



DataScientest • com

Rapport Technique d'évaluation



Promotion : Aout 2021, formation continue

Participants :

Lionel Bérenger

Nicolas Freychet

(Eva Lefort, Sora Derraz)

Chef de cohorte :

Frédéric Francine

Table des matières

Contexte.....	2
Objectifs.....	2
Data.....	3
Jeux de données.....	3
1- context_content_features.csv (2,19Go)	3
2- sentiment_values.csv (373Ko)	4
3- user_track_hashtag_timestamp.csv (1,22Go)	4
Aperçu des données.....	5
Relations, analyse et nettoyage des données.....	6
1- context_content_features.csv	7
2- sentiment_values.csv.....	8
3- user_track_hashtag_timestamp.csv	9
Constitution d'un dataframe de travail	10
Projet.....	11
Classification du problème.....	11
Choix des modèles & Optimisation.....	11
○ <i>Modèle 1 : Popularité et analyse de sentiments (Régression)</i>	11
○ <i>Modèle 2 : Recommandation par clustering sur les caractéristiques musicales</i>	16
Modèle K-Means.....	17
Visualisation des clusters.....	18
Evaluation du modèle	21
Test sur un autre jeu de données :	25
○ <i>Modèle 3 : Recommandation par similarités sur le champ lexical</i>	27
○ Conclusion sur les méthodes de clustering.....	31
Description des travaux réalisés	32
Répartition de l'effort sur la durée et dans l'équipe	32
Bibliographie	32
Difficultés rencontrées lors du projet.....	33
Bilan & Suite du projet.....	33

Contexte

Ce projet nous a permis de mettre en œuvre diverses techniques de Machine Learning pour bâtir un système de recommandations (ici de morceaux de musique) en fonction des goûts des utilisateurs. Il s'agit de se rapprocher de l'usage quotidien des utilisateurs pour en extraire les informations, et offrir la meilleure expérience possible. C'est un enjeu important dans un contexte de concurrence entre différentes plateformes de diffusion. Plus les recommandations sont pertinentes et plaisent aux utilisateurs, plus la plateforme sera populaire.

Objectifs

Le titre et la description du projet permettaient plusieurs approches possibles :

Titre : Recommandations musicales – Prédire les musiques les plus appréciées sur Twitter

Description : Prédire si une musique sera plus ou moins appréciée à partir d'analyses et retours Twitter.

Prédire si une musique sera plus ou moins appréciée :

Une première approche est de suggérer des morceaux de musiques en fonction de leur critère d'appréciation, le dataset comportant des dictionnaires de sentiments. En explorant cette piste (régression), nous avons constaté que les données ne permettaient cependant pas de prédire les musiques plus appréciées (score de sentiment) ni les plus populaires (les plus partagées).

Recommandations musicales :

Une autre approche est de bâtir un système de recommandation en fonction des spécifications musicales (clustering) ou des hashtags renseignés. Cette piste nous mena plus loin, bien que les données ne permettent pas d'avoir un rendu très satisfaisant.

En effet, si des groupes ou des recommandations ont pu être faites, le dataset ne contient aucun nom de morceau, seulement des identifiants « track_id », il nous a donc fallu faire confiance aux hashtags renseignés par les utilisateurs pour évaluer notre modèle.

De plus, il nous était impossible de mesurer la performance d'un tel modèle.

Ce rapport explore ces deux approches.

Data

Jeux de données

Le jeu de données utilisé provient d'un projet kaggle, disponible publiquement :

<https://www.kaggle.com/chelseapower/nowplayingrs>

Il se compose de 3 fichiers csv séparés, apportant des informations complémentaires.

1- context content features.csv (2,19Go)

Ce jeu de données de 22 variables et 11 614 671 entrées présente, pour différentes pistes de musiques référencées par des utilisateurs, des caractéristiques musicales telles que « instrumentalness », « liveness », ... sous forme de valeur numérique.

Chaque entrée contient de plus le contexte de mise en ligne (localisation et date, langue, identifiant de l'utilisateur, ...).

Variables

<u>coordinates</u> : données GPS [float64, 38058 non-null]	<u>mode</u> : valeur égale à 0 ou 1, majeur ou mineur (1 = majeur et 0 = mineur) [float64, 11611957 non-null]
<u>instrumentalness</u> : contient ou non du chant [float64, 11611869 non-null]	<u>key</u> : tonalité "pitch class notation" (entier de 0 à 11, 0 = do, 1 = do#/réb, 2=ré, ...,11=si) [float64, 11611957 non-null]
<u>liveness</u> : valeur entre 0 et 1. probabilité que l'enregistrement soit en live (1 = forte probabilité de live) [float64, 11611757 non-null]	<u>artist_id</u> : identifiant de l'artiste [object, 11614671 non-null]
<u>speechiness</u> : valeur entre 0 et 1, balance entre discours et musique (0 = sans parole) [float64, 11610783 non-null]	<u>place</u> : dictionnaires de données type json, contenant des informations utilisateurs, url api twitter, ... énormément de données nulles (inexploitable) [object, 44344 non-null]
<u>danceability</u> : potentiel de la chanson à faire danser (valeur entre 0 et 1, 1 = potentiel élevé) [float64, 11610783 non-null]	<u>geo</u> : coordonnées GPS [object, 38058 non-null]
<u>valence</u> : caractère positif de la chanson (valeur entre 0 et 1, 1 = très positif et joyeux) [float64, 11609883 non-null]	<u>tweet_lang</u> : langue utilisée pour écrire le tweet [object, 11614671 non-null]
<u>loudness</u> : intensité en décibel (valeur en dB) [float64, 11614671 non-null]	<u>track_id</u> : identifiant de la chanson [object, 11614671 non-null]
<u>tempo</u> : tempo en beats per minute (valeur en BPM) [float64, 11614671 non-null]	<u>created_at</u> : date de création du tweet [object, 11614671 non-null]
<u>acousticness</u> : valeur entre 0 et 1, probabilité que l'enregistrement soit acoustique. (1 = forte probabilité d'enregistrement acoustique) [float64, 11611884 non-null]	<u>lang</u> : paramètre langue qui semble être en rapport avec l'utilisateur (et non avec le morceau de musique) [object, 11614607 non-null]
<u>energy</u> : caractère intense et actif de la chanson (valeur entre 0 et 1, 1 = chanson très énergique) [float64, 11611910 non-null]	<u>time_zone</u> : indication de la position géographique (fuseau horaire, temps UTC, nom du pays) [object, 8353946 non-null]
	<u>user_id</u> : identifiant de l'utilisateur [float64, 11570327 non-null]
	<u>id</u> : probable identifiant du tweet [int64, 11614671 non-null]

2- sentiment values.csv (373Ko)

Ce jeu de données de 21 variables et 5290 entrées met en relation des hashtags (mots-clés) et des scores de sentiments, indiquant ainsi si un hashtag est associé à un sentiment plutôt positif ou négatif. Il présente 20 variables descriptives + 1 variable "cible" (hashtag).

Les variables descriptives sont basées sur différents lexiques de sentiments (AFINN, Opinion Lexicon, Sentistrength Lexicon, Vader) et leurs variabilités (minimum, maximum, somme et moyenne).

Tous les lexiques de sentiments attribuent à un hashtag un score entre 0 et 1, du sentiment le plus négatif au positif. Ainsi, en utilisant les hashtags renseignés pour un morceau de musique, il serait possible de savoir si ce morceau de musique dégage un sentiment plutôt positif ou plutôt négatif.

Variables

hashtag : [object, 5290 non-null]
vader_min : [float64, 3867 non-null]
vader_max : [float64, 3867 non-null]
vader_sum : [float64, 3867 non-null]
vader_avg : [float64, 3867 non-null]
afinn_min : [float64, 255 non-null]
afinn_max : [float64, 255 non-null]
afinn_sum : [float64, 255 non-null]
afinn_avg : [float64, 255 non-null]
ol_min : [float64, 2635 non-null]
ol_max : [float64, 2635 non-null]
ol_sum : [float64, 2635 non-null]
ol_avg : [float64, 2635 non-null]
ss_min : [float64, 2823 non-null]
ss_max : [float64, 2823 non-null]
ss_sum : [float64, 2823 non-null]
ss_avg : [float64, 2823 non-null]

X_min : [float64, 2160 non-null]
X_max : [float64, 2160 non-null]
X_sum : [float64, 2160 non-null]
X_avg : [float64, 2160 non-null]

Les quatre dernières colonnes ci-dessus n'étaient pas explicitement nommées ou associées à un dictionnaire de sentiment.

Cependant, elles ont la même structure que les autres types de colonnes (minimum, maximum, somme et moyenne), et possèdent une forte corrélation entre elles, nous en déduisons qu'il s'agit d'un 5^{ème} dictionnaire de sentiments sans nom (que nous avons nommé X_...).

Cf matrice de corrélation dans la partie « Nettoyage des données ».

3- user track hashtag timestamp.csv (1,22Go)

Ce jeu de données de 4 variables et 17 560 113 entrées relie les morceaux de musiques à différents hashtags. Chaque piste de musique (identifiée par le « track_id ») est associée à un ou plusieurs hashtags par le ou les utilisateurs.

Un même morceau de musique peut être renseigné plusieurs fois (jusqu'à 100 000 entrées pour un même « track_id »).

