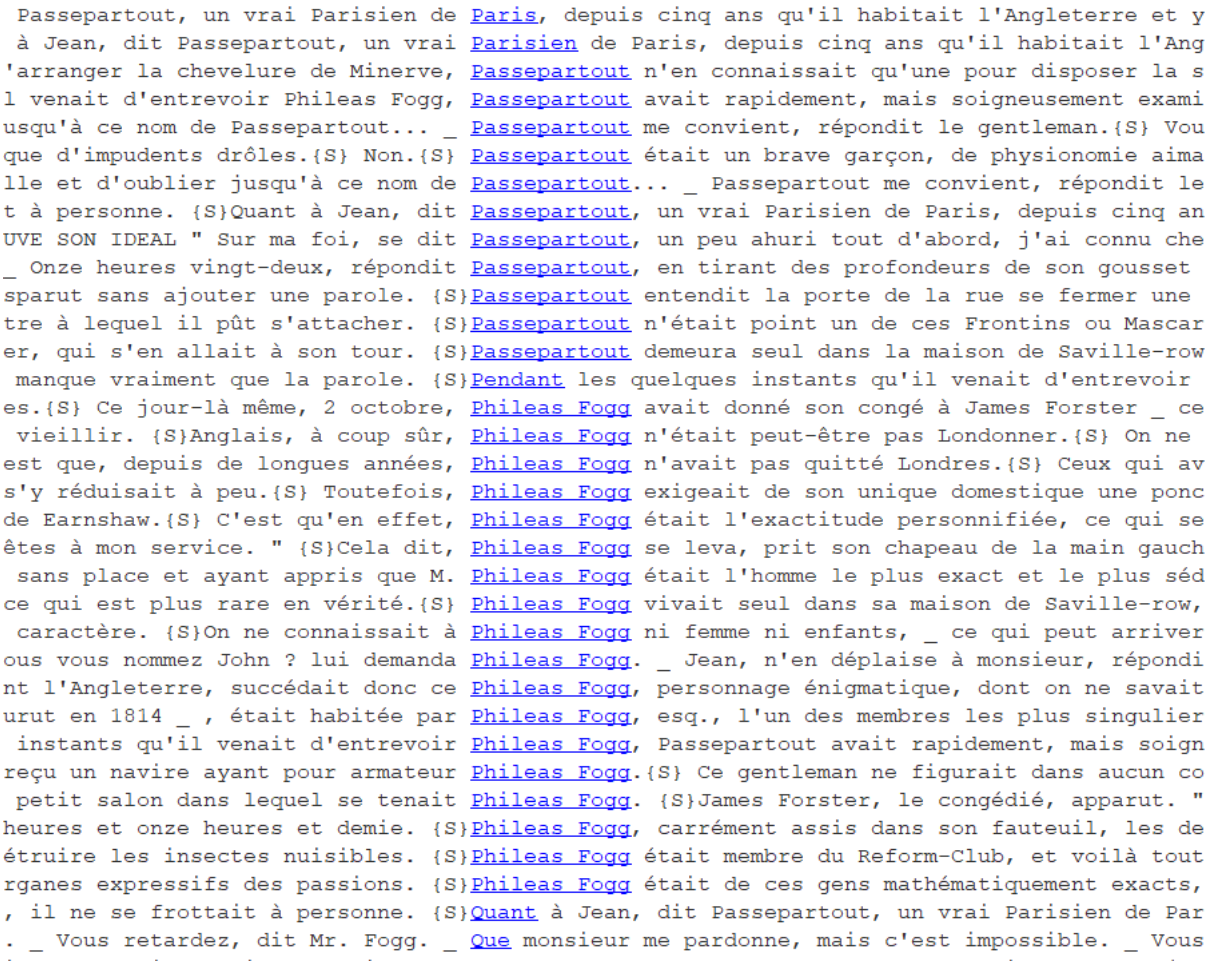

Figure 17: mots reconnus dans le tour du monde en 80 jours avec l'automate précèdent

Certains des mots reconnus peuvent ne pas être ce que l'on souhaite car il existe toujours des cas compliqués à prévoir, mais dans le cas présent, c'est certainement car le graph fait partie d'une cascade, et n'est normalement appliqué qu'après d'autre graph, ce qui permet de réduire les erreurs.

10.2. CasEN

CasEN est une cascade d'analyse pour Unitex afin de trouver les entités nommées (EN) d'un texte. Une EN est une expression linguistique référentielle, souvent associée aux noms propres et aux descriptions définies. CasEN a été développé par le LIFAT afin de répondre aux problématiques de reconnaissance des EN. Il se compose de plusieurs centaines

d’automates qui, grâce à leur ordre de passage, permet de trouver des EN sans qu’il y ait trop d’erreur.

Figure 18 : application de CasEN sur le tour du monde en 80 jours

Comme on peut le voir sur la figure 18, toute sortes d’EN sont ainsi trouvées. Néanmoins, le résultat que l’on obtient n’est pas très lisible, notamment lorsque l’EN est composé de plusieurs autre EN. CasEN permet donc également de faire une synthèse de ce résultat grâce à une deuxième cascade qui permet de mettre le résultat sous une forme de simili-XML.

Figure 19 : synthèse de CasEN sur le tour du monde en 80 jours

Tout ce processus peut être automatisé à l’aide de scripts. Deux en particulier sont fournis pour pouvoir utiliser CasEN : traiter un seul fichier et traiter l’ensemble des fichiers d’un dossier. J’ai principalement utilisé le deuxième afin de pouvoir analyser les descriptions de tous les programmes sélectionnés. Ces scripts peuvent être lancés depuis un invite de commande, mais il s’agit de fichier .bat, il ne me semble pas qu’il soit possible de les exécuter avec du

code Java, donc mon programme indique dans son déroulement d'exécuter la commande manuellement, et de lui dire quand Unitex a fini son analyse (ça peut prendre du temps).

10.3. Bugs trouvés

Lors de mon utilisation d'Unitex, j'ai rencontré deux bugs que j'ai pu reporter à M. Maurel. Le premier a été le plus embêtant : certaines balises <date> contenaient un attribut 'when' qui contenait du XML, par exemple :

<date when="2015-janvier-<extent>Du <val>21</val>au <val>26</val></extent>">

Cela rendait impossible la lecture de certains fichiers par mon parser. Le problème vient du fait que certains cas pour les dates n'ont pas été pris en compte. La solution que M. Maurel a pu me fournir sur le moment a été de désactiver les balises extent ainsi que d'autre et de désactiver l'automate pour les dates et le remplacer par un autre. Nous avons considéré que ces balises n'étaient pas les plus importantes pour mon étude, donc je pouvais m'en passer.

