MediaEval - Groupe Karadoc

Rapport de projet

Salima Azzou
Axel Bellec
Max Halford
Joseph Meunier
Giovanni Zanitti

Table des matières

Introduction

Le challenge MediaEval est une initiative de benchmarking dédiée à l'évaluation de nouveaux algorithmes d'accès et de récupération multimédia. Il met l'accent sur le "multi" dans "multimédia" et se concentre sur les aspects humains et sociaux des tâches de multimédia. MediaEval attire les participants qui s'intéressent aux approches multimodales du multimédia, par exemple : la reconnaissance vocale, l'analyse de contenu vidéo, audio ou encore textuel. Concrètement, l'objectif de ce challenge est de mettre en place un algorithme capable de prédire le thème principal d'une vidéo à partir de ses données audio, vidéo et textuelles.

Dans le cadre de notre Master 2 Statistique et Informatique Décisionnelle, notre équipe, composée de Salima Azzou, Axel Bellec, Max Halford, Joseph Meunier et Giovanni Zanitti, s'est engagée dans le challenge MediaEval 2016. Les objectifs de ce projet sont d'appliquer les théories vues en cours concernant l'analyse audio, vidéo et textuelle, mais aussi d'utiliser nos compétences en statistique concernant les algorithmes d'apprentissage. De plus, il nous a été demandé de mettre en place une organisation de groupe pour mener à bien ce projet. A noter que nous sommes en concurrence avec un autre groupe dans ce projet, un des objectifs est donc d'obtenir de meilleurs résultats qu'eux.

Ce rapport a donc pour but d'expliquer notre démarche et d'exposer nos résultats en développant d'abord les extractions de données que nous avons faites, puis les apprentissages non-supervisées réalisés ainsi que les algorithmes de classification utilisés. Enfin nous décrierons notre organisation globale.

Partie 1

Extraction

Nous avons écrit plusieurs scripts pour extraire des informations pertinentes des données fournies. Au final nous nous retrouvons avec plusieurs tableaux de données où les lignes sont les noms de fichiers sans extension et les colonnes sont les features que l'on utilisera pour faire de la classification.

Nous avons ignoré les documents *Dahowlett PulpoGaliciana440* et *Nitaai 00009LordNrisingha825* car il leur manquait des informations (leurs vidéos et les métadonnées existaient, cependant il n'y avait pas de shots associés). Nous nous retrouvons donc avec 573 documents.

1.1 Méta-données

Nous avons extrait :

- *attributes.csv* : un tableau d'attributs avec comme colonnes
  - *duration*, la durée de la vidéo
  - *licence*, la license sous laquelle a été partagée la vidéo
  - *size*, la taille mémoire des vidéos
  - *title*, le titre de la vidéo
  - *uploader_id*, l'identifiant de la personne qui a partagé la vidéo
  - *uploader_login*, le pseudo de la personne qui a partagé la vidéo
- *tf_descriptions.csv* : une matrice d'occurrences (TF) pour les termes contenus dans les description
- *tf_keywords.csv* : une matrice d'occurrences (TF) pour les tags
- *tf_titles.csv* : une matrice d'occurrences (TF) pour les termes dans les titres

Nous ne nous attendons pas à ce que les attributs nous permettent de distinguer des vidéos les unes des autres. En effet la durée des vidéos n'est pas forcément significative et la taille est directement corrélée à la durée. Cependant, l'identifiant des uploaders peut servir dans le cas où une personne poste de nombreuses vidéos pour un thème donné (par exemple une chaîne de cuisine).

Nous avons séparé les matrices d'occurrences pour nous réserver la possibilité les utiliser individuellement. Nous pouvons aussi les fusionner en faisant attention à som-

mer les valeurs pour les colonnes qui existent dans plusieurs matrices d'occurrences. Après quelques aller-retours avec la classification nous avons décidé de lemmatiser les termes et de supprimer les termes définis par moins de 3 caractères.

1.2 Transcriptions

Nous avons extrait :

— *speakers.csv* : un tableau donnant les locuteurs et leurs attributs par fichier avec comme colonnes
  — *dur*, le temps de parole du speaker
  — *gender*, le genre du speaker
  — *spkid*, l'identifiant du speaker
— *speakers_features.csv* : à partir du fichier précédent, nous avons pu extraire les features suivantes
  — *entropy*, indicateur sur la distribution de la parole (proche de 1 si la distribution est homogène, proche de 0 si une personne parle plus souvent que les autres par exemple)
  — *nb_M*, nombre de speakers homme
  — *nb_F*, nombre de speakers femme
  — *Freq_M/F*, fréquence homme/femme

Nous avons créé le fichier *speakers_features.csv* pour pouvoir obtenir un tableau avec autant de lignes que de documents. Évidemment en faisant une aggrégation des lignes pour chaque document en une seule alors on est susceptible de perdre de l'information, cependant nous avons fait ça pour rester cohérent avec les autres tableaux créés.

1.3 Images clés (« keyframes »)

1.3.1 Extraction du nombre de `shots` par vidéo

Pour chaque vidéo, nous détenons des « keyframes » qui représentent des changements de plans. Ils peuvent être considérés comme des « moments clés » d'une vidéo. Le nombre de shots par vidéos est un descripteur assez intéressant. En effet, pour une conférence on pourrait supposer qu'il y est un petit nombre de shots. Pour un reportage journalistique on peut imaginer qu'il y est un très grand nombre de plans.

Techniquement, cette extraction était très facile. Comme les shots d'une vidéo sont rangés dans un dossier parent correspondant à une vidéo, il faut juste effectuer un comptage du nombre de fichiers images dans un dossier.

1.3.2 Histogrammes de couleurs

Nous avons extrait de chaque shot son histogramme de couleur pour pouvoir ensuite les comparer. Ils sont stockés dans le fichier *hist_data.json*. Chaque histogramme contient 255 valeurs.

En travaillant sur ces histogrammes, nous nous sommes rendu compte que certains shots étaient de couleur unique. Nous avons décidé de récupérer cette information par shot dans une variable booléen valant `True` si le shot est de couleur unique.