Variables

user_id : identifiant de l'utilisateur
[int64, 17560113 non-null]
track_id : identifiant de la chanson
[object, 17560113 non-null]

hashtag : mot clé
[object, 17560112 non-null]
created_at : date de création du tweet
[object, 17560113 non-null]

Aperçu des données

sentiment_values

	2	5
hashtag	greatmusic	againbecauseitissogood
vader_min	0.8	0.7
vader_max	0.8	0.7
vader_sum	2.4	0.7
vader_avg	0.8	0.7
afinn_min	1.0	1.0
afinn_max	1.0	1.0
afinn_sum	1.0	1.0
afinn_avg	1.0	1.0
ol_min	0.8875	0.7375
ol_max	0.8875	0.7375
ol_sum	0.8875	0.7375
ol_avg	0.8875	0.7375
ss_min	1.0	1.0
ss_max	1.0	1.0
ss_sum	1.0	1.0
ss_avg	1.0	1.0
X_min	0.8	0.8
X_max	0.8	0.8
X_sum	0.8	0.8
X_avg	0.8	0.8

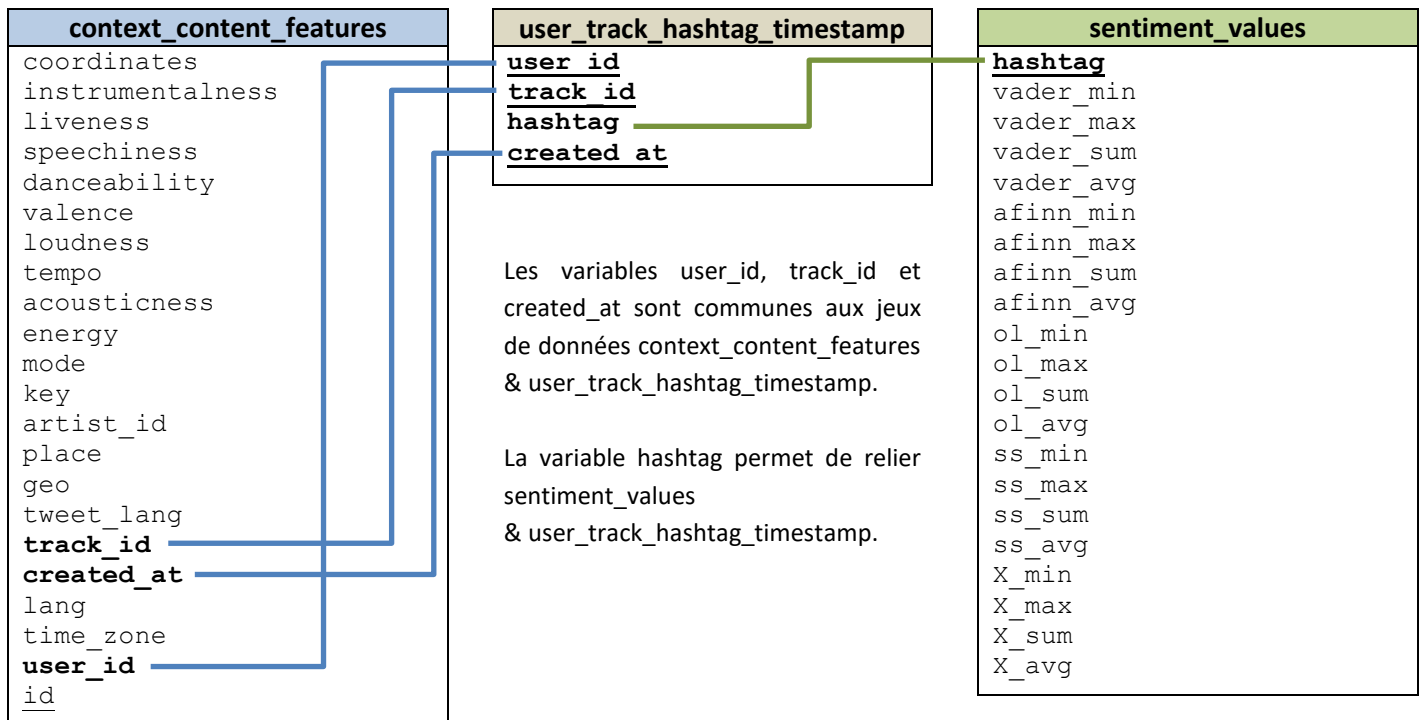
context_content_features

	116	745
coordinates	{u'type': u'Point', u'coordinates': [114.01848...	{u'type': u'Point', u'coordinates': [13.119993...
instrumentalness	0.0	0.0
liveness	0.141	0.251
speechiness	0.0264	0.0316
danceability	0.449	0.683
valence	0.318	0.654
loudness	-4.8	-6.912
tempo	170.04	98.981
acousticness	0.149	0.163
energy	0.602	0.601
mode	0.0	1.0
key	6.0	7.0
artist_id	19713e433ee96b5b0e7ddf7accdbf3b9	d7e3dd80f6f0f13230cd1d13025196ea
place	{u'name': u'Mini', u'url': u'https://api.twitt...	{u'name': u'Potsdam', u'url': u'https://api.tw...
geo	{u'type': u'Point', u'coordinates': [4.4107847...	{u'type': u'Point', u'coordinates': [52.381905...
tweet_lang	en	en
track_id	99e17f3adedc87044d5e569de0f38504	31f69e3f6d30dcde0d05e407aee1be3f
created_at	2014-01-01 05:58:41	2014-01-01 06:31:49
lang	en	de
time_zone	Beijing	Berlin
user_id	NaN	NaN
id	826865180	160874621

user_track_hashtag_timestamp

user_id	track_id	hashtag	created_at
81496937	cd52b3e5b51da29e5893dba82a418a4b	nowplaying	2014-01-01 05:54:21
81496937	cd52b3e5b51da29e5893dba82a418a4b	goth	2014-01-01 05:54:21
81496937	cd52b3e5b51da29e5893dba82a418a4b	deathrock	2014-01-01 05:54:21
81496937	cd52b3e5b51da29e5893dba82a418a4b	postpunk	2014-01-01 05:54:21
2205686924	da3110a77b724072b08f231c9d6f7534	NowPlaying	2014-01-01 05:54:22

Relations, analyse et nettoyage des données



Cohérence des données :

Hashtags (mots clés) :

Les hashtags peuvent être similaires mais écrits légèrement différents, avec des majuscules ou non, ce qui perturbe la qualité des données. C'est pourquoi nous les avons convertis en minuscule. Certains hashtags sont sans intérêt : « now playing, nowplay, np, music, onlineradio, radio, ... », ils seront supprimés.

Le fichier `user_track_hashtag_timestamp` comporte beaucoup plus de hashtags que le fichier `sentiment_values` : 42 412 hashtags distincts dans `user_track_hashtag_timestamp` contre 5 290 hashtags présents et distincts dans le fichier `sentiment_values`.

Ceci laisse supposer que le fichier `sentiment_values` a été figé (afin de générer les lexiques / scores de sentiment) alors que le fichier `user_track_hashtag_timestamp` a continué d'être alimenté par la suite.

Toutefois, un utilisateur renseigne généralement plusieurs hashtags pour un même morceau, ce qui génère plusieurs entrées pour le même utilisateur & morceau dans le fichier `user_track_hashtag_timestamp`. Ainsi nous avons constaté que les tuples (`user_id`, `track_id`, `created_at`) de `user_track_hashtag_timestamp` sans hashtag identifié dans `sentiment_values` sont assez rares en réalité : il n'y a que 2188 entrées (`user_id`, `track_id`, `created_at`) réellement isolées (sans hashtag équivalent dans `sentiment_values`), ceci représente 0.01% des entrées totales du fichier `user_track_hashtag_timestamp`.

Tuple (`user_id`, `track_id`, `created_at`) :

Sur les 11 614 671 entrées du fichier `context_content_features` :

- 11 293 889 (97,24%) concernent des utilisateurs `user_id` présents dans le fichier utilisateur `user_track_hashtag_timestamp`
- 11 593 775 (99,82%) concernent des morceaux `track_id` présents dans le fichier `user_track_hashtag_timestamp`
- 11 271 167 (97,04%) concernent des dates `created_at` présentes dans le fichier utilisateur `user_track_hashtag_timestamp`

Au final, seules 10 797 entrées (0,09% des entrées du fichier) du tuple (`user_id`, `track_id`, `created_at`) ne sont pas présentes dans le fichier utilisateur `user_track_hashtag_timestamp`, tout le reste est « cohérent ».