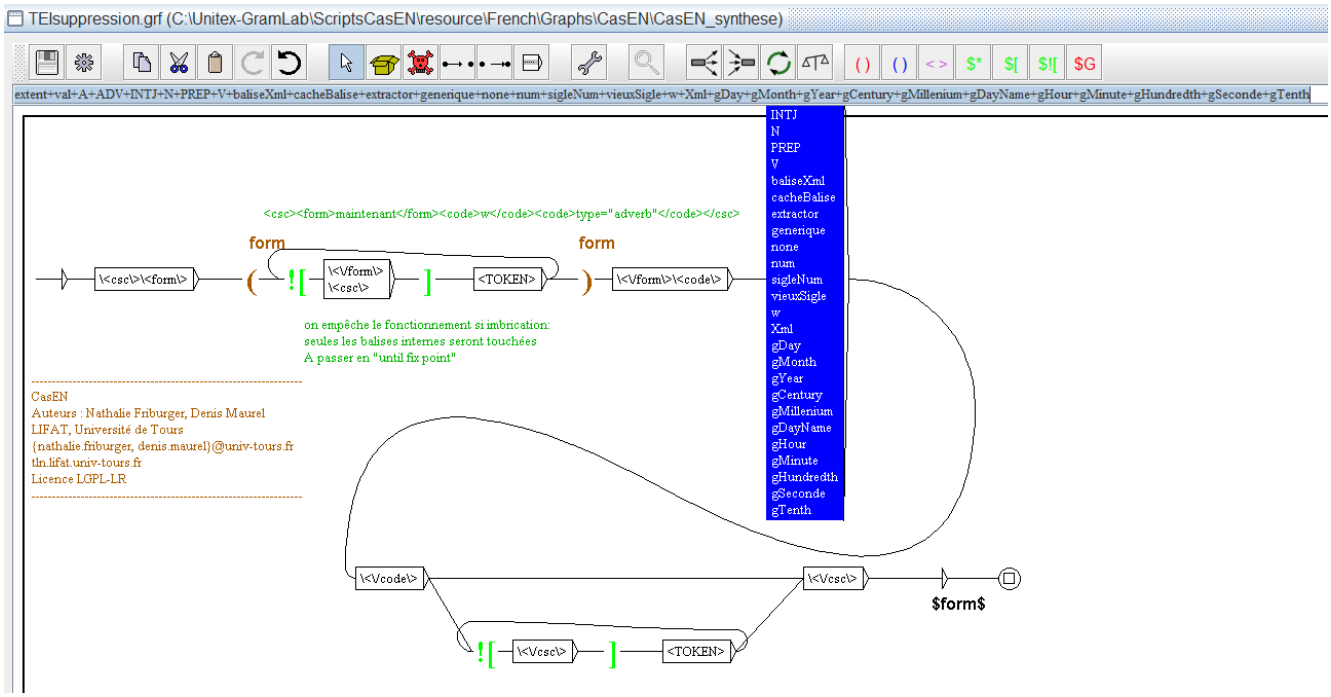

Figure 20: suppression de balises

Le deuxième bug est une inversion de deux balises de fin :

<date>2017 <postInclude>déjà</date></postInclude>

Pour ce bug, puisse qu'il ne concernait que deux fichiers, je l'ai juste résolu à la main.

11. Etudier les sorties d'Unitex

Comme dit précédemment, les fichiers de sortie d'Unitex avec CasEN sont des simili-XML : ils ressemblent à des fichiers XML mais ne le sont pas. Par exemple, voici le fichier intégral de sortie sur la description d'un programme :

<p><s>Un rendez-vous d'information synthétique, pour comprendre le monde et les enjeux <time type="adverb">du moment</time>.</s> <s>Un décodage moderne de l'actualité <extent>en <measure type="duration">quinze <unit>minutes</unit></measure></extent> d'infos denses et pédagogiques.</s>

La racine du fichier, <p>, n'est jamais refermé, ce qui empêche les parsers de pouvoir traiter le fichier. Également, tout le texte de la description est présent dans le fichier, alors que seules les EN nous intéressent, le contexte nous importe peu. J'ai donc dû faire un prétraitement des fichiers de sortie avant de pouvoir les parser. Ainsi, on obtient un fichier de ce type :

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<programme>
	<time type="adverb">du moment</time>
	<extent>en
		<measure type="duration">quinze
	<unit>minutes</unit>
		</measure>
	</extent>
</programme>
```

De plus, certaines descriptions contenaient des caractères problématiques pour le XML comme '<' et '>', il a donc fallu les supprimer avant de les analyser avec Unitex.

Ces fichiers contiennent du XML non prédictible dans le sens où je ne connaissais pas toutes les balises possibles et les liens parents/enfants peuvent être très variable. Par exemple, la balise <persName> peut avoir un ou plusieurs enfants, mais s'il n'en a qu'un seul, cet enfant peut ne pas toujours être le même.