1.3.3 Séparation en blocs

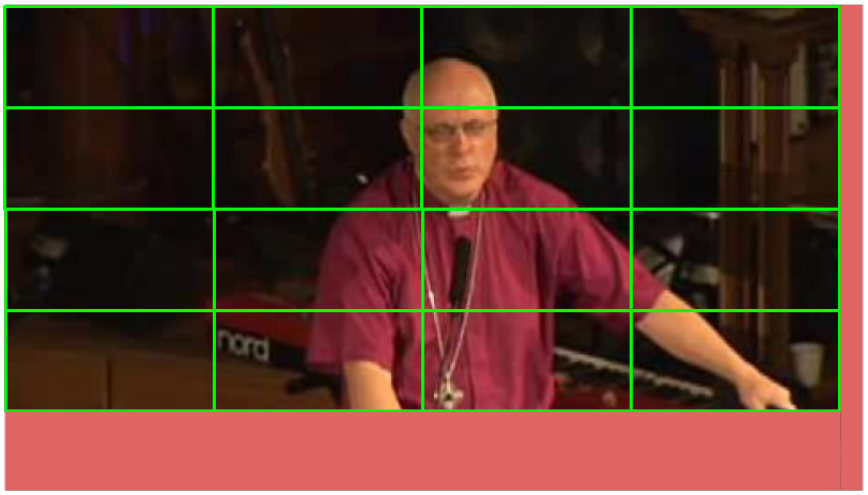

Figure 1.1 – Découpage

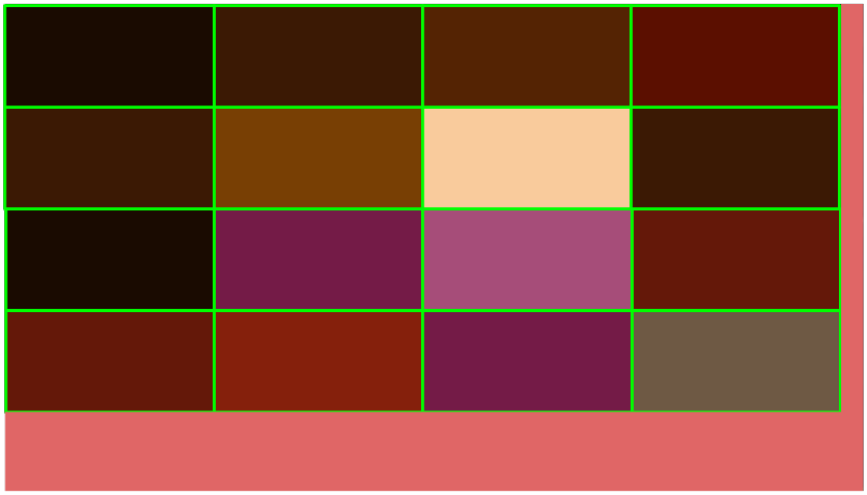

Figure 1.2 – Moyennage

Nous avons voulu utiliser les intensités de couleurs de chaque image pour faire de la classification. Pour éviter d'avoir trop de features nous avons segmenté les images en blocs de tailles égales en moyennant les intensités de couleurs de chaque pixel au sein de chaque groupe. Nous avons rogné des images les bords de droite et d'en bas pour ne pas avoir des blocs de taille différente. Nous avons opté pour cette solution par facilité, en se rendant bien compte que l'on perdait en information.

1.3.4 Reconnaissance faciale sur les `keyframes`

Nous avons trouvé pertinent d'extraire le nombre de visages présents sur une image. La problématique est toutefois simple : « Comment parvenir à reconnaître un visage ? »

En effet, un visage est une représentation complexe, avec des formes et des couleurs qui peuvent différer. Naïvement, il semblerait qu'une bonne méthode soit d'effectuer un test qui renseigne si on a trouvé un visage ou non. Et c'est ce que fait l'algorithme. Il décompose l'image de manière vectorielle, identifie les visages parmis des milliers de petits blocs qu'ils décompose. Il commence en haut à gauche de l'image et descend par micro-bloc de données. Afin de ne pas interroger chaque micro-bloc un par un en s'interrogeant ainsi « est-ce un visage ? », l'algorithme utilise ce que l'on appelle en anglais des « cascades » : il réalise un test rapide pour chaque bloc. S'il est réussi, il effectue un autre test plus complexe et détaillé.

Les « cascades » sont stockées dans des fichiers `.xml` qui contiennent des classifieurs qui ont été préablablement entraînés pour détecter des objets plus ou moins complexes. On initialise notre code avec le classifieur souhaité (main, oeil, visage, silhouette, etc..).

Nous avons utilisé la librairie open source `OpenCV` (Open Computer Vision) pour

ensuite charger ces classifieurs afin de reconnaître des visages.

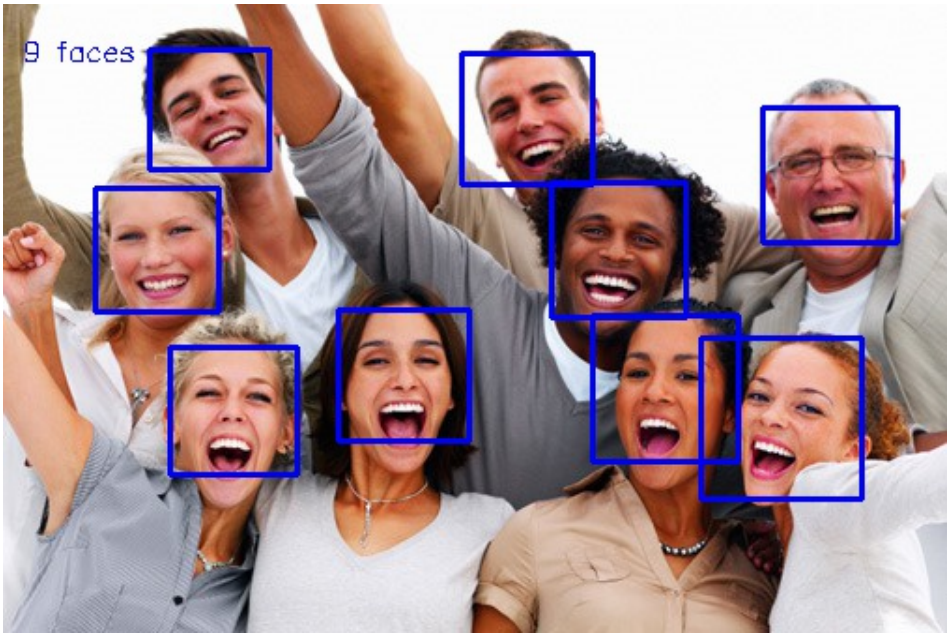

Figure 1.3 – Reconnaissance faciale d'un groupe d'individus

1.3.5 Reconnaissance de texte présent dans une image (OCR)

En observant quelques shots, nous nous sommes aperçus que certains contenaient des portions de texte. Pour pouvoir détecter du texte sur une image, la solution la plus efficace réside en l'utilisation de la Reconnaissance Optique de Caractères (OCR en anglais). Nous avons utilisé `Tesseract` qui est moteur OCR qui a commencé à être développé par Hewlett Packard 1984 et qui a été repris par Google depuis 2005.

Prenons ici une image avec du texte :

Figure 1.4 – Shot avec une portion de texte

Grâce à `tesseract-ocr`, nous obtenons :

SHALOM HARTMAN
INSTITUTE
RABBINIC TORAH
STUDY SEMINAR
JULY 6-16, 2009
Jerusalem, Israel
hartman.org.il

Figure 1.5 – Sortie de `tesseract-ocr`

Nous en avons conclu que `tesseract` était un OCR très performant.

1.4 Extraction du signal audio

1.4.1 Conversion des vidéos en pistes audio

Puisque nous disposions des fichiers vidéos, nous pouvions également faire du traitement audio. Il a donc fallu extraire la partie audio de toutes les vidéos. Nous avons utilisé `ffmpeg` qui est une application de lecture et encodage de la vidéo. Grâce au client en ligne de commande, nous avons converti toutes les vidéos du format `.ogv` en fichier audio (`.wav`).

1.4.2 Extraction de features

Concernant le traitement du signal, nous n'avons pas souhaité nous pencher sur le calcul de la MFCC (Mel-Frequency Cepstral Coefficients). En effet, le vecteurs de coefficients MFCC est davantage utilisé pour faire de la reconnaissance de locuteur. Or, comme nous avions déjà à notre disposition les transcriptions, nous n'avons pas eu besoin de mettre en oeuvre un process de reconnaissance de la parole. Nous nous plus penchés sur l'énergie du signal. Nous avons calculé la racine carrée de l'énergie au carré pour chaque fenêtre. Pour chaque fichier audio, nous obtenons donc un vecteur d'energie. Comme tous ces vecteurs n'ont pas la même taille. Nous avons échantillonné pour obtenir un vecteur de taille n pour chaque piste audio. Nous avons ensuite créé un jeu de donnés destiné aux classifieurs avec en index le nom de la vidéo et autant de colonnes que de valeurs dans le vecteur.