1- context content features.csv

	nb null	% null
coordinates	11576613	99.67
geo	11576613	99.67
place	11570327	99.62
time_zone	3260725	28.07
user_id	44344	0.38
valence	4788	0.04

Certaines colonnes contiennent trop de valeurs nulles pour être exploitées, elles sont supprimées :

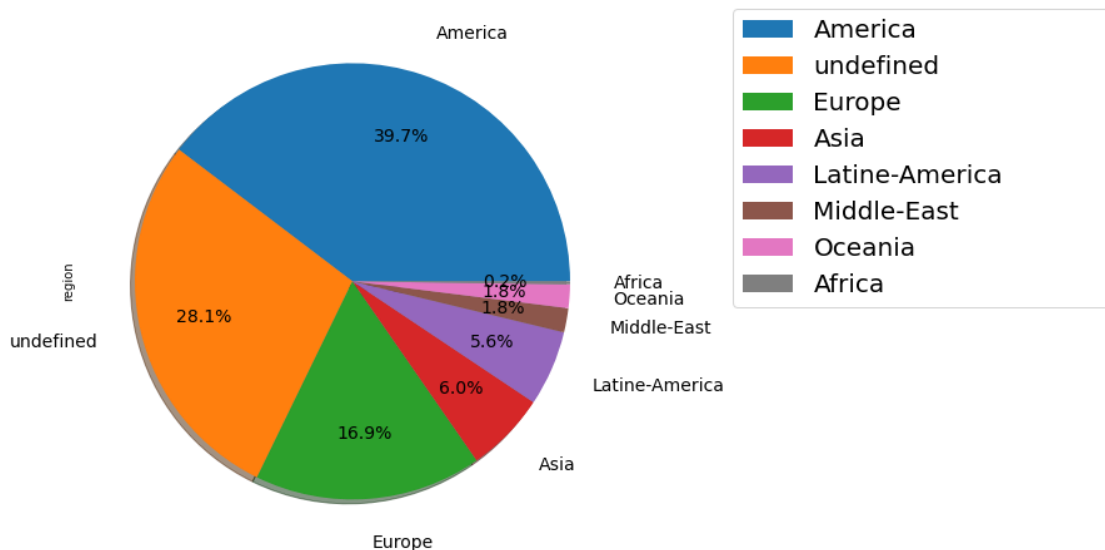
- **coordinates** (99.67% de NaN),
- **geo** (99.67% de NaN),
- **place** (99.62% de NaN).

La variable « id » (probable identifiant du jeu de données) ne nous sert à rien dans le cas présent. Les variables « tweet_lang » et « lang » (langue du tweet et celle de la chanson) ne sont pas pertinentes.

Nous avons regroupé les informations géographiques présentes dans la variable time_zone dans divers continents :

- **America** = US, USA, America, Alaska, Hawaii, Honolulu, Atlantic Time, Arizona, Indiana, Midway Island, Mid-Atlantic, Saskatchewan, Newfoundland;
- **Europe** = EU, Europe, EST, Greenland, Kyiv, Belgrade, Yerevan, Warsaw, Sofia, Riga, Azores, Ljubljana, Volgograd, Sarajevo, Skopje, Bucharest, Vladivostok, Magadan, Brussels, Tallinn, Kamchatka, Novosibirsk, Krasnoyarsk, Bratislava, Zagreb, Vilnius, Yakutsk, Prague, Minsk, Copenhagen, Petersburg, Ekaterinburg, Irkutsk, Vienna, Budapest, Paris, Moscow, Lisbon, Bern, London, Amsterdam, Athens, Stockholm, Helsinki, Dublin, Madrid, Rome, Berlin, Edinburgh;
- **Asia** = Tokyo, Asia, Hanoi, Chennai, Mumbai, New Delhi, Singapore, Kathmandu, Urumqi, JST, Osaka, Beijing, Rangoon, Chongqing, Sapporo, Port Moresby, Solomon Is., Jakarta, Bangkok, Guam, Kolkata, Ulaan Bataar, Kuala, Hong Kong, Taipei, Seoul, Sri Jayawardenepura;
- **Latine-America** = Monterrey, Lima, Bogota, Guadalajara, La Paz, Mazatlan, Chihuahua, Georgetown, Santiago, Brasilia, Quito, Buenos Aires, Caracas, Mexico City, Tijuana;
- **Middle-East** = Tehran, Baghdad, Cairo, Casablanca, Istanbul, Tbilisi, Jerusalem, Kabul, Karachi, Muscat, Abu Dhabi, Kuwait, Dhaka, Baku, Tashkent, Almaty, Astana, Riyadh, Islamabad;
- **Africa** = Pretoria, Africa, Harare, Nairobi, Cape Verde Is., Monrovia;
- **Oceania** = Perth, Melbourne, Brisbane, Sydney, New Caledonia, Adelaide, Wellington, Marshall Is., Nuku'alofa, Canberra, Auckland, Wellington, Hobart, Samoa, Fiji, Darwin.

Près de 30% des informations restent malheureusement « undefined », par conséquent nous avons abandonné l'idée de « recommandation géographique » et n'avons pas conservé cette variable.



2- sentiment values.csv

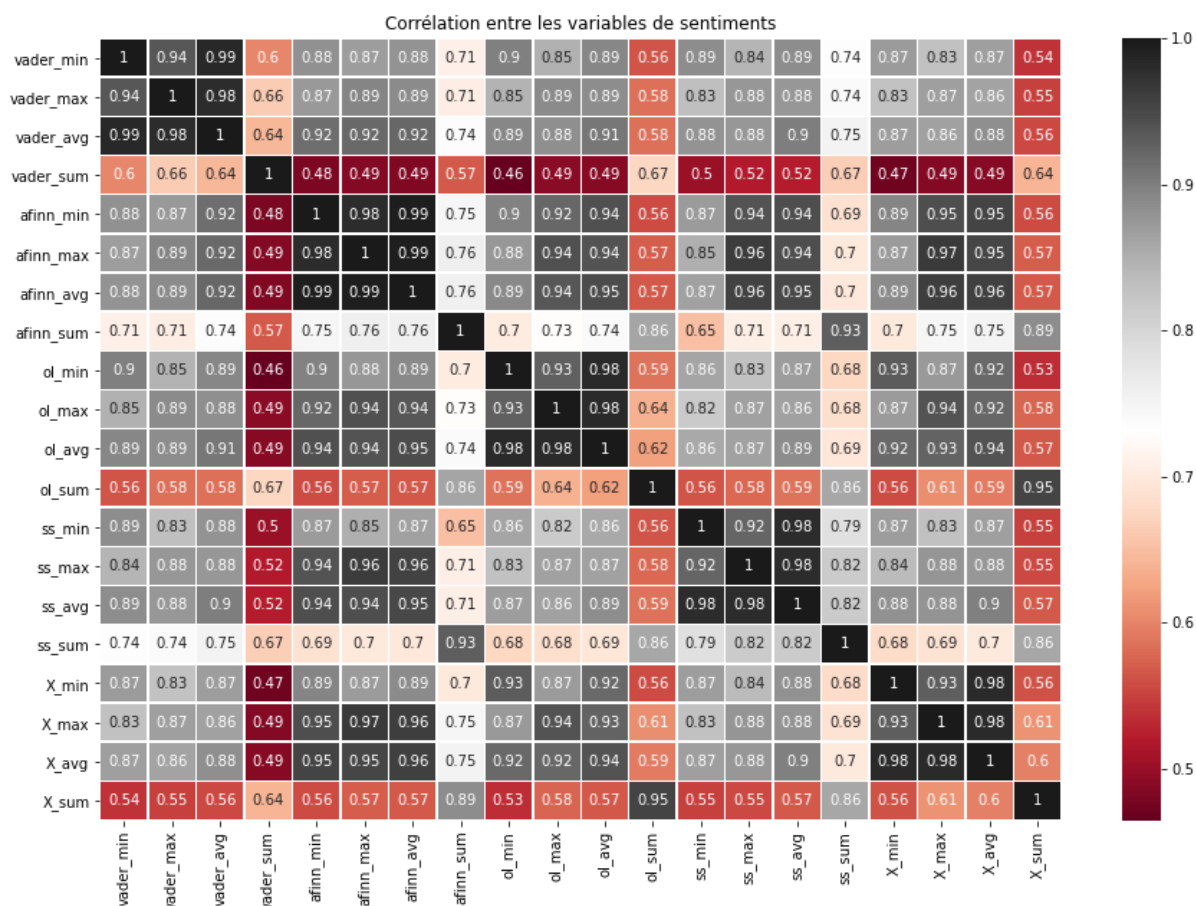
	mean	std	min	25%	50%	75%	max
vader_min	0.495035	0.239754	0.1000	0.3000	0.4000	0.7000	1.0
vader_max	0.513007	0.239695	0.1000	0.3000	0.6000	0.7000	1.0
vader_avg	0.503563	0.236347	0.1000	0.3000	0.6000	0.7000	1.0
afinn_min	0.745098	0.436663	0.0000	0.0000	1.0000	1.0000	1.0
afinn_max	0.752941	0.432149	0.0000	1.0000	1.0000	1.0000	1.0
afinn_avg	0.749020	0.432158	0.0000	0.5000	1.0000	1.0000	1.0
ol_min	0.523107	0.262607	0.0375	0.2875	0.6125	0.7375	0.9
ol_max	0.544905	0.260557	0.0375	0.3125	0.6500	0.7375	0.9
ol_avg	0.534016	0.256876	0.0375	0.3000	0.6125	0.7375	0.9
ss_min	0.451647	0.497745	0.0000	0.0000	0.0000	1.0000	1.0
ss_max	0.491321	0.500013	0.0000	0.0000	0.0000	1.0000	1.0
ss_avg	0.471248	0.489264	0.0000	0.0000	0.0000	1.0000	1.0
X_min	0.518287	0.240837	0.0000	0.3000	0.6000	0.7000	1.0
X_max	0.536991	0.236595	0.0000	0.3000	0.6000	0.8000	1.0
X_avg	0.527677	0.234509	0.0000	0.3000	0.6000	0.7000	1.0

Dans la grande majorité des cas, les valeurs minimums et maximums des notes sentiments sont similaires au sein d'un même dictionnaire (vader, afinn, ol, ...).

Cela signifie que pour un hashtag donné, même si plusieurs entrées ont été utilisées, elles ont toutes conduit aux mêmes résultats.

La matrice de corrélations (ci-dessous) nous indique des corrélations très élevées entre le minimum, le maximum et la moyenne de tous les dictionnaires.

Par la suite, nous nous contenterons donc d'utiliser la valeur moyenne (####_avg).