```
<persName><surname>Dubois<surname><persName>
<persName><firstName>Bertrand><firstName><persName>
<persName><firstName>Bertrand><firstName><surname>Dubois<surname><persName>
```

De plus, certains enfants peuvent également être des parents, la profondeur du XML est donc également inconnue. Ma solution a été de faire en sorte que chaque élément tiré d'une balise puisse avoir une potentielle liste d'enfants, ainsi la profondeur peut être infinie, et chaque élément a un attribut pour définir le nom de la balise. J'ai ensuite parser les fichiers avec le parser DOM récursivement afin de trouver toutes les balises. J'ai également utilisé une fonction récursive afin de lier les éléments tirés de balises parent et enfant et ainsi gardé la richesse trouvée par Unitex dans mon code.

La dernière étape est de regrouper les objets récupérés d'Unitex avec la liste des programmes. Pour ce faire, chaque fichier traité par Unitex a été généré avec comme nom le hash de son

programme. Après analyse, ce hash reste toujours dans le nom du fichier, donc on peut le récupérer afin de pouvoir relier les programmes et les objets liés à Unitex.

12. Statistiques sur les entités nommées

En tout, plus de quarante types d'EN ont été identifiés dans les descriptions des programmes. On peut donc faire quelques statistiques sur ces EN trouvées :

- Moyenne de balises par programme : 294,2420749
- Nombre de programmes sans balise : 4
- Nombre de programmes avec plus de 10 balises : 683
- Nombre de fois où une balise est en plus de 50 exemplaires dans un programme : 787
- Nombre de fois où une balise est en plus de 100 exemplaires dans un programme : 358
- Nombre de fois où une balise est en plus de 200 exemplaires dans un programme : 155
- Nombre de fois où une balise est en plus de 500 exemplaires dans un programme : 39
- Nombre de fois où une balise est en plus de 1000 exemplaires dans un programme : 14

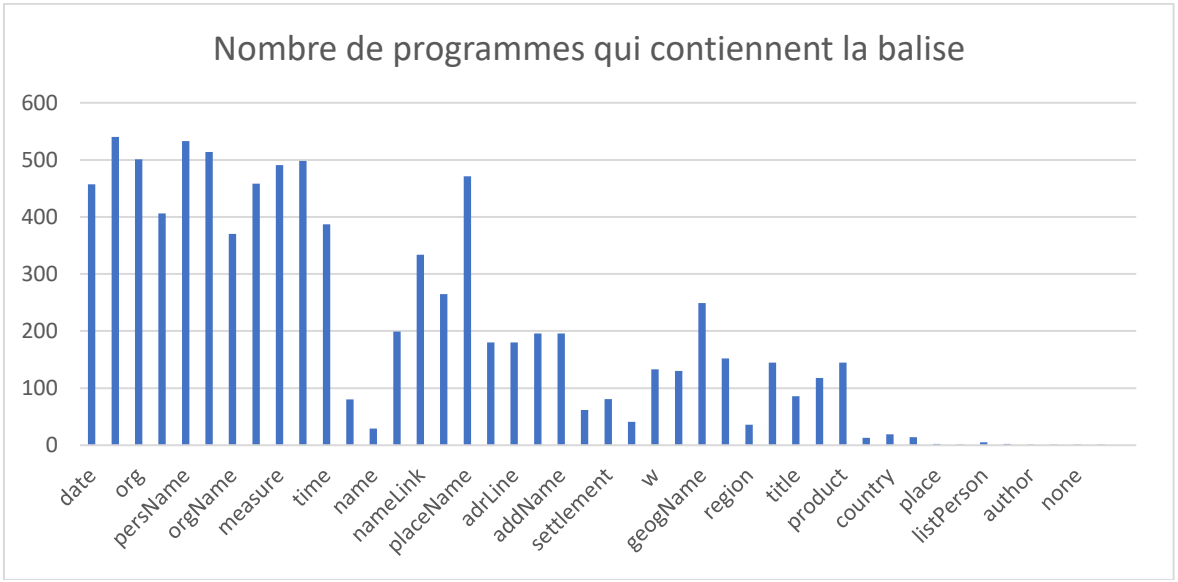

Figure 21

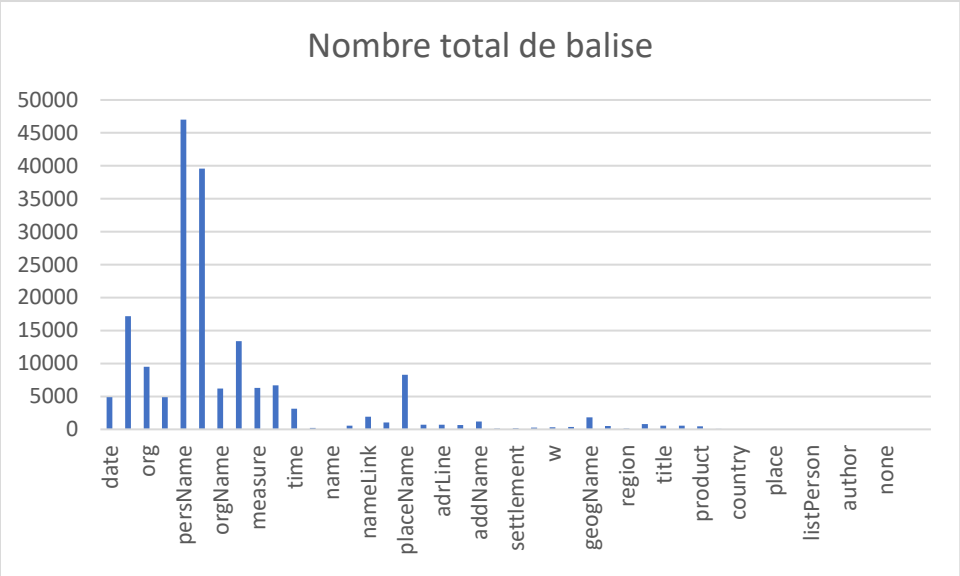

Figure 22

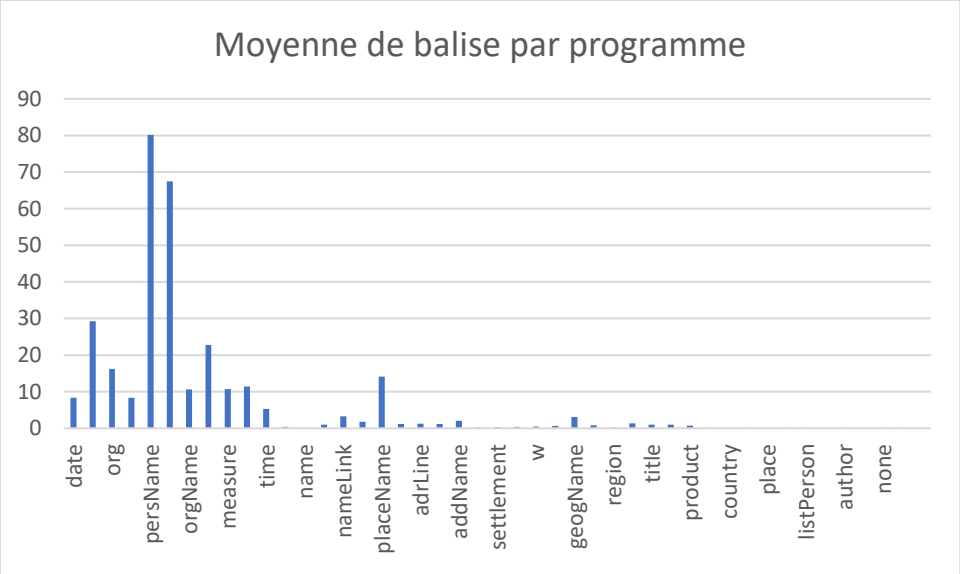