Partie 2

Clustering

Nous n'avons pas passé beaucoup de temps sur l'apprentissage non-supervisé mais nous en avons néanmoins tiré des informations intéressantes. Dès le départ, nous nous somme rendus compte que les features textuelles fonctionnaient le mieux pour la classification, nous nous sommes donc penchés sur ces features là pour l'apprentissage non-supervisé. Nous avons appliqué plusieurs méthodes où à chaque fois nous avons tenté d'identifier 8 groupes.

2.1 Termes les plus fréquent par tag

Dans un premier temps nous avons extrait les termes qui reviennent le plus fréquemment par tag. Ainsi nous pourrons comparer les résultats des clustering avec ceux ci.

Top	1	2	3	4	5	6	7	8	9	10
autos and vehicles	bumper2bumpertv	car	gregmorrison	insight	got	look	new	2009	acura	altima
food and drink	recipe	food	chef	like	tea	drink	visit	episode	cook	wwwlegourmettv
health	health	study	new	disease	jamareport	help	risk	woman	patient	cancer
movies and television	commercial	public	classic	domain	common	advertisement	archiveorg	creative	license	spot
music and entertainment	new	york	city	paparazzo	music	fan	nyc	autograph	tmz	interview
politics	grittv	obama	lauraflanders	politics	political	u	2008	news	grit	lunch
religion	tarot	psychic	reading	bayside	nick	spiritual	free	story	god	lady
sports	dvd	muscle	bodybuilding	receive	posing	nabba	figure	power	like	new

Table 2.1 – Les 10 termes les plus fréquents par tag

On peut voir que les mots revenant le plus souvent par tag sont assez explicite et représentent bien les tags. Par exemple : "muscle" et "bodybuilding" pour le sport, "obama" et "politics" pour politics ou encore "food" et "chef" pour food and drink. Ces résultats nous permettrons d'identifier relativement facilement les classes.

2.2 K-means

Le premier algorithme de clustering que nous avons utilisé est l'algorithme des K-means avec 8 clusters pour les 8 thèmes. L'objectif étant de voir si les clusters obtenus peuvent être associés à des thèmes précis. Vous pouvez voir ci dessous les 10 termes les plus fréquents par cluster.

Top	1	2	3	4	5	6	7	8	9	10
Cluster 0	dvd	bodybuilding	muscle	receive	posing	nabba	figure	workout	olympia	ifbb
Cluster 1	york	city	new	paparazzo	autograph	fan	nyc	tmz	fashion	marc
Cluster 2	grittv	lauraflanders	grit	f	word	commentary	2008	july	flanders	august
Cluster 3	news	ointment	comedy	funny	fake	tatham	clinton	politics	bayside	daily
Cluster 4	tarot	psychic	reading	free	nick	magick	witchcraft	newage	occult	spiritual
Cluster 5	video	study	health	new	music	u	recipe	disease	ha	help
Cluster 6	snack	lunch	political	politicallunch	politics	mccain	tuesday	obama	clinton	look
Cluster 7	commercial	domain	classic	public	advertisement	archiveorg	license	spot	creative	common

Table 2.2 – Les 10 termes les plus fréquents pour les clusters obtenus avec l'algorithme du K-means

On voit très bien ici que le cluster 0 est très lié au thème du sport puisque 8 de ses 10 mots les plus fréquents font également partis des 10 mots les plus fréquents du thème "sport". Avec le même raisonnement, on peut dire que le cluster 1 est probablement lié au thème "music and entertainment". Cependant les autres cluster ne semblent pas être lié à un thème en particulier. Par exemple les termes "lauraflanders" et "obama" sont dans deux clusters différent alors qu'ils appartiennent au même thème. Par conséquent, on peut dire que les thèmes "sports" et "music and entertainment" sont très bien décrit par leurs descriptions et leurs keywords alors que les autres le sont peut être moins.

2.3 Latent Dirichlet Allocation (LDA)

L'allocation de Dirichlet latente est un algorithme permettant de classer des documents par sujet grâce à une factorisation de matrices. Par exemple, si les observations sont les mots collectés dans un document textuel, LDA suppose que chaque document est un mélange d'un petit nombre de sujets ou thèmes, et que la création de chaque mot est attribuable à l'un des sujets du document. Ci-dessous, les 10 mots es plus fréquents par topics :

Top	1	2	3	4	5	6	7	8	9	10
Topic 0	news	obama	tax	clinton	tuesday	politics	political	tea	funny	ointment
Topic 1	commercial	public	domain	classic	creative	spot	license	common	archiveorg	advertisement
Topic 2	clubhousegas	fca	vfestival	video	2009	cancer	family	golf	anglicantv	shrimp
Topic 3	dvd	bodybuilding	muscle	receive	bayside	movie	new	like	weight	tarot
Topic 4	study	film	snack	miracle	charismatic	patient	cardiac	arrest	dubriels	food
Topic 5	grittv	lauraflanders	grit	2008	word	f	july	commentary	vfestival	flanders
Topic 6	city	york	new	health	paparazzo	week	comedy	news	fake	politics
Topic 7	political	politics	lunch	obama	clinton	snack	commentary	man	news	michigan

Table 2.3 – Les 10 termes les plus fréquents pour les topics obtenus avec l'algorithme LDA

2.4 Multiple Correspondance Analysis

Trois ACM ont été effectuées sur les matrices d'occurrences des titres, des mots-clefs et des descriptions pour chaque vidéo. Pour chacune des ACM, une restriction sur les mots a été faite afin d'avoir des mots qui permet de regrouper des différents textes mais pas trop présent pour qu'ils ne représentent pas tous les documents. Un dendrogramme a été fait avec les coordonnées des catégories afin de voir si certaines sont proches.

2.4.1 Les titres

Pour les titres ont garde 79 mots présent dans 1 à 30 pourcent des documents.