Légende : corrélations entre les différents attributs des dictionnaires de sentiments.

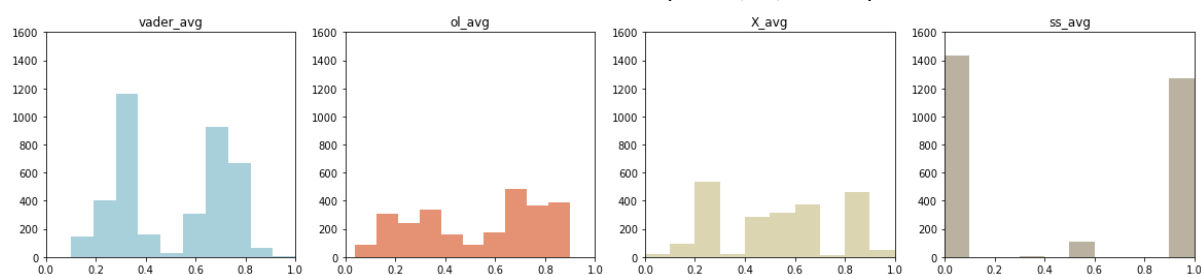
Les seules valeurs plus faiblement corrélées avec les autres sont les « sum », ce qui est normal puisqu'elles dépendent du nombre d'entrées utilisées.

Chaque lexique contient des valeurs manquantes.

	Nbre de valeurs nulles	% de valeurs nulles
vader_avg	1423	27%
afinn_avg	5035	95%
ol_avg	2655	50%
ss_avg	2467	47%
X_avg	3130	59%

Celui avec le moins de valeurs manquantes est Vader (3867 valeurs nulles sur 5290 entrées, soit 27% de valeurs manquantes, ou 73% de données exploitables). Le lexique afinn n'est pas retenu car il contient trop de valeurs nulles (95%).

Distribution des dictionnaires de sentiments conservés (vader, ol, X et ss) :

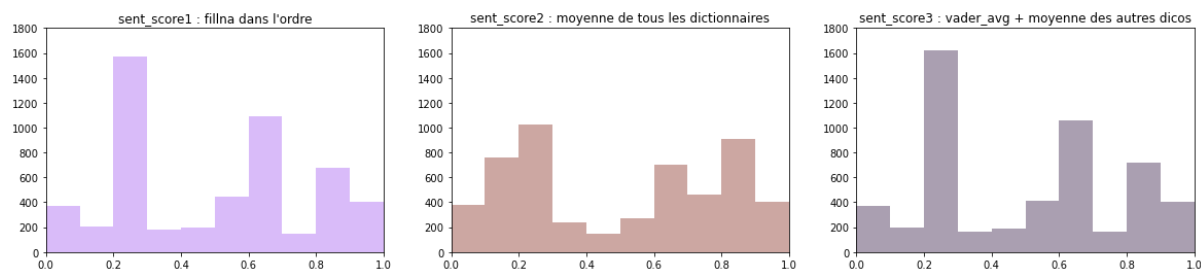


Nous décidons de créer un dictionnaire de sentiment unique basé sur ces quatre dictionnaires.

Plusieurs tests sont effectués, 3 dictionnaires sont ainsi créés :

- sent_score1 : qui complète - pour une même entrée - les valeurs nulles de 'vader_avg' par les valeurs de 'ss_avg', puis les valeurs nulles de 'ss_avg' par les valeurs de 'ol_avg', puis les valeurs nulles de 'ol_avg' par les valeurs de 'X_avg'
- sent_score2 : qui fait une moyenne de toutes les valeurs 'vader_avg', 'ol_avg', 'X_avg', 'ss_avg'
- sent_score3 : qui se base sur 'vader_avg' et complète par la moyenne des autres variables

Distribution des dictionnaires créés :



Le dictionnaire « sent_score2 » nous semble le plus équilibré, c'est celui que nous conservons pour les recherches futures.

3- user_track_hashtag_timestamp.csv

Le fichier ne contient qu'une entrée avec un NaN (hashtag), elle est donc éliminée.

Il comporte beaucoup de redondance, car chaque utilisateur renseigne autant d'entrée pour un morceau que de hashtag renseigné.

Ainsi nous constatons

- qu'un même morceau peut être renseigné par un même utilisateur :
au minimum : 1 fois
au maximum : 50 383 fois
en moyenne : 5.8 fois

et

- qu'un même utilisateur peut renseigner un même morceau à la même date :
au minimum : 1 fois
au maximum : 28 fois
en moyenne : 1.6 fois

Un même morceau renseigné par plusieurs utilisateurs différents indique sa popularité (bonne ou mauvaise, tout dépend des scores de sentiment).

Constitution d'un dataframe de travail

Pour simplifier les tests, nous avons créé 3 jeux correspondants aux datasets « initiaux » (nettoyés), et un dataframe global qui regroupe ces données. C'est ce dernier dataframe global qui est décrit ici.

11 614 671 entrées.

context_content_features
coordinates
instrumentalness
liveness
speechiness
danceability
valence
loudness
tempo
acousticness
energy
mode
key
artist_id
place
geo
tweet_lang
track_id
created_at
lang
time_zone
user_id
id

5290 entrées.

sentiment_values
hashtag
vader_min
vader_max
vader_sum
vader_avg
afinn_min
afinn_max
afinn_sum
afinn_avg
ol_min
ol_max
ol_sum
ol_avg
ss_min
ss_max
ss_sum
ss_avg
X_min
X_max
X_sum
X_avg

17 560 113 entrées.

user_track_hashtag_timestamp
user_id
track_id
hashtag
created_at

Ce dataframe df_global est construit comme suit :

- 1) user_track_hashtag_timestamp « mergé » avec sentiment_values, sur la base du hashtag (le dictionnaire conservé correspond au « sent_score2 » recréé dans sentiment_values)
- 2) « merge » du tout avec context_content_features, sur la base du track_id

df_global
index
user_id
track_id
artist_id
created_at
instrumentalness
liveness
speechiness
danceability
valence
loudness
tempo
acousticness
energy
hashtag
sent_score
hashtag_found *

df_global : 6 356 931 entrées.
df_global_light : 431 906 entrées.

En partant du fichier user_track_timestamp, et en l'enrichissant avec les informations présentes dans les autres fichiers, nous conservons le maximum d'informations.

Cependant, tous les hashtags présents dans user_track_timestamp n'existent pas dans sentiment_values (donc tous les sent_score ne sont pas renseignés, seuls 431 906 le sont).

Une variable « hashtag_found » * a été créée pour faciliter la recherche, elle vaut 1 si le hashtag existe dans sentiment_values (et possède donc une note dans le lexique de sentiments), 0 sinon.

Un autre fichier de travail df_global_light a été créé à partir de celui-ci, avec seulement les hashtag_found=1.

Le fichier df_global comporte 6 356 931 entrées.

10 Le fichier df_global_light en contient 431 906.

Projet

Classification du problème

Notre projet a été abordé sous différentes perspectives.

Tout d'abord, nous l'avons considéré comme une analyse de sentiments sur les morceaux de musique, une problématique d'analyse prédictive type régression linéaire. C'est le modèle 1.

Ensuite, nous l'avons abordé comme un système de recommandation, avec une recherche de similarités entre morceaux via leurs spécifications musicales (clustering), ou via leur mots clés (vectorisation + similarités). Ce sont les modèles 2 et 3.

Chaque modèle est détaillé ci-dessous avec les résultats associés.

Choix des modèles & Optimisation

○ Modèle 1 : Popularité et analyse de sentiments (Régression)

Notre première approche et compréhension du projet fut de définir un modèle de prédiction d'appréciation et de popularité des chansons en fonction de divers paramètres comme les caractéristiques musicales ou encore la localisation géographique des utilisateurs.

Nous comptons l'aborder tel un problème de régression et d'analyse de sentiments, pour relier des variables descriptives (caractéristiques musicales) à un score de sentiment (variable continue) ou une popularité.

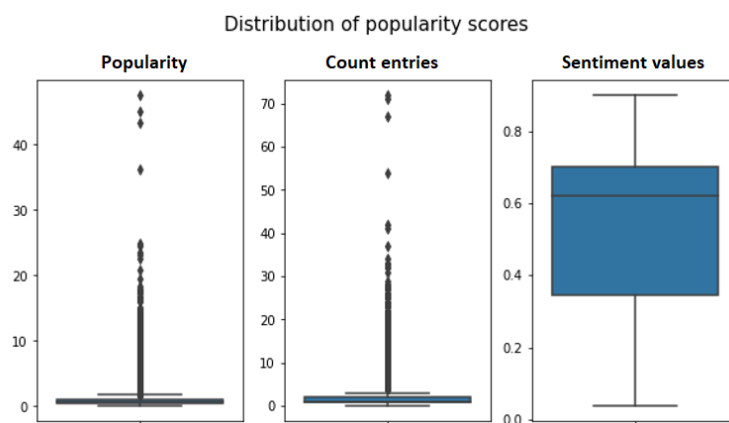
Construction du jeu de données :

Dès notre premier jeu de données nous avons été amenés à trier les hashtags et identifier les scores de sentiments pertinents. C'est ainsi que nous avons éliminé les hashtags correspondant à un score moyen (0.35-0.65), comme par exemple le terme « nowplaying » qui ne porte aucune signification. Certains utilisateurs ayant mis plusieurs hashtags par chanson (ce qui génère plusieurs entrées pour une même chanson), nous avons moyenné les scores pour avoir une seule entrée par utilisateur/chanson.