Figure 23

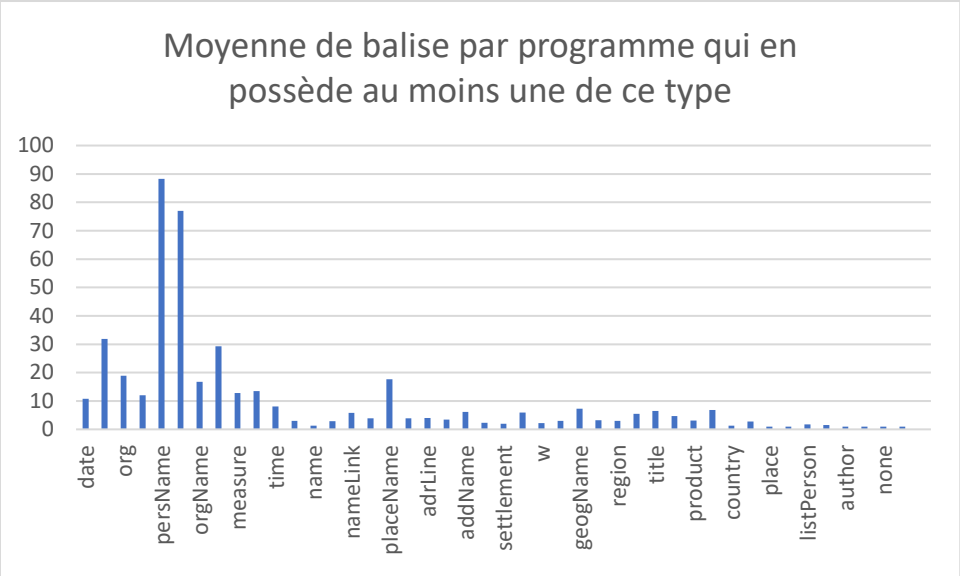

Figure 24

Comme on peut le constater, la très grande majorité des programmes possède au moins une EN, et une majorité en contient un nombre suffisant pour les étudier (plus de dix EN). Avec la figure 21, on peut voir que les programmes contiennent généralement une grande variété de type d'EN, mais avec les autres graphiques, on constate que cette diversité cache une grande disproportion sur le nombre d'EN par type. Ainsi, placeName est présent dans plus de 450 programmes, mais n'est présent qu'environ 20 fois par description, pour un total de moins de 10 000 occurrences, alors que dans le même temps, persName est présent dans plus de 500 programmes, apparaît presque 90 fois en moyenne par programme et représente au total plus de 45 000 EN. PersName fait également certainement parti des balises qui sont présentes en très grand nombre dans certains programmes, avec parfois plus de 1000 occurrences, tandis que d'autres balises comptabilisent moins de 10 occurrences sur tous les programmes, comme author ou timePeriod.

13. Faire le lien entre les descriptions et les images

Maintenant que nous avons des informations sur les descriptions grâce aux EN, et aux images grâce aux tags, nous pouvons commencer à trouver des liens entre eux. Dans un premier temps, j'ai fait une LUT remplissable à la main. Il a donc fallu réduire le nombre de programmes à considérer à des fins de test. Les nouveaux critères étaient donc :

- Plus de 1500 mots de description
- Uniquement les programmes diffusés sur France 2, France 3 et France 5 pour correspondre au futur projet sur l'influence de la télévision sur le tourisme
- Uniquement les programmes faisant plus de 20 minutes pour être sûr de pouvoir capturer des images de cette émission (elle peut avoir du retard ou terminer plus tôt que prévu)
- Prendre les images ayant un taux de confiance suffisant, tel que défini par l'utilisateur (80% par défaut)
- Utiliser uniquement les tags représentant au moins 10% de l'ensemble des tags représentant un programme