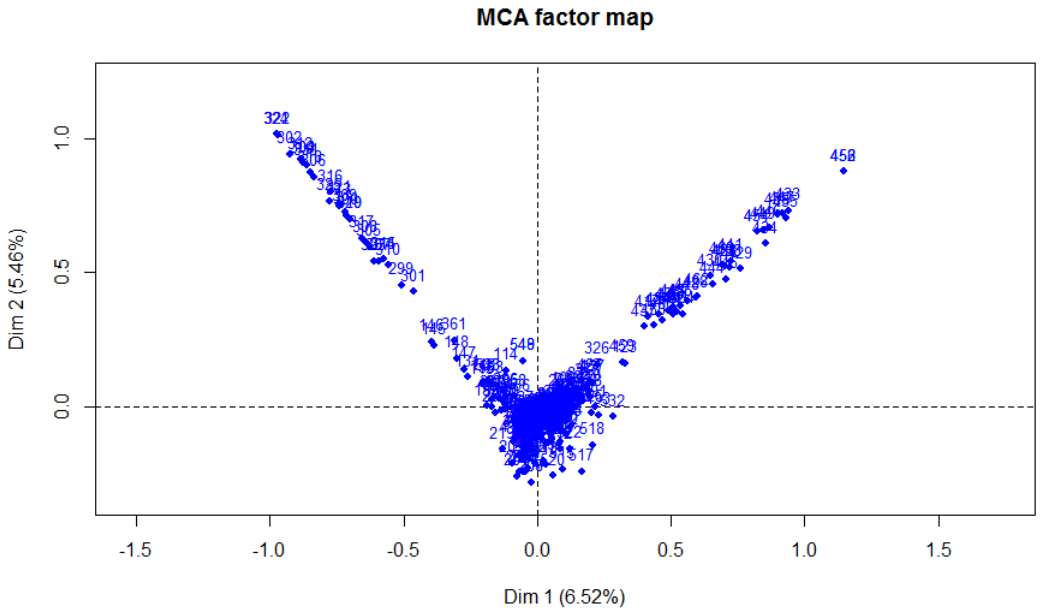

Figure 2.1 – Graphique des individus sur les titres

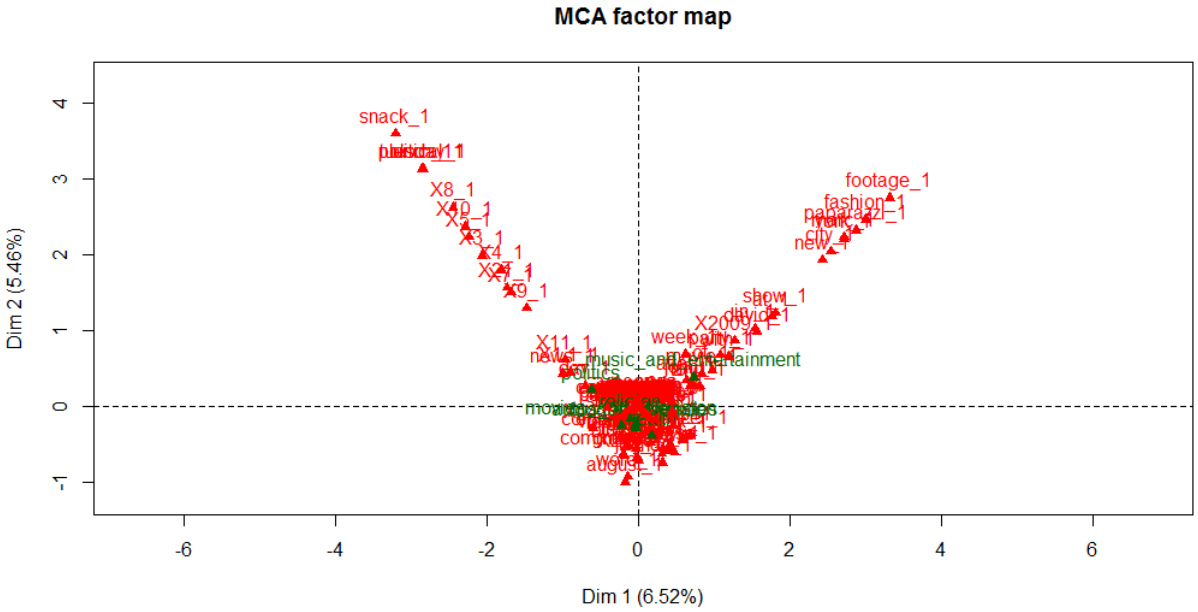

Figure 2.2 – Graphique des variables sur les titres

La catégorie *music_and_entertainment* ressort bien avec par exemple les mots “footage”, “fashion” et “show”. Les documents concernant la politique se démarquent aussi. L’ensemble des autres catégories sont proches dans l’ACM.

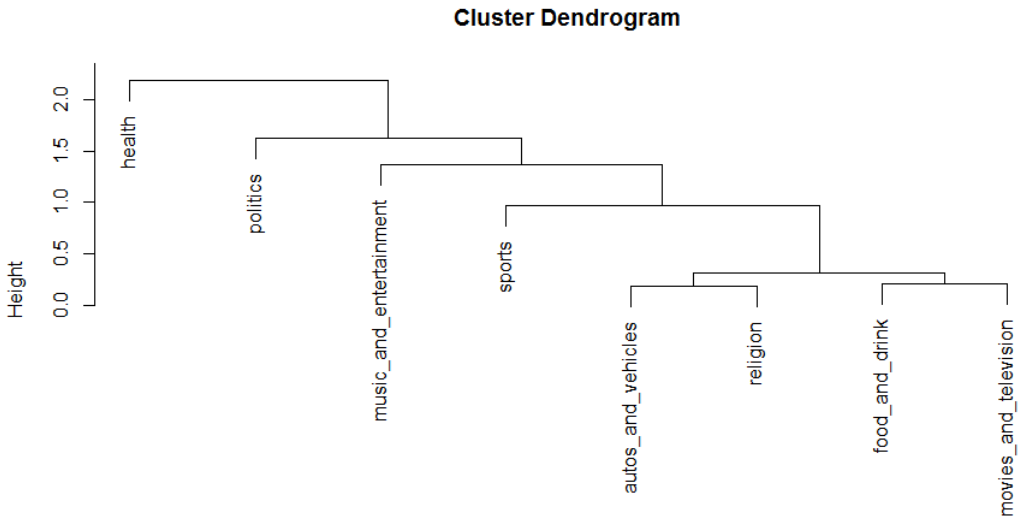

Figure 2.3 – Dendrogramme de la catégorie sur les titres

On remarque que 4 catégories sont très proche, *autos_vehicules*, *religion*, *food_and_drink* et *movies_and_television*. Les autres catégories sont assez différentes les unes des autres.

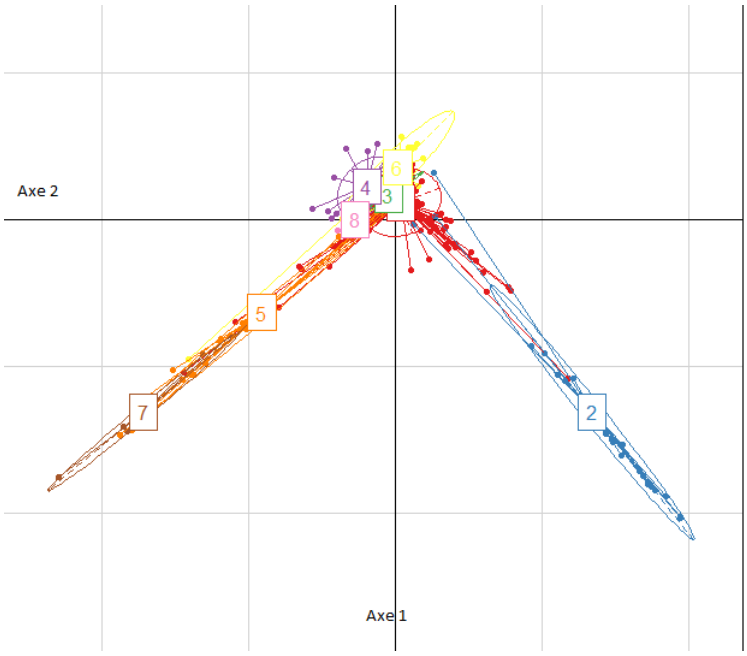

Figure 2.4 – Classification des individus sur les titres

L'estimation par une ACM fait ressortir 5 classes très proches. La catégorie *music* est bien distincte. Deux autres, la politique et le sport sont identifiable. Les titres permettent de faire ressortir les documents concernant la musique, la santé et le sport. Contrairement au retour fait par ACM la politique ne ressort que légèrement.