A partir de là, nous avons pu construire plusieurs variables descriptives :

- « sentiment values » : le score moyen de sentiment pour chaque chanson, en moyennant les scores entre tous les utilisateurs d'un morceau, sachant que cette variable varie entre 0 et 1 (0.5 étant neutre, > 0.5 représente un sentiment positif et < 0.5 est un sentiment négatif).
- « count entries » : le score de popularité en comptant le nombre d'utilisateurs par morceaux (c'est-à-dire est-ce qu'une musique est plus ou moins écoutée). Cette métrique est continue de 1 à un nombre quelconque.
- « popularity » : le score global en multipliant les deux variables précédentes. Ainsi, une musique plutôt partagée (avec une popularité élevée) mais peu appréciée (avec un score de sentiment proche de 0) aura un score global plutôt faible.

Distribution finale des trois différentes variables de popularité :



Le score « Sentiment Values » est plutôt bien distribué (légèrement biaisé vers les hautes valeurs, c'est-à-dire un sentiment positif). Les deux autres scores sont très asymétriques ce qui est plutôt normal. En effet le nombre d'entrées par morceaux est souvent de 1 ou 2, il y a moins de morceaux qui sont partagés de nombreuses fois.

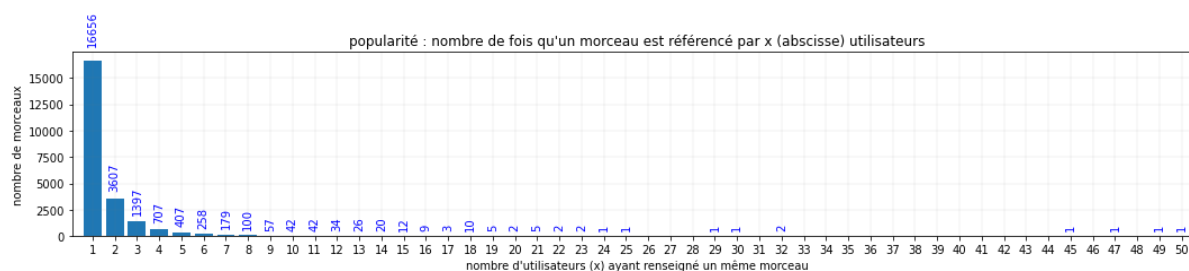
Relation entre popularité et score de sentiment

La popularité d'un morceau est le nombre de fois qu'un même morceau est partagé par différents utilisateurs. Nous souhaitons voir s'il existait une relation entre la popularité et le score de sentiment d'un morceau.

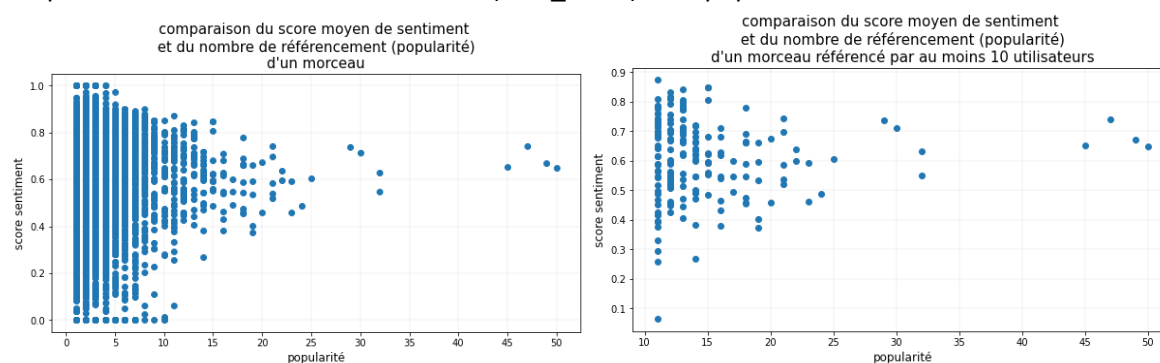
Pour ce faire, nous avons dédoublonné les entrées sur le couple (user_id, track_id) car il arrive qu'un même utilisateur renseigne plusieurs fois le même morceau. Ceci nous permet d'identifier les morceaux référencés par des utilisateurs différents.

Pour les plus populaires : 1 morceau est référencé par 50 utilisateurs, un autre par 49, ..., 2 par 32 utilisateurs, 10 par 19 utilisateurs, etc...

Pour les moins populaires : 16656 morceaux différents renseignés par un seul utilisateur, 3607 renseignés par 2 utilisateurs, ... 100 renseignés par 8 utilisateurs, etc

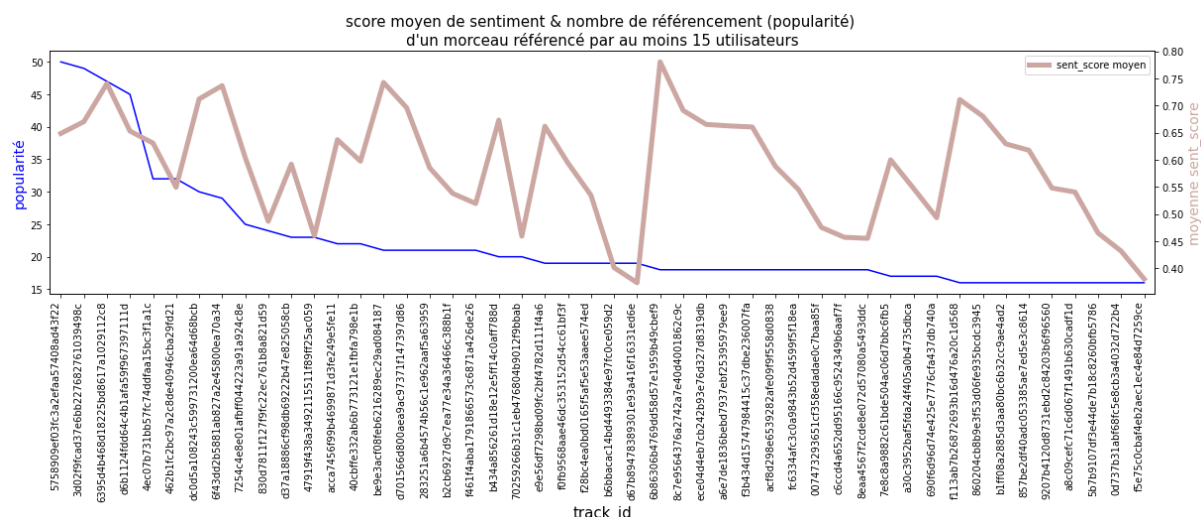


Comparaison entre le score de sentiment (sent_score) et la popularité d'un morceau :



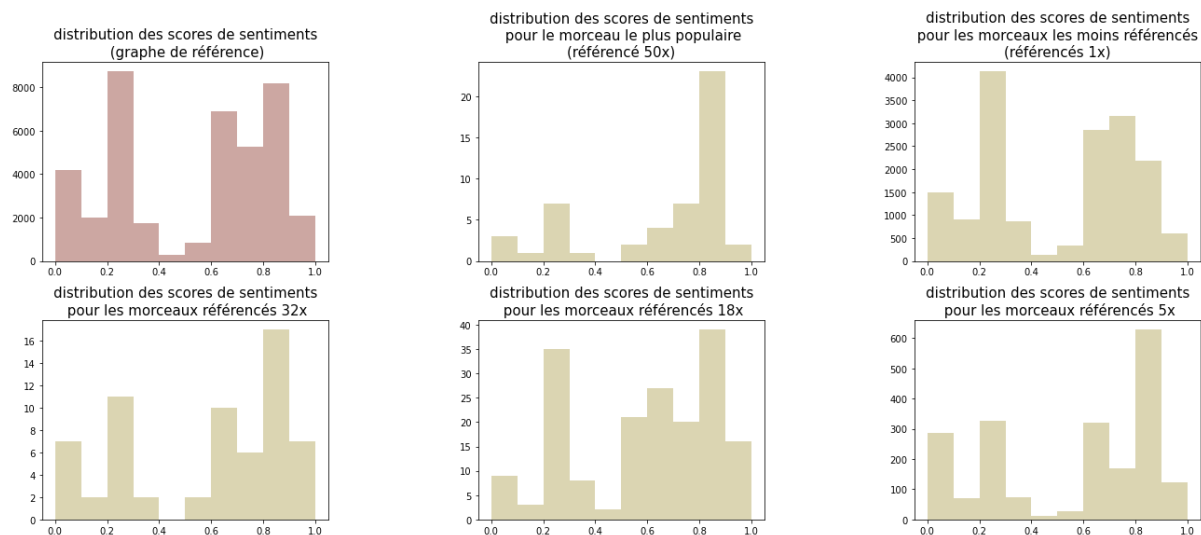
Ce n'est guère parlant pour les morceaux peu populaires, le score de sentiment varie grandement.

Illustration pour des morceaux (abscisse) référencés par au moins 15 utilisateurs :
En bleu la courbe de popularité, en marron la moyenne des scores de sentiment.



Idem, il ne semble pas y avoir de relation entre le score moyen de sentiment d'un morceau et sa popularité.