- Seule les 11 EN les plus fréquentes sont utilisées pour l'analyse

L'objectif est donc de pouvoir trouver manuellement des liens entre les EN et les tags grâce aux connaissances de l'utilisateur et de sa maitrise de google. Il peut ainsi donc indiquer qu'un lien existe entre le tag 'volcano' et les EN 'Réunion' et 'volcan'.

13.1. Interface v1

Pour faciliter le travail de l'utilisateur, j'ai créé une interface graphique avec javafx :

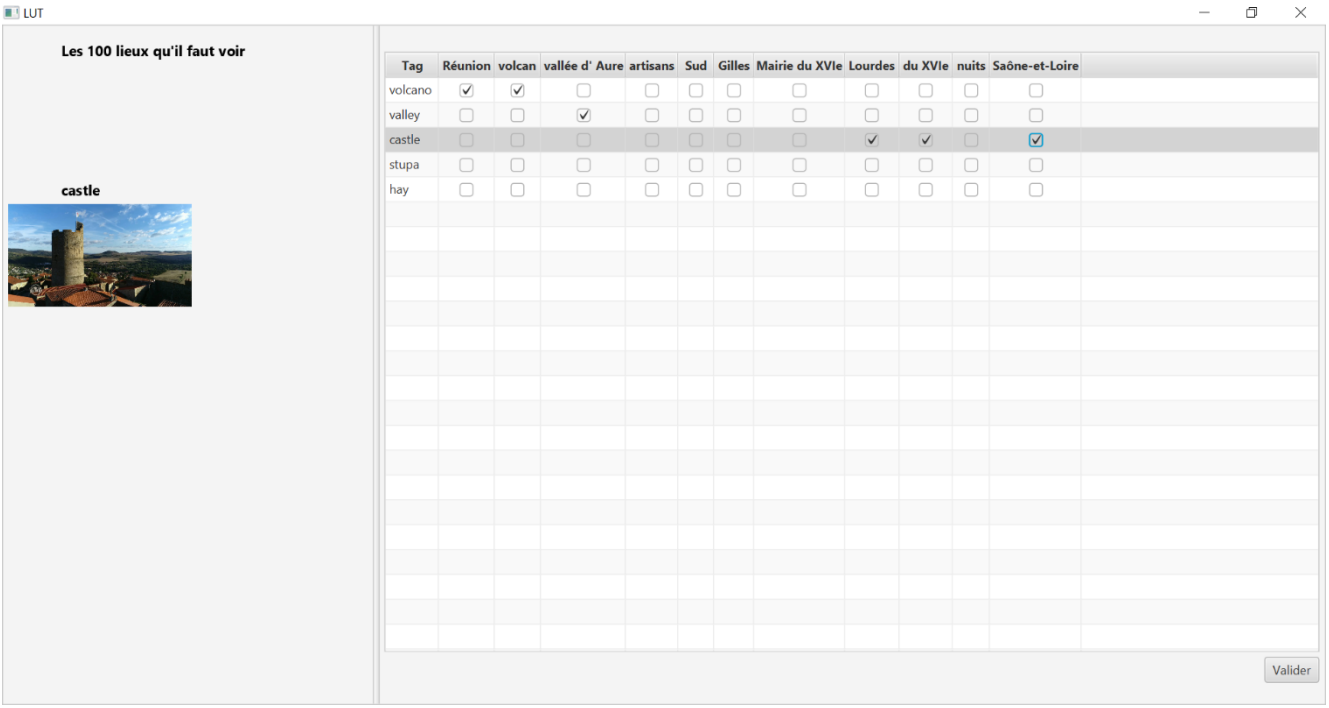

Figure 25

Cette première version présente la liste des 11 meilleurs EN dans les colonnes, et la liste des tags retenus sur les lignes. Les liens sont indiqués grâce aux checkbox. Un seul programme est étudié à la fois, appuyer sur le bouton valider permet de changer de programme et de conservés les liens trouver sous forme d'un fichier CSV. Le titre du programme est indiqué en haut à gauche afin de pouvoir facilement se renseigner sur le programme, et l'image associé au tag est affiché afin d'avoir une idée de ce qu'on étudie, mais ne doit pas servir de base pour créer les liens. En cliquant sur une ligne, l'image et le tag affichés à gauche changent pour correspondre à la ligne concernée.

Différents problèmes avec cette interface sont apparus. Premièrement, elle est orientée image : chaque image a un tag qui a une ligne dans le tableau. Cela va poser un problème quand on aura la base vidéo qui contiendra beaucoup d'images, donc plusieurs à peu de temps d'intervalle, et donc avec toujours un même tag.

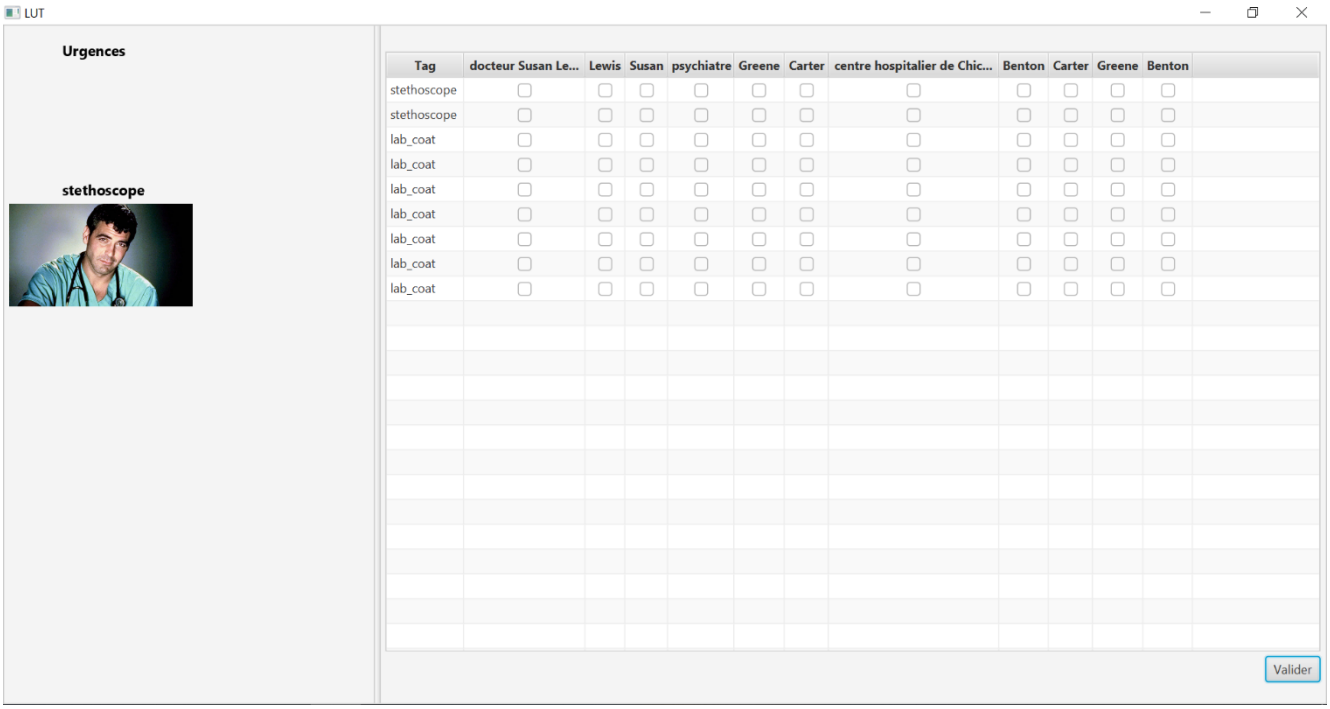

Figure 26

Comme on peut le voir sur la figure 26, ces tags, tous identiques, prennent de la place, et surtout du temps à l'utilisateur pour remplir le tableau, alors qu'ils ont tous exactement les mêmes liens. Un deuxième problème concerne la présentation des EN : on ne sait pas leur type, et certains semblent être des doublons, comme Berton et Berton dans la figure 26. Ce dernier vient du fait que certains EN parents ont exactement la même valeur que