2.4.2 Keywords

Pour les mots clefs 139 mots sont gardés représentant 1 à 30 pourcent des documents.

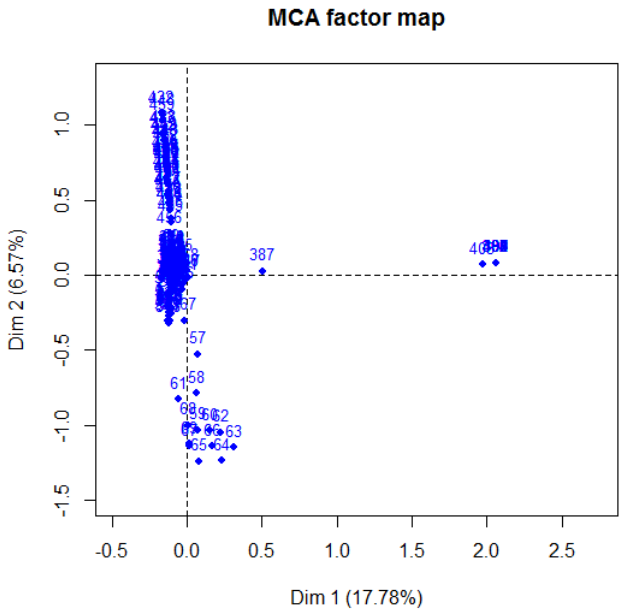

Figure 2.5 – Graphique des individus sur les mots-clefs

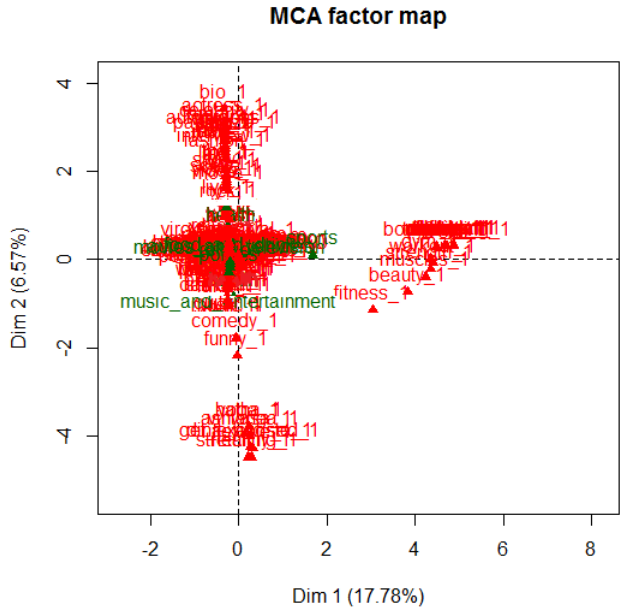

Figure 2.6 – Graphique des variables sur les mots-clefs

3 catégories sont bien différentiables grâce aux mots clefs. Le sport avec “beauty”, “fitness”, “muscles”, la musique avec “comedy”, “funny” et la santée avec “bio” et “live”.

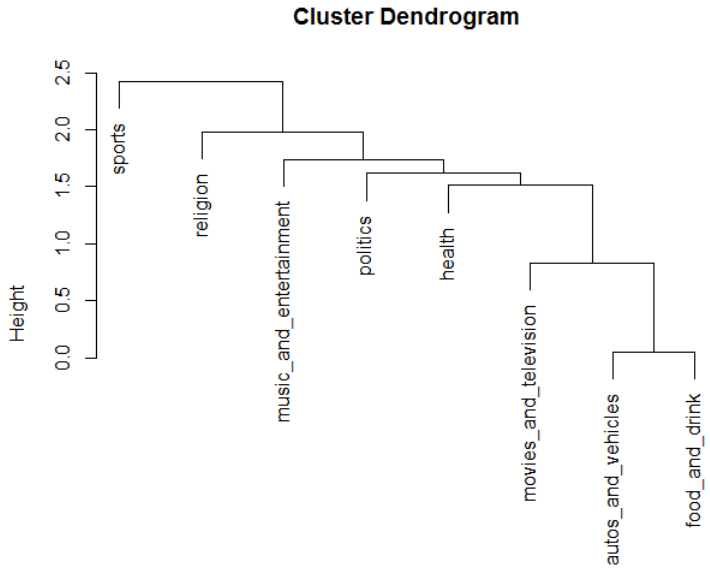

Figure 2.7 – Dendrogramme de la catégorie sur les mots-clefs

On retrouve les catégories *autos*, *food* et *movies* proches et peut différentiables. Comme pour l'ACM les catégories sports, musique et santé sont bien séparables.

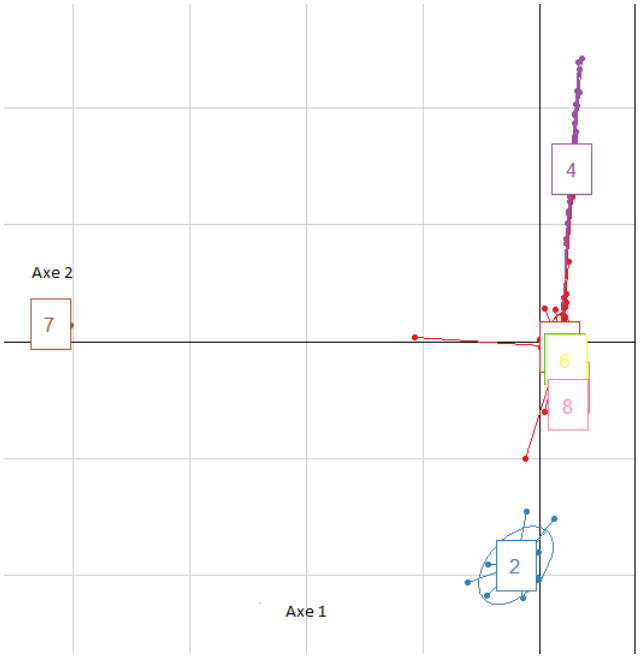

Figure 2.8 – Classification des individus sur les mots-clefs

L'estimation par l'ACM sépare bien une partie des documents de sports, de musique et de la santé. Par contre les autres documents de ces catégories où l'ensemble des

autres documents se regroupent principalement dans une catégorie. Les mots-clefs permettent de faire ressortir une partie des document liés au sport, à la musique et la santé va confondre l'ensemble des autres.