Distribution des scores de sentiment en fonction de diverses popularités :



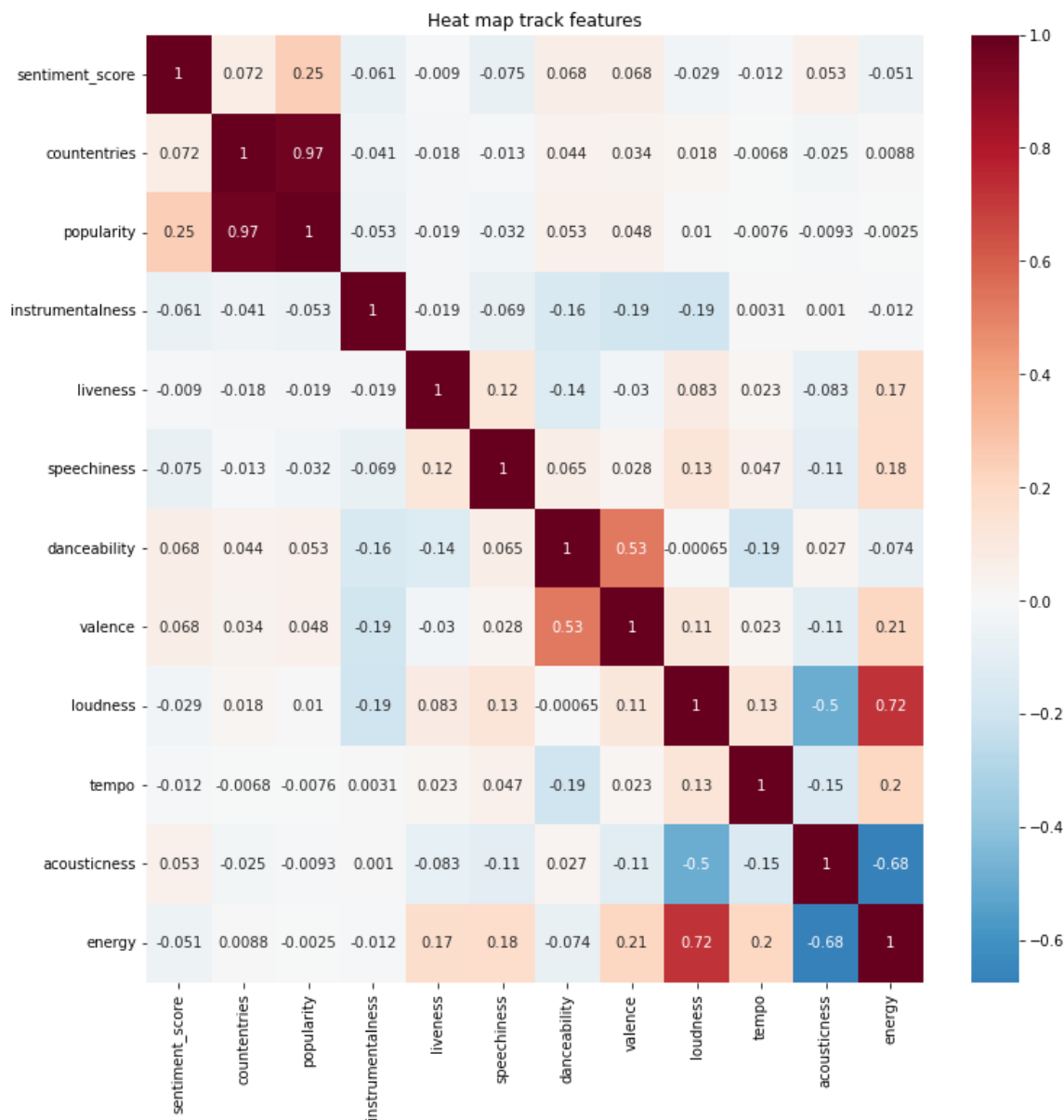
Ces distributions ne permettent pas non plus d'identifier un lien entre score de sentiment et popularité d'un morceau.

En conclusion :

Il semblerait que la popularité d'un morceau n'ait pas d'incidence sur le score de sentiment, et inversement. Les deux variables sont peut-être complémentaires.

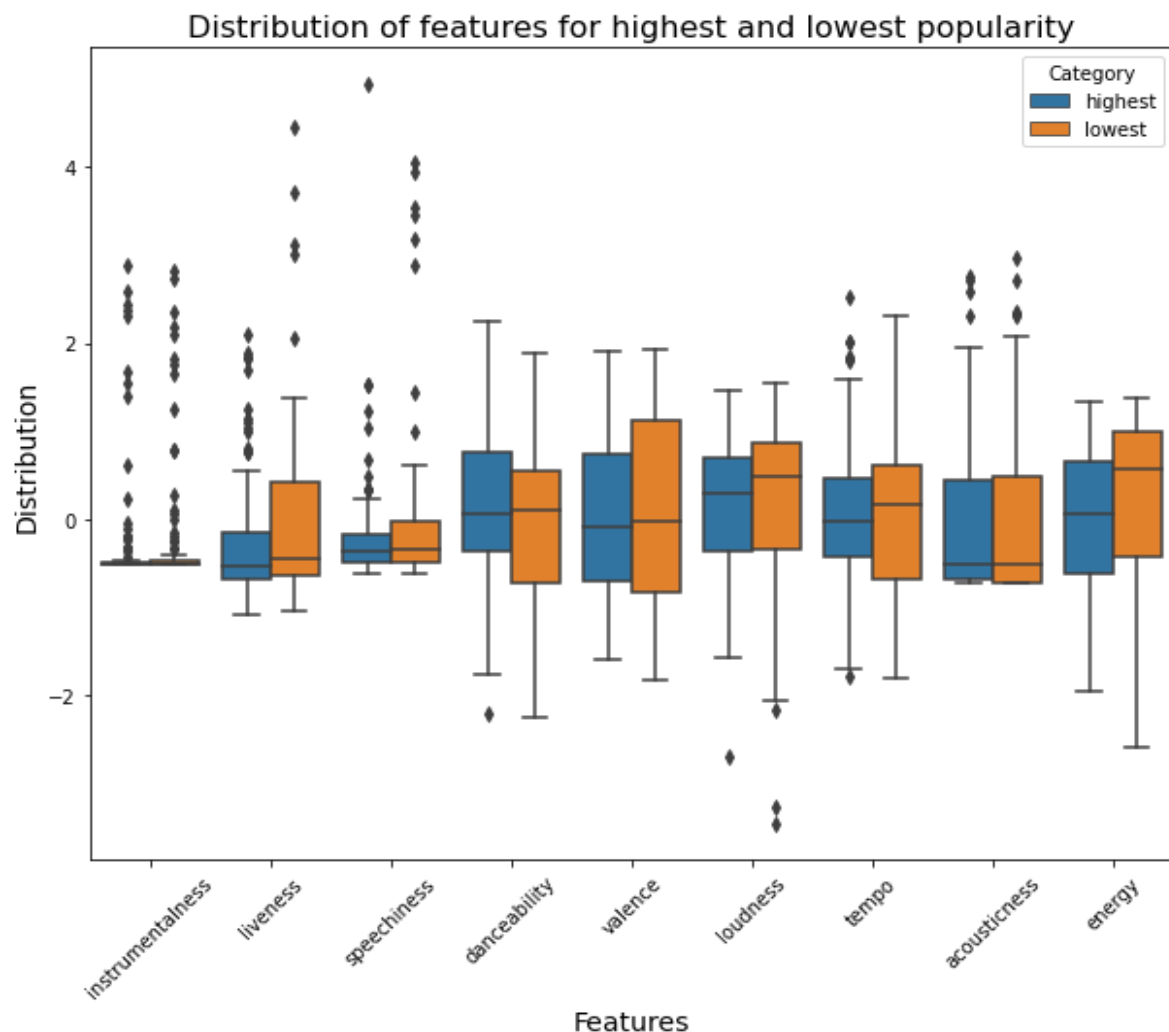
Relation entre popularité et variables prédictives :

Nos trois cibles (sentiment_score, count_entries, popularity) étant des variables continues, nous voulions nous orienter vers un modèle de régression. Nous avons regardé leur lien avec les variables prédictives (de type continu uniquement). Les corrélations entre chaque score et les variables prédictives sont représentées sur la figure qui suit :



Hélas, il apparaît que les caractéristiques musicales n'ont pas de lien évident avec les scores de popularité (corrélations très faibles).

Pour nous en assurer, nous avons regardé les distributions des variables prédictives pour les chansons les plus et les moins appréciées (en prenant les 100 entrées avec les plus hauts et les plus bas scores respectivement). Les résultats sont présentés sur la figure suivante pour la variable de popularité combinée (nombre d'entrées x score de sentiment) :



Cela confirme qu'il n'existe pas de différence apparente dans les caractéristiques musicales, entre les morceaux les plus populaires et les morceaux les moins populaires.

Au vu de ces premiers résultats peu convaincants, nous avons décidé de nous orienter vers une approche totalement différente : bâtir un système de recommandation.

- Modèle 2 : Recommandation par clustering sur les caractéristiques musicales

Dans un second temps, nous avons abordé le sujet comme un problème de recommandation musicale. Il fallait alors tenter de proposer des morceaux en fonction des goûts ou des écoutes d'un utilisateur. Pour cela, nous devons identifier des groupes correspondants au morceau écouté, en regroupant les musiques par similarités, ce qui revient à résoudre un problème de clustering.

Nous avons choisi un modèle K-Mean, relativement facile à implémenter.

Nous nous sommes basés sur les caractéristiques musicales des morceaux (issues originellement de « context_content_features.csv ») pour tenter d'identifier des groupes. L'avantage de se baser uniquement sur les caractéristiques musicales est que le modèle pourrait ensuite être appliqué à n'importe quelle chanson sans autre renseignement que ses caractéristiques (liveness, danceability...).