leur enfant, mais comme les deux sont présents en grand nombre, ils font tous deux parties du top 11 des EN les plus fréquent de leur programme. Les EN qui contiennent une partie d'une autre (docteur Susan et Susan) sont néanmoins à conserver car on peut avoir des homonymies qui peuvent être distingués par ces cas.

13.2. Interface v2

La deuxième version de l'interface a donc résolu les problèmes de la première version :

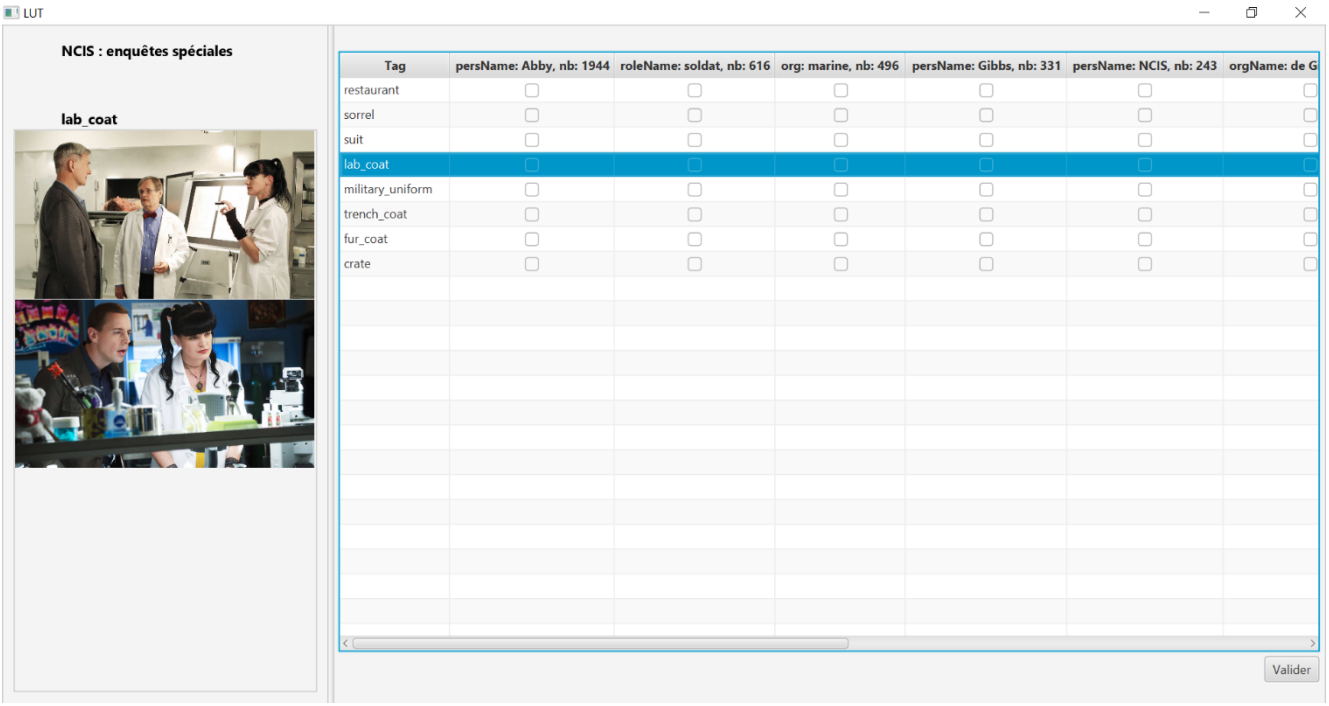

Figure 27

L'interface est maintenant orienté vidéo : elle regroupe ensemble des images ayant le même tag et les affiche en même temps, leur nombre pouvant être infinie grâce à une scroll bar. La taille des images est également automatiquement ajustée à la taille de leur espace afin de pouvoir les voir en plus grand. Le nom des colonnes a été modifié afin de contenir le nom du type de l'EN ainsi que son nombre d'occurrence sur les conseils de Mme. Friburger afin de savoir la pertinence de l'EN pour le programme.

14. Automatisation des liens

L'objectif désormais est de pouvoir automatiser ce processus. Pour ce faire, nous avons fait appel à des api afin de recueillir des informations sur les EN et les tags.

14.1. Wikidata

Wikidata est une base de données libre, collaborative, multilingue, et secondaire qui collecte des données structurées pour alimenter Wikipédia, Wikimédia Commons, les autres projets du mouvement Wikimédia et pour n'importe qui partout dans le monde.

Il possède donc une liste d'éléments dans plusieurs langues qui sont interconnectés entre eux par des propriétés comme sur la figure 28. L'objectif est donc de pouvoir récupérer la liste de tous les éléments auquel un élément est relié. Les éléments sont représentés par leur identifiant qui commence par un 'Q', et les propriétés par un identifiant commençant par 'P'.

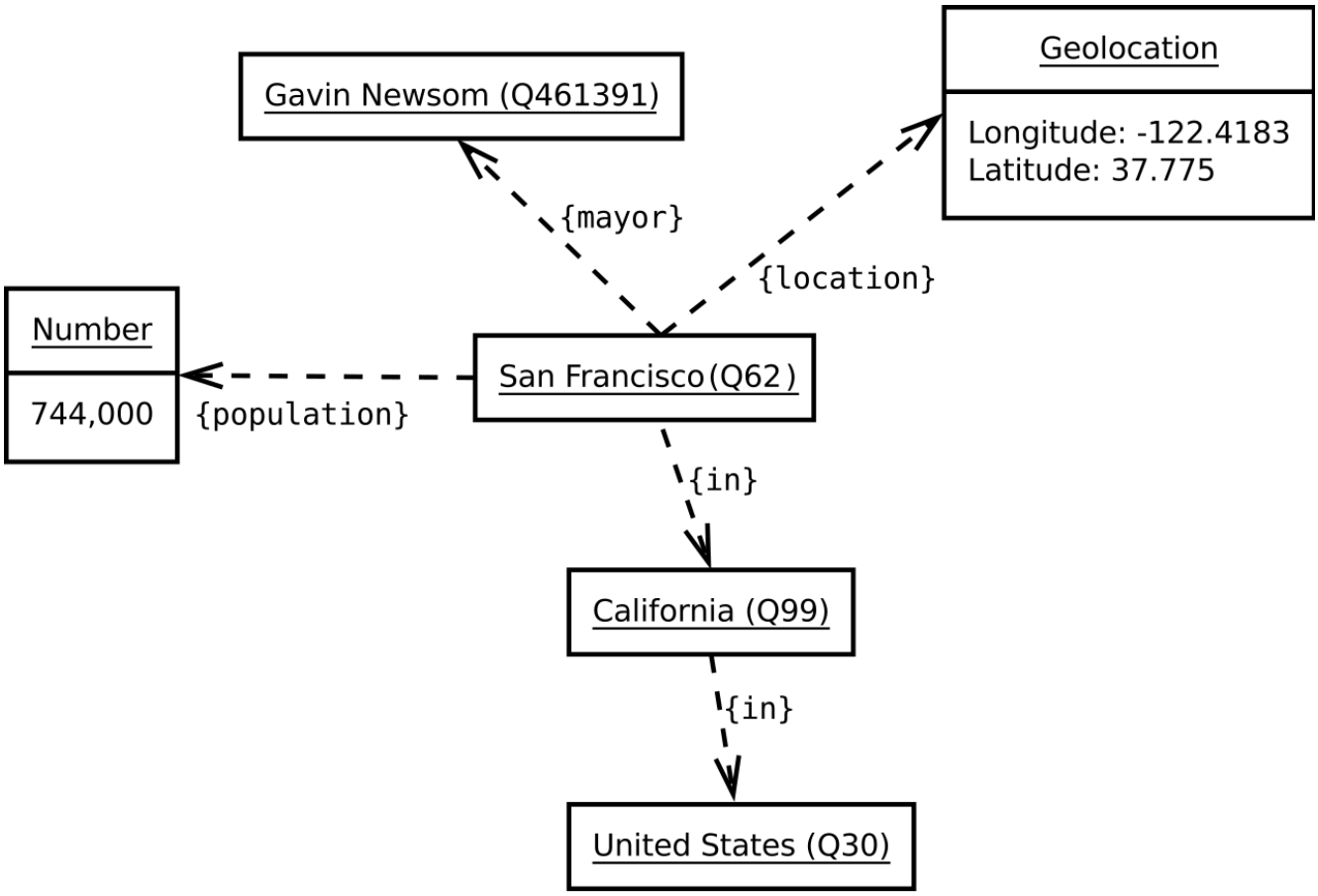

Figure 28

14.2. Le problème de sparQL

Wikidata possède un service de requête se basant sur sparQL. Une requête basique avec cet outil se présente sous la forme suivante :