2.4.3 Description

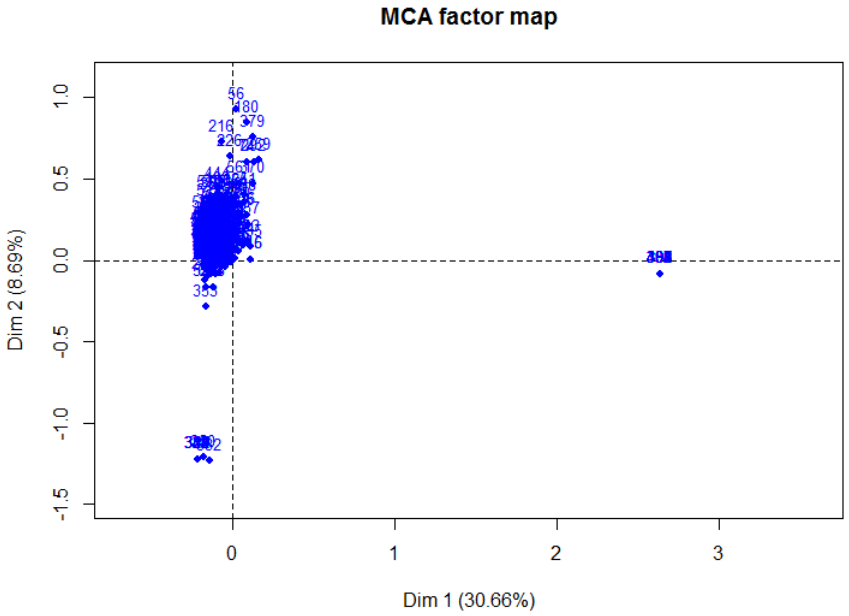

Figure 2.9 – Graphique des individus sur les descriptions

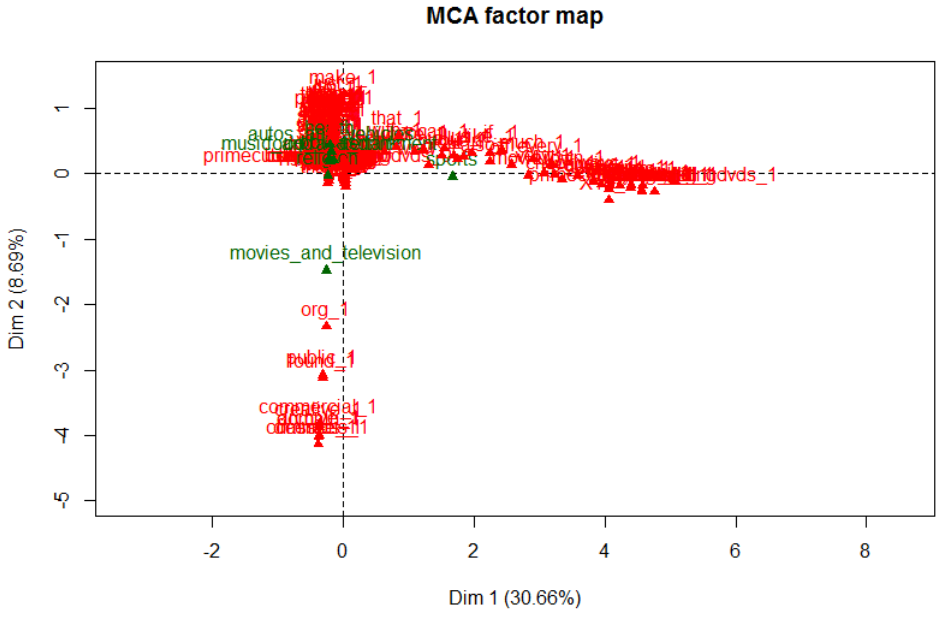

Figure 2.10 – Graphique des variables sur les descriptions

La catégorie *movies_and_television* est bien reconnaissable grâce à des mots comme public, commercial. Une seconde catégorie ressort assez fortement, le sport. Les documents concernant la santé qui même assez proche des autres restes assez différentiables. Les autres catégories ne peuvent pas être séparables.

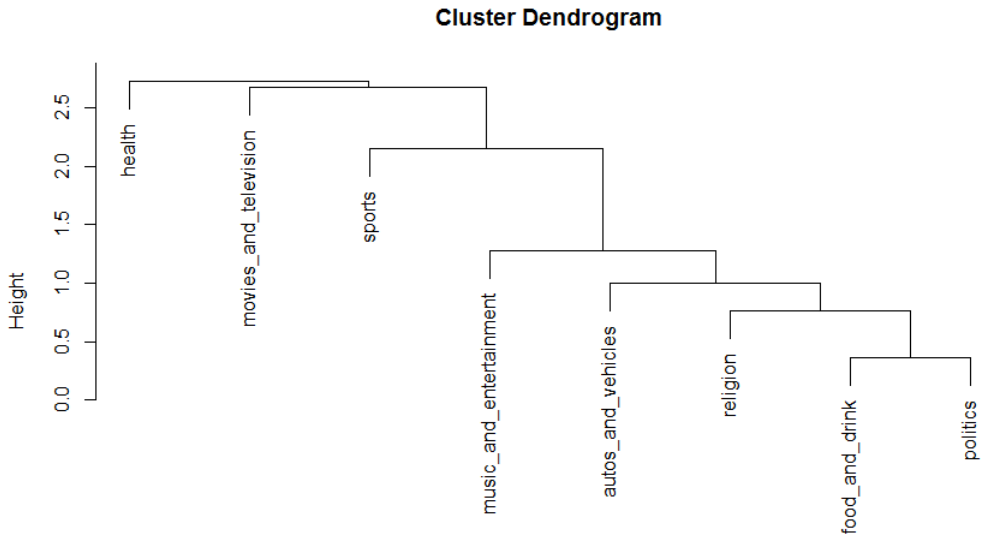

Figure 2.11 – Dendrogramme de la catégorie sur les descriptions

On retrouve les 5 catégories non séparables sont bien très proches. Contrairement aux trois autres qui sont identifiables.

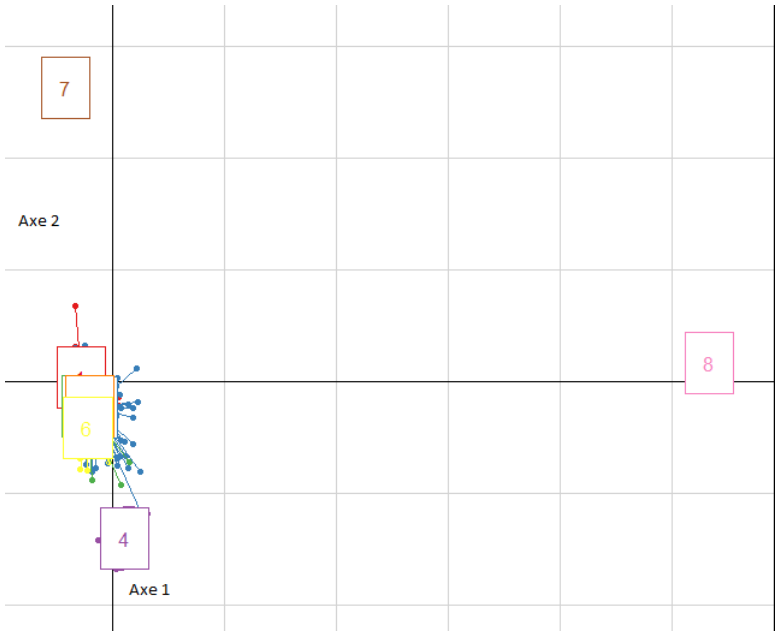

Figure 2.12 – Classification des individus sur les descriptions

La classification par l'ACM par la description fait ressortir 5 groupes proches et 3 bien identifiable. Le sport et les films récupèrent que des documents les concernant sans pour autant les avoir tous. Le groupe de documents associé à la santé est bien distinct mais aussi a ceux relativement proches. Pour l'ensemble des autres documents, ils sont visuellement mélangé. En combinant les trois ACM, on peux bien séparer des documents concernant le sport, les films, la musique, la santé et politiques.