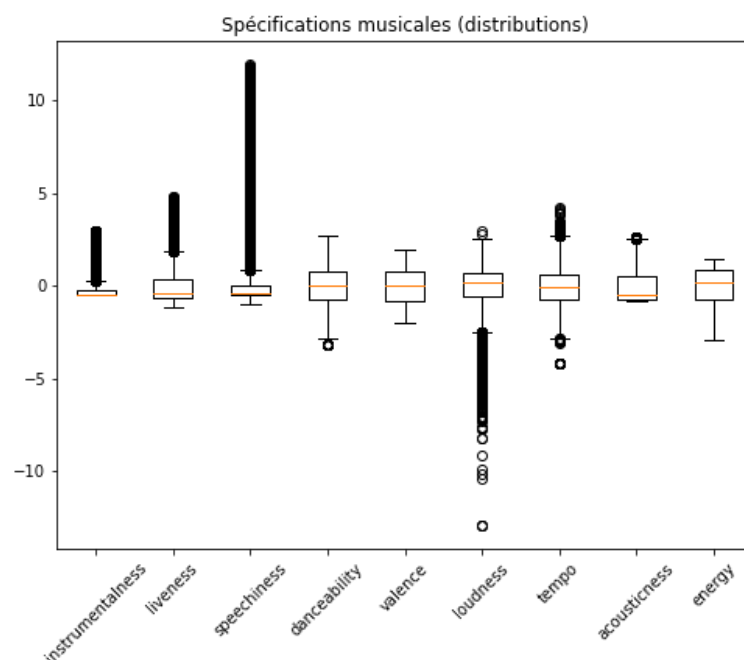
L'un des problèmes avec les méthodes de clustering est qu'il est difficile d'évaluer la pertinence des résultats (sauf dans le cas où on connaît les groupes cibles à identifier). Nous avons toutefois trouvé un moyen d'évaluer la cohérence de notre modèle par des méthodes indirectes, en observant les hashtags les plus prédominants dans chaque cluster (cluster de « style musical »).

Construction du jeu de données :

Seules les caractéristiques numériques des morceaux (liveness, danceability...) ont été conservées. Nous avons ignoré les autres variables (liées aux utilisateurs ou aux sentiments) et retiré ces variables catégorielles : « mode » (qui prend une valeur de 0 ou 1 pour mineur ou majeur) et « key » (qui a un nombre fini de valeurs entières et représente la clé / tonalité du morceau).

Ainsi chaque entrée de notre modèle est une piste de musique avec 9 variables descriptives (qui sont normalisées pour avoir des échelles similaires).

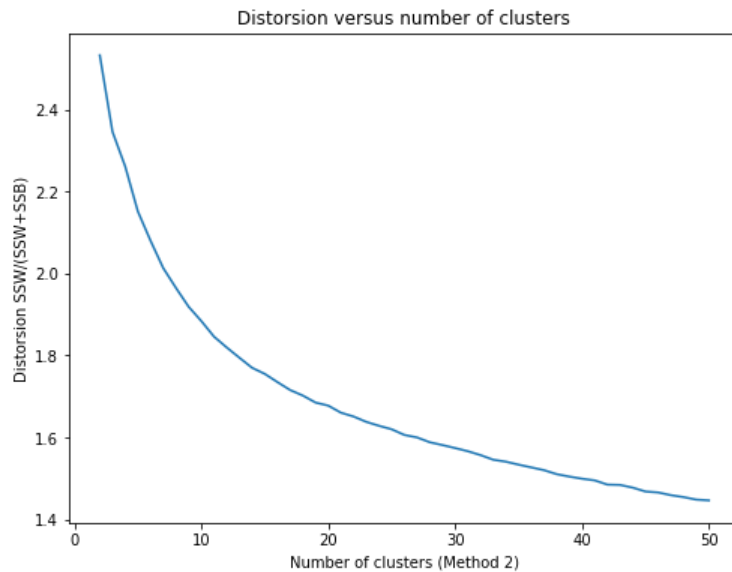
L'identifiant des chansons (track_id) est conservé de côté pour voir ensuite être utilisé plus tard.



Modèle K-Means

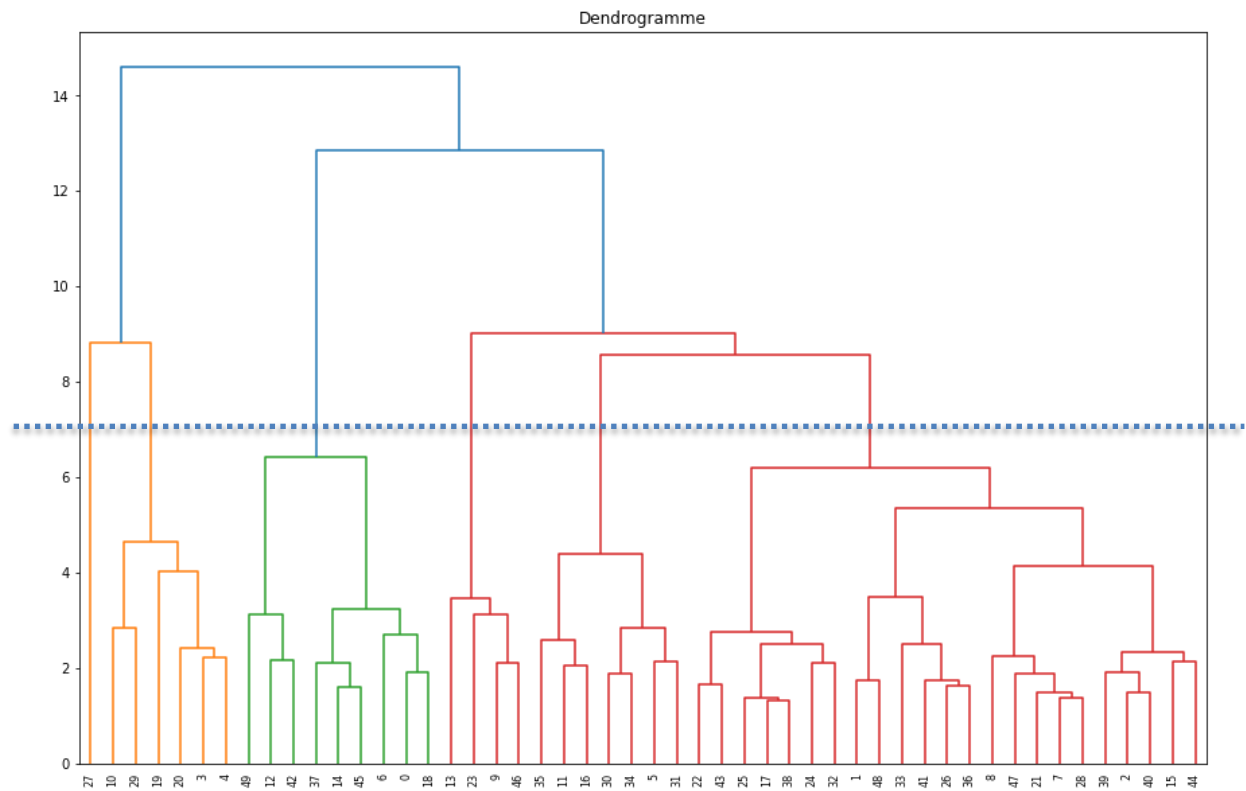
L'analyse du coude permet parfois de déterminer le nombre optimal de classes à définir, en cherchant une rupture dans la courbe. Cette rupture indique que la valeur ajoutée des classes supplémentaires devient plus faible (au départ, en augmentant le nombre de classe on réduit fortement la distorsion, mais au bout d'un moment ajouter de nouvelles classes ne permet plus de réduire efficacement ce score).

Cette rupture n'est pas nette ici, pourtant nous notons un changement de la pente, raide au début, douce ensuite à partir de 6 à 10 classes. Il est difficile de déterminer directement un nombre optimal de classes sur notre jeu de données avec cette méthode.