```
#Selectionne tous les éléments ayant pour nature chat
SELECT ?item ?itemLabel
WHERE
{
  ?item wdt:P31 wd:Q146. # doit avoir comme nature chat
  SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en". }
}
```

SparQL fonctionne avec des triplets : on a l'élément à trouver, la propriété, et l'élément qu'on connait. On ne peut pas avoir plus d'une inconnue par triplet. Malgré mes recherches, je n'ai pas trouvé comment faire une requête avec cet outil pour me donner la liste des propriétés d'un élément. Il existe des milliers de propriétés, et chaque élément en a des différentes. Les EN et les tags que je veux étudier sont très variés, donc je ne peux pas juste définir une liste arbitraire de propriétés, car il y en aura toujours une qui sera très importante pour un type de mot que je vais oublier. De plus, pour chaque propriété, la requête peut se faire dans deux sens. Dans la requête précédente, en inversant ?item et wd:146, on peut obtenir la nature de

l'élément chat. Il y aurait donc un très grand nombre de requête à faire, dont beaucoup de retourneraient rien.

14.3. Récupérer tous les liens avec Wikidata

Wikidata possède d'autres manières de faire des requêtes, l'une d'elle est par cette url :

https://www.wikidata.org/wiki/Special:EntityData/Q2599.json?flavor=simple

Figure 29

Celle-ci permet, avec l'identifiant unique d'un élément, d'obtenir la liste des propriétés qui lui sont associés, ainsi que les valeurs (sous forme d'identifiant) de ces propriétés pour cet élément, le tout au format JSON. Ainsi, en une seule requête, on a accès à tous les éléments adjacents à notre mot d'origine.

Il faut maintenant pouvoir passer d'un mot à son identifiant sur Wikidata, et de l'identifiant à son nom. Quand on a un mot au départ, on peut utiliser l'url :

https://fr.wikipedia.org/w/api.php?action=query&list=search&srlimit=1&format=json&srsearch=girafe

Figure 30

Ce qui permet d'obtenir le titre exact de l'élément sur Wikidata, ce qui permet de gérer les problèmes de majuscules, accents ou autre faute de frappe. Par exemple, la figure 30 nous apprend que le mot 'girafe' doit être écrit 'Girafe' pour pouvoir être utiliser dans d'autres requêtes. Ensuite, on utilise l'url :

https://fr.wikipedia.org/w/api.php?action=query&format=json&prop=info|pageprops&titles=Girafe

Figure 31

Elle permet d'obtenir l'identifiant de notre mot sur Wikidata. Pour faire le chemin inverse, de l'identifiant au nom, on peut à nouveau utiliser la première url, mais en regardant dans la partie labels.

Figure 32

On peut donc obtenir le nom de notre élément dans plusieurs langues, mais il manque parfois des traductions, donc on ne peut pas se baser uniquement sur Wikidata pour obtenir des mots en français.

14.4. La traduction avec des api

Beaucoup d'api de traductions sont payantes, mais elles permettent de pouvoir traduire beaucoup de données. Dans notre cas, la volumétrie de données à traduire est faible : les tags et les éléments retourné par les requêtes de Wikidata. L'api de traduction la plus connu est google traduction, mais elle est payante. J'ai réussi à trouver comment l'utiliser par une url gratuitement, mais à terme, il faudra certainement passer à la version payant pour avoir une version officielle du projet qui ne se base pas sur un élément piraté. L'url utilisé est la suivante :

https://translate.googleapis.com/translate_a/single?client=gtx&sl=en&tl=fr&dt=t&q=volcano

Elle retourne un fichier JSON contenant notre mot traduit dans la langue ciblée depuis la langue indiquée. Dans notre cas, on veut une traduction en français depuis l'anglais.

14.5. Créer les liens

Maintenant que l'on possède des mots associés aux tags et aux EN, et que tous sont dans la même langue, nous pouvons commencer à crée automatiquement des liens entre eux. Tous les liens ne pourront pas forcément être trouvés, mais cela permettra à l'utilisateur d'avoir une base qu'il pourra compléter si besoin. Pour ce faire, pour chaque case du tableau, on compare les mots, le tag et l'EN entre eux. Si l'EN, le tag ou un mot est compris dans un des mots ou

l'EN/tag, alors on note qu'il y a un lien. Par exemple, pour l'émission 'les 100 lieux qu'il faut voir', le tag Château et l'EN Saône-et-Loire ont un lien car l'EN est associé au mot Châteauneuf.

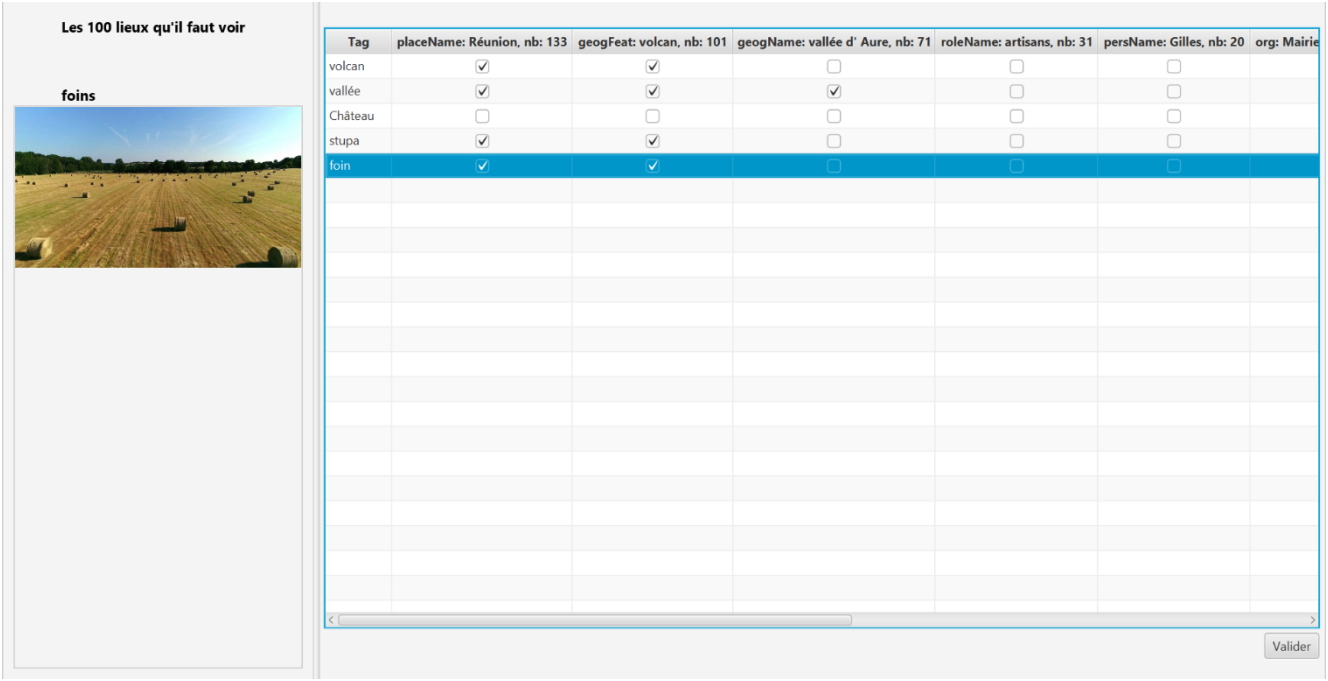

Figure 33 : annotation automatique d'un programme

Pour ce programme, sans intervention de l'utilisateur (hors bouton valider), le fichier CSV produit est le suivant :

url (1 exemple); **tag**; liste EN choisi
https://focus.telerama.fr/1111x625/2021/12/04/185747d20f6858f66a968aef7e6d45c5de4281e1.jpg; **volcan**; Réunion; volcan; nuits; Saône-et-Loire; forêts;
https://focus.telerama.fr/1111x625/2021/12/07/50ac62bb0628fb327d3404282d043ce4f21ad41d.jpg; **vallée**; Réunion; volcan; vallée d' Aure; nuits; forêts;
https://focus.telerama.fr/1111x625/2021/12/11/09f065aa129af87eba7648a8849f8a0c20109a39.jpg; **Château**; Saône-et-Loire;
https://focus.telerama.fr/1111x625/2021/12/13/f777624eefa14bdcf63528da231fd369fdbfa7e7.jpg; **stupa**; Réunion; volcan; Lourdes; nuits; Saône-et-Loire; forêts;
https://focus.telerama.fr/1111x625/2022/03/29/0f72bc86bdf371eafbde22fdcffd278da08ed1f3.jpg; **foin**; Réunion; volcan; nuits; forêts;