	key	descri	titre	initiale
sports	23	25	29	63
food_and_drink	404	244	415	46
music_and_entertainment	41	7	33	135
health	12	186	15	73
politics	15	24	6	141
autos_and_vehicles	8	8	34	8
religion	47	42	29	45
movies_and_television	25	39	14	64

Figure 2.13 – Tableau récapitulatif du nombres d'affectations de documents par catégorie

On voit que même si grâce aux ACM des documents vont bien être associés à leur catégorie. Par contre les documents n'ayant pas assez de mots significatifs où des catégories qui ne sont pas différentiables vont se rassembler principalement dans une seule catégorie. Entre 20 et 30 documents, sur les 63, sur le sport sont bien reconnaissables. La religion obtient un nombre de documents proches du réel mais comme jamais bien séparable grâce aux ACM on ne peut pas confirmer qu'il en fasse partie. Avec les mots-clefs et titre on peut bien séparer, comme vu dans les ACM, une partie des documents concernant la musique. Deux tiers des documents sur les films sont identifiables grâce au texte de la description. Par contre à l'inverse pour la catégorie sur la cuisine récupère jusqu'à dix fois plus de documents. Ce sont principalement les documents proches de plusieurs catégories.

Partie 3

Classification

3.1 Classifieurs implémentés

3.1.1 Naïve Bayes sur les matrices d'occurrences de termes

La méthode Naïve Bayes est une méthode populaire pour classifier du texte. Dès le départ nous avons eu de bons résultats en classifiant par rapport aux matrices d'occurrences de termes. Nous sommes allés plus loin en effectuant du bagging pour gagner en robustesse.

3.1.2 Classifieur se basant sur les termes les plus fréquents

Nous avons développé un classifieur "fait maison" où nous utilisons la distance de Jacard entre les mots les plus fréquents associés à un tag et les mots présents dans un document. Le partie apprentissage calcule les n mots les plus fréquent pour un groupe donné. Ceci ce fait très facilement en effectuant une agrégation par rapport à chaque tag puis en cherchant les plus fréquents. Bien que ce classifieur ne soit pas aussi performant qu'un SVM ou un Naïve Bayes, il n'en demeure pas moins bien plus rapide à entraîner.

3.1.3 KNN sur les matrices d'occurrences de termes

Nous avons aussi utilisé un KNN car il marche très bien sur les matrices d'occurrences. Intuitivement un KNN va exhiber les directions que des documents ont en commun. Sur une matrice d'occurrences directions sont les termes. Nous aurions pu aller plus loin et appliquer un KNN aux composantes principales de la matrice d'occurrences résultantes d'une ACP mais nous n'avons pas eu le temps et le gain à gagner n'était pas évident.

3.1.4 SVM sur les matrices d'occurrences de termes

Puisque le nombre de document est assez, le nombre de features est relativement bien plus élevé ($p >> n$) ; ceci fait que l'espace est potentiellement linéairement séparable. Il est donc naturel qu'un SVM à noyau linéaire fonctionne correctement. Les SVM sont très utilisés pour la classification pour cette raison. Lors de nos essais c'est principalement la méthode du SVM qui a le mieux fonctionné.

3.1.5 Random Forest sur les features globales

Après le travail de récupération de données effectué par chacun d'entre nous, nous avons décidé de combiner des features provenant de différentes sources et d'utiliser un algorithme de `Random Forest` avec. Les features que nous avons utilisé sont les suivantes :

- A propos des shots :
  - *has_text* : "true" si la vidéo a des shots comportant du texte, "false" si non.
  - *nb_faces_max* : Donne le nombre maximum de visages détectés sur les shots de la vidéo
  - *nb_shot* : Nombre de shot(s) par vidéo
- A propos des speakers (*cf.* 1.2) :
  - *entropy*
  - *nb_M*
  - *nb_F*
  - *Freq_M/F*
- A propos des métadonnées :
  - *duration* : durée de la vidéo
  - *size* : taille de la vidéo
  - *uploader_id* : ID de l'uploader de la vidéo

3.1.6 KNN sur les histogrammes de couleurs des shots

Grâce au fichier *hist_data.json* que nous avons généré, nous avons les histogrammes de couleurs de chaque shot de chaque vidéo. L'idée ici est de comparer chaque histogramme à tous les autres pour en ressortir les plus proches.

La première chose que nous avons fait était de normaliser les histogrammes. En effet, les shots ne sont pas tous de la même taille et par conséquent ne possède pas le même nombre de pixel. Or cela n'a pas de sens de comparer des histogrammes sur le nombre de pixel par couleur si ces images ne sont pas de la même taille. Une fois normaliser, les histogrammes donnent donc le pourcentage de pixel par couleur.

Ensuite, nous avons comparer ces histogrammes en regardant combien donnait l'intersection des histogrammes. Si celle si est élevé, alors les histogrammes sont proches et inversement. Ainsi, par image, nous avons récupérer les 9 images les plus proches d'elles.

Une fois ces images récupérées, nous avons extrait les tags liés à chacune de ces images. Le tag revenant le plus souvent dans ces 9 tags, est notre prédiction pour

l'image traité. Etant donné que chaque vidéo possède plusieurs shots, on récupère les tags prédit pour chaque shots d'une vidéo, et le tag revenant le plus souvent est notre prédiction pour la vidéo.

3.1.7 Réseau de neurones sur les images découpés en blocs

Nous avons réalisé un traitement d'image avec le réseau de neurones afin de classifier les images, c'est-à-dire de prédire les tags qui leurs sont associés.
Les réseaux de neurones sont des modèles très performants pour classer des images. Le fonctionnement d'un réseau de neurones et sa capacité d'apprentissage, s'explique dans un premier temps par l'apprentissage. En entrée une image est fournie sous la forme d'une matrice vectoriel de pixels à 3 dimensions pour une image en couleur[Rouge, Vert, Bleu].

Donc il nous faut réaliser un traitement d'image pour appliquer ce classifieur puisque les images que nous allons traiter sont de formats différents, dont chaque pixel est codé sur 8 bits. La liste de vecteurs de même taille pour chaque images est générée après l'application de la méthode de Moyennage dans chaque bloc (cf. 2.3.2 Séparation en blocs)

L'augmentation du taux de compression (dû à l'effet de blocs) est suivie par une dégradation de la qualité des images reconstruites et de la qualité des images (surtout au niveau des contours de l'image).
Par conséquent la qualité de l'image fourni par sa matrice vectoriel que prend le classifieur en entrée n'est pas assez optimal pour que le système "apprend" et prédit correctement les tags associés à ces images.

3.1.8 Réseau de neurones sur l'énergie du signal sonore

A l'étape d'extraction de données, concernant le traitement audio, nous avions obtenu un jeu de données composé d'un index basé sur le nom des vidéos, et il comportait autant de colonnes que de valeurs présentes dans les vecteurs d'énergie. Nous avons essayé de mettre en place un réseau de neurones. Les résultats obtenus ont été assez décevants car nous avions que 25% de précision.