On a donc 21 de trouvé. Certains sont logiques (Réunion et volcan, vallée et vallée d'Aure) mais d'autres le semblent moins comme les liens avec stupa. Le problème est que certains des mots trouvés sont très génériques et reviennent souvent comme 'Dictionnaire encyclopédique Brockhaus et Efron' ou 'L'Encyclopédie Nuttall'. Il faudrait donc trouver un moyen d'identifier ces mots pour ne pas les considérer pour l'étude afin d'obtenir des résultats plus précis. Pour ce programme, stupa et foin n'ont des liens que grâce à ces mots générique. De plus, si l'on décide d'étudier également les liens plus lointain (deux ou plus de distance), ces mots risquent de fausser les résultats puisqu'il associe des choses qui ont peu de liens.

15. Génération de fichiers CSV pour les autres membres du projet

Pendant la durée de mon stage, j'ai pu fournir des fichiers CSV à différents membres du projet station TV.

Pour Aurélien, je lui ai fourni la liste des titres des programmes ayant plus de 300 mots de description afin qu'il puisse les étudier dans le cadre SEO. Je lui ai ensuite donné le même fichier mais avec la base 300 chaînes. Enfin, je lui ai généré la liste des EN représentant au moins 0,3% du nombre total de mots de la description d'un programme.

Pour Léo, comme expliqué dans la partie 9, je lui ai donné la liste des images des programmes ayant au moins 300 mots de description et trois images. Il a ainsi pu me donner la liste des tags de ces images.

Pour Jules, je lui ai donné la liste des programmes diffusé sur France 2, France 3 et France 5 pour trois semaines précises, avec la contrainte habituelle de 300 mots de description. Cela lui permet de savoir quel sont les programmes à extraire de la base vidéo en priorité. Avec la base de programme à contenu politique et la base Factoscope, je lui ai fourni le nombre de mot total pour chacune, le nombre de mots uniques, le nombre d'EN trouvées par Unitex ainsi que la liste de ces EN. J'ai également fourni l'un de mes fichiers de statistique à son encadrant, et réalisé des statistiques pour la base Factoscope.

Dans le cadre d'une présentation du projet station TV, j'ai fourni à M. Delalandre les chiffres suivants :

	Hash codes	Text	Named entities
Web images	585	1.3 M words	176 996
TV videos	51	57 611 words	xx

16. Dans le futur

Si mon projet est repris, des pistes d'améliorations seraient :

- Pour l'automatisation des liens, ne pas prendre en compte les mots revenant trop souvent.
- Essayer de réduire le temps d'exécution : multithreading, meilleure gestion de l'appel aux apis, …
- Trouver les éléments ayant des liens avec les EN/tags à deux ou plus de distance.
- Relier l'interface/projet à la base vidéo.
- Choisir les EN à utiliser pour faire les liens en prenant en compte leur intérêt SEO.

17. Conclusion

Selon Mme. Friburger, j'ai réussi à accomplir l'objectif fixé, en dehors du fait d'utiliser la base vidéo car elle n'a pas pu être prête à temps. Mon travail va donc servir de base pour pouvoir enrichir les métadonnées et pouvoir analyser les programmes avec d'autres méthodes (SEO, …).

Ce stage m'a permis de réaliser un projet dans son intégralité, et en ayant une dynamique de groupe avec les autres membres du projet station TV. J'ai pu consolider mes connaissances dans certains domaines (Excel, javafx, …) et découvrir d'autres choses comme Unitex et Wikidata.

Je tiens à remercier Mme. Friburger pour m'avoir permis de faire ce stage, M. Delalandre pour sa disponibilité, et M. Maurel pour avoir résolu rapidement les bugs que j'ai pu trouver sur Unitex.

Bibliographie

- https://unitexgramlab.org/fr
- https://tln.lifat.univ-tours.fr/version-francaise/ressources/casen
- https://www.wikidata.org/wiki/Wikidata:Main_Page
- https://www.labnol.org/code/19909-google-translate-api
- https://stackoverflow.com

Enrichissement des métadonnées

Résumé :
Le projet station TV mené depuis 2019 par le Laboratoire LIFAT, il a pour objectif le développement de nouveaux services numériques autour de la Télévision, du Replay et d'Internet. Il recouvre aujourd'hui de multiples services pour la capture smart de vidéos et d'images, l'analytics TV, le fact checking et le scraping Web de données TV. Ce scraper collecte des métadonnées sur les programmes TV (titres, résumés, horaires, chaînes, URLs des images, etc.) stockées au format XMLTV, mais ces métadonnées sont faiblement enrichies. Il faut donc regrouper les programmes, identifier des mots clés, détecter les descriptions doublons, … Un objectif est également de faire des liens entre ces métadonnées, comme entre les images/vidéos du programme et sa description afin qu'ils puissent s'enrichirent mutuellement.

Mots-clés :
XMLTV, parsing, entités nommées, métadonnées

Abstract:
The TV station project is lead since 2019 by the LIFAT laboratory, its objective is to develop new digital services around television, replay and internet. Nowadays, it covers multiples services for smart captures of videos and images, TV analytics, fact checking and web scraping of TV data. This scraper gathers metadata of TV programs (title, summary, schedule, channel, URL of images, …) store in XMLTV format, but those metadata are dimly enriched. We must regroup the programs, identify key words, detect double descriptions, … An objective is also to make links between those metadata, like between images/videos of the program and its description, so they can enrich mutually.

Keywords :
XMLTV, parsing, named entity, metadata