3.2 Classification par agrégation de modèles hétérogènes

3.2.1 Récapitulatif des données extraites

1. La matrice TF des tags et des descriptions lemmatisés contenus dans les métadonnées
2. La matrice TF-IDF extraite du jeu de données 1
3. La matrice TF des termes ayant un taux de confiance supérieur à 80% contenus dans les transcriptions

4. La matrice TF-IDF extraite du jeu de données 3
5. L'information sur la distribution de la parole des locuteurs
6. L'information sur les features contenus dans les images
7. Les histogrammes de couleurs pour chaque image 255 valeurs
8. Les intensités RGB moyenne des 10×30 blocs pour chaque image
9. L'énergie du signal sonore pour chaque vidéo

3.2.2 Récapitulatif des classifieurs

Après avoir fait de la recherche et du grid search dans des notebooks Jupyter, nous avons retenus les classifieurs suivants :

1. Bagging de Naïve Bayes sur le jeu de données 1
2. "Top terms" sur le jeu de données 1
3. KNN sur le jeu de données 1
4. SVM linéaire sur le jeu de données 2
5. Bagging de Naïve Bayes sur le jeu de données 3
6. "Top terms" sur le jeu de données 3
7. KNN sur le jeu de données 3
8. SVM linéaire sur le jeu de données 4
9. Random Forest sur les jeux de données 5 et 6 concaténés
10. KNN sur le jeu de données 7
11. Réseau de neurones sur le jeu de données 8
12. Réseau de neurones sur le jeu de données 9
13. Réseau de neurones sur le jeu de données 9
14. Réseau de neurones sur le jeu de données 9

3.2.3 Évaluations individuelles

Comme convenu, nous effectuons une validation croisée *leave one out* (LOO). C'est à dire que l'on entraîne nos modèles sur $n-1$ observations et on prédit la classe de la ligne restante. En LOO, les jeux de données d'entraînement se superposent à $\frac{n-1}{n}$%, ce qui fait qu'ils sont corrélées et qu'ils surestiment la variance des classifieurs entraînés dessus. Nous avons donc décidé d'évaluer les classifieurs entraînés en calculant la précision, le rappel et le F1-score sur les prédictions. Nous avons aussi calculé le temps consacré à l'entraînement de chaque classifieur, bien que celui ne rentre en jeu dans l'évaluation finale il n'en demeure pas moins intéressant.

Table 3.1 – Résultats pour les classifications individuelles

	Precision	**Mean training time**
Naïve Bayes on metadata TF	0.839	0.697
Top terms on metadata TF	0.682	0.356
KNN on metadata TF	0.822	0.285
Linear SVM on metadata TF-IDF	0.879	0.287
Naïve Bayes on transcription TF	0.548	0.372
Top terms on transcription TF	0.354	0.313
KNN on trans TF	0.654	0.294
Linear SVM on trans TF-IDF	0.665	0.269
Random Forest on speakers and shots features	0.661	0.241
Neural network on signal energy	0.249	0.231

3.2.4 Méta-classification

Lors des évaluations individuelles des classifieurs, nous avons stocké les prédictions de chaque classifieurs dans un tableau de données. Le tableau a donc autant de lignes que de documents et autant de colonnes que de classifieurs. Nous avons ensuite lancer un Random Forest sur ce tableau pour tenter d'obtenir un meilleur score global. Derrière cette approche, l'intuition est que les classifieurs vont "voter" pour la classe qu'ils ont prédit et la classe avec le plus votes l'emportera.

Grâce à cette méthode nous avons gagné 0.6% de précision par rapport à notre meilleur classifieur avec au final 88.5% de précision. Ça n'est pas beaucoup mais c'est probablement du au fait que le classifieurs classifie les documents de la même façon et donc n'apporte pas beaucoup d'informations en plus de celle du meilleur classifieur. Intuitivement il n'est pas très probable que le meilleur classifieur se trompe alors 3 autres classifieurs plus faibles ait raison pour une observation.

Nous pensons que cette façon de faire peut porter ses fruits si on ajoute des classifieurs plus variés. En effet 8 des 10 classifieurs que nous avons utilisés se base sur le texte. Hélas nous n'avons pu eu le temps d'en intégrer plus pour d'autres médias. En tout cas nous pensons que cette approche est intéressante et nous sommes satisfait d'au moins avoir pu expérimenter avec.

Partie 4

Organisation

Nous n'avons pas assigné de membre de groupe à une seule thématique pour l'ensemble du projet ; nous avons souhaité faire en sorte que chaque puisse membre puisse travailler sur des choses qui lui plaisent et donc il était fréquent que l'on échange parfois nos tâches avec d'autres membres. Cependant, les thèmes récurrents pour chaque membre ont été :

- Salima qui s'est plus occupé de la partie traitement des images.
- Axel qui s'est occupé du traitement des images mais du traitement du signal
- Max qui s'est concentré sur l'apprentissage non-supervisé et la partie textuelle
- Joseph qui a plus fait de l'analyse exploratoire
- Giovanni qui a aussi fait du traitement d'image mais qui a aussi analysé la distribution de la parole dans les transcriptions

En plus de cela, Axel et Max se sont occupés d'intégrer le code des uns et des autres dans l'outil final.

Nous avons nommé l'outil final *Karadoc*. Il est disponible sur le GitHub du CMISID (https://github.com/cmisid/Karadoc) où une description de l'outil est disponible. L'extraction est écrite en Python. Les analyses sont des notebooks IPython.

Pour l'assignation des tâches nous n'avons pas trouvé cela forcément utile d'utiliser Trello. Nous nous y retrouvions en discutant à l'oral durant les séances de travail. Nous avons néanmoins utilisé un tableau Kanban disponible sur GitHub comme bloc-notes.

Nous avons utilisé Slack pour la communication et le partage des fichiers.

Nous avons principalement utilisé Python pour réaliser ce projet. Nous avons utilisé les librairies suivantes :

- *scikit-learn* pour les algorithmes d'apprentissages
- *BeautifulSoup* pour le parsage de fichiers XML
- *Tesseract* pour la reconnaissance de texte
- *OpenCV* pour la reconnaissance de visages
- *NLTK* pour l'analyse textuelle
- *pandas* pour la manipulation de tableaux de données

Conclusion

Ce projet nous a permis de travailler sur un vrai challenge demandant plusieurs compétences dû à la diversité des sources de données (audio, vidéo et texte). Grâce à une organisation solide, nous avons su répartir les tâches entre les différents membres du groupe de telle façon à ce que chaque membres travail plus ou moins sur chaque source de données.

Nous avons pu également mettre en pratique nos connaissances en statistique et même les développer en utilisant des classifieurs que nous connaissions bien (KNN, Naïve Bayes, SVM) et d'autres que nous connaissions moins bien (Random Forest, Réseau de neurones).

Le seul point négatif que nous pouvons évoquer est notre manque de connaissance. Nous ne nous sommes pas sentis très à l'aise par rapport au traitement des images. Bien que nous ayons fait quelques exercices en TP, nous n'avions pas les bonnes intuitions pour savoir quelles features extraire. Au contraire, nous nous sommes un peu plus appuyés sur les données textuelles avec lesquelles nous avions plus d'expérience. Nous n'avons pas fait dans un souci facilité ou de performance, c'est simplement une préférence.

Nous sommes ravi d'avoir fait un projet avec autant de différents types de données. Nous nous sommes bien rendus compte que l'extraction de données pertinentes est aussi importante voir plus que la classification. En effet bien qu'on puisse torturer un algorithme d'apprentissage pour gratter quelques pourcent, il est évident qu'avoir des features pertinentes fait l'essentiel du travail.

Table des figures

Liste des tableaux