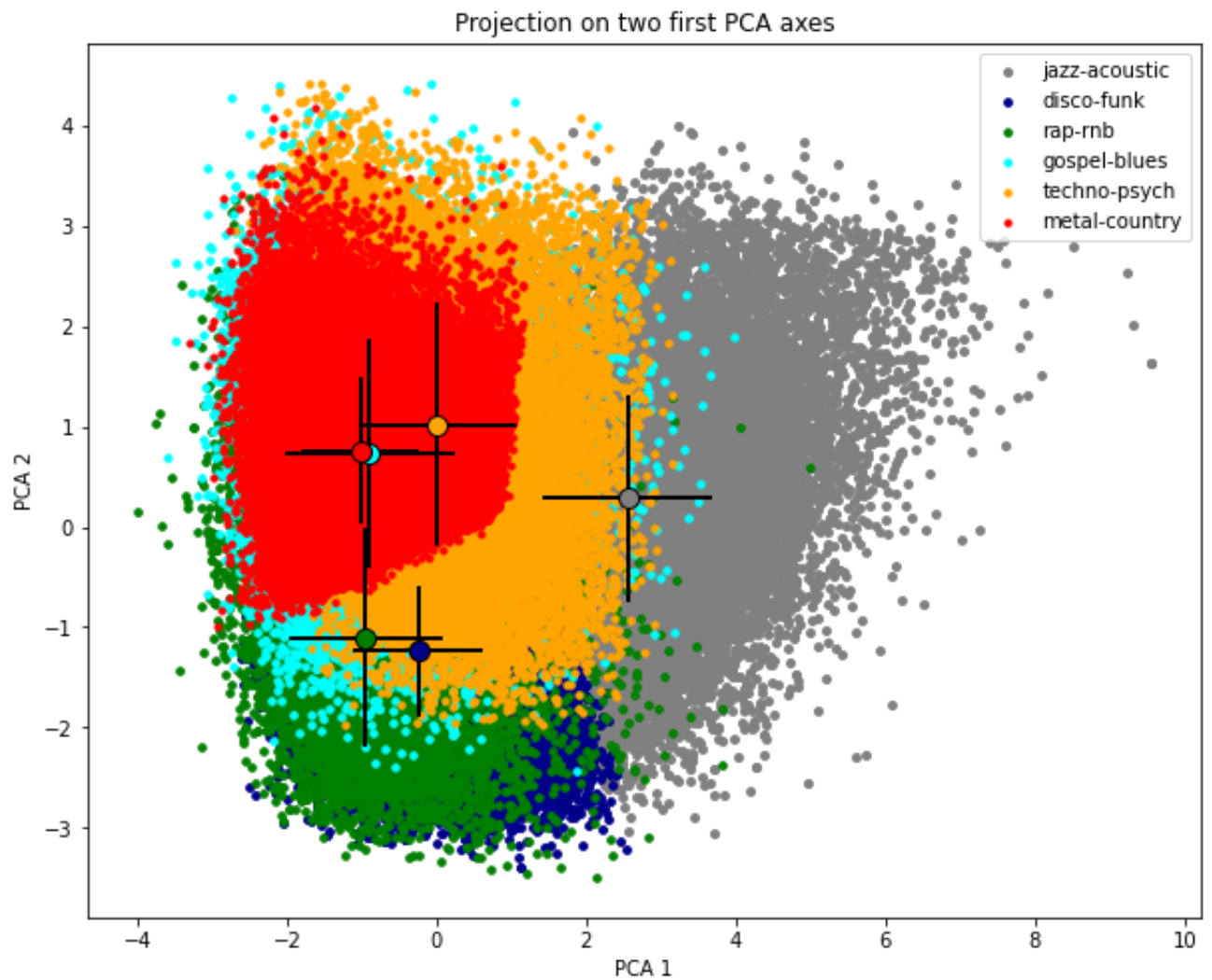
En complément de cette analyse, un dendrogramme montre que 6 classes semblent être un nombre correct (niveau 7 du dendrogramme). Les groupes semblent suffisamment séparés les uns des autres (figures ci-dessous).

Nous avons testé plusieurs combinaisons, de 6 à 10 groupes et avons conservé 6 groupes, ce qui nous a semblé conduire à des résultats valables.



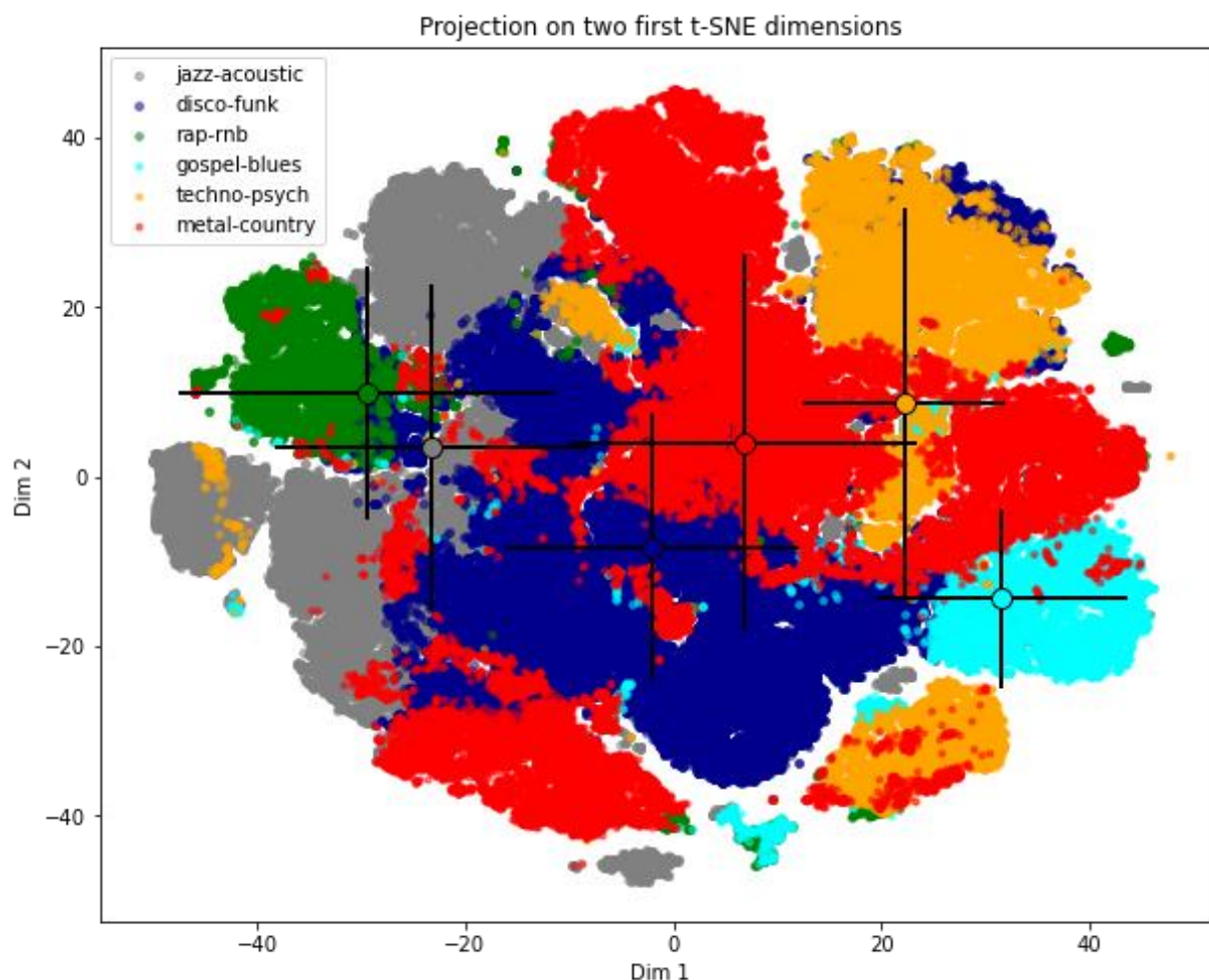
Visualisation des clusters

Après avoir entraîné le modèle avec 6 clusters sur le jeu de données complet, une analyse en composante principale permet de visualiser où se trouvent les clusters selon les 2 premiers axes principaux (le nom des groupes est basé sur une analyse Word-Cloud effectuée par la suite) :



Certains clusters sont assez bien isolés des autres (groupe 0 ici) mais d'autres ont tendance à se superposer. Cela ne veut pas forcément dire que le clustering est mauvais, mais peut-être que 2 dimensions ne sont pas suffisantes pour démarquer ces clusters.

Une autre façon de visualiser les clusters dans un espace réduit est d'appliquer un modèle t-SNE, qui permet de regrouper les morceaux par affinité dans l'espace réduit. Les résultats sur 2 dimensions sont présentés sur la figure ci-dessous.



Certains clusters se démarquent en partie (par exemple gospel-blues) mais dans l'ensemble on ne peut pas dire que leur répartition est clairement définie.

Encore une fois, cela peut vouloir dire que 2 dimensions ne sont pas suffisantes pour bien représenter leur séparation, ou alors que les musiques ne peuvent pas être clairement regroupées en groupes ce qui semble aussi plausible.

En effet, même au sein d'un même genre musical, les caractéristiques musicales peuvent grandement varier.

Evaluation du modèle

Pour évaluer la cohérence des résultats nous identifions les hashtags des morceaux de chaque groupe de musique créé (cluster).

Cependant, de nombreux hashtags correspondent plus à l'appréciation de l'utilisateur qu'à un style de musique.

Nous avons donc d'abord sélectionné uniquement les hashtags qui correspondent à des styles musicaux généraux, d'après une liste prédéfinie de 28 genres :

Acoustic, Alternative, Blues, Classic, Country, Dance, Disco, Electronic, Funk, Gospel, Goth, Hip-hop, House, Indie, Jazz, Latin, Lounge, Metal, Pop, Psych, Punk, Rap, Reggae, RNB, Rock, Soul, Techno, Triphop.

Cette liste fut créée en cherchant les genres inclus dans les hashtags de chaque morceau (par exemple « rock » est inclus dans « radiorock » ou « vivelerock »).

Cette sélection réduit fortement le nombre de pistes disponibles, puisque beaucoup ont des hashtags qui ne correspondent pas forcément à des genres.

Ce jeu de données filtré est uniquement utilisé pour la validation du clustering (le modèle étant entraîné sur l'ensemble des pistes du jeu de données de base).

Pour analyser le champ lexical associé à chaque cluster du modèle, nous appliquons le modèle au jeu de données filtré (dont les hashtags correspondent à des genres spécifiques), puis nous effectuons une analyse du champ lexical de chaque groupe (c'est-à-dire que l'on compte les occurrences des différents hashtags associés aux pistes de chaque groupe).

Chaque groupe est alors nommé en fonction de termes représentatifs de celui-ci : 0 : jazz-acoustic, 1 : disco-funk , 2 : rap-rnb, 3 : gospel-blues, 4 : techno-psych, 5 : metal-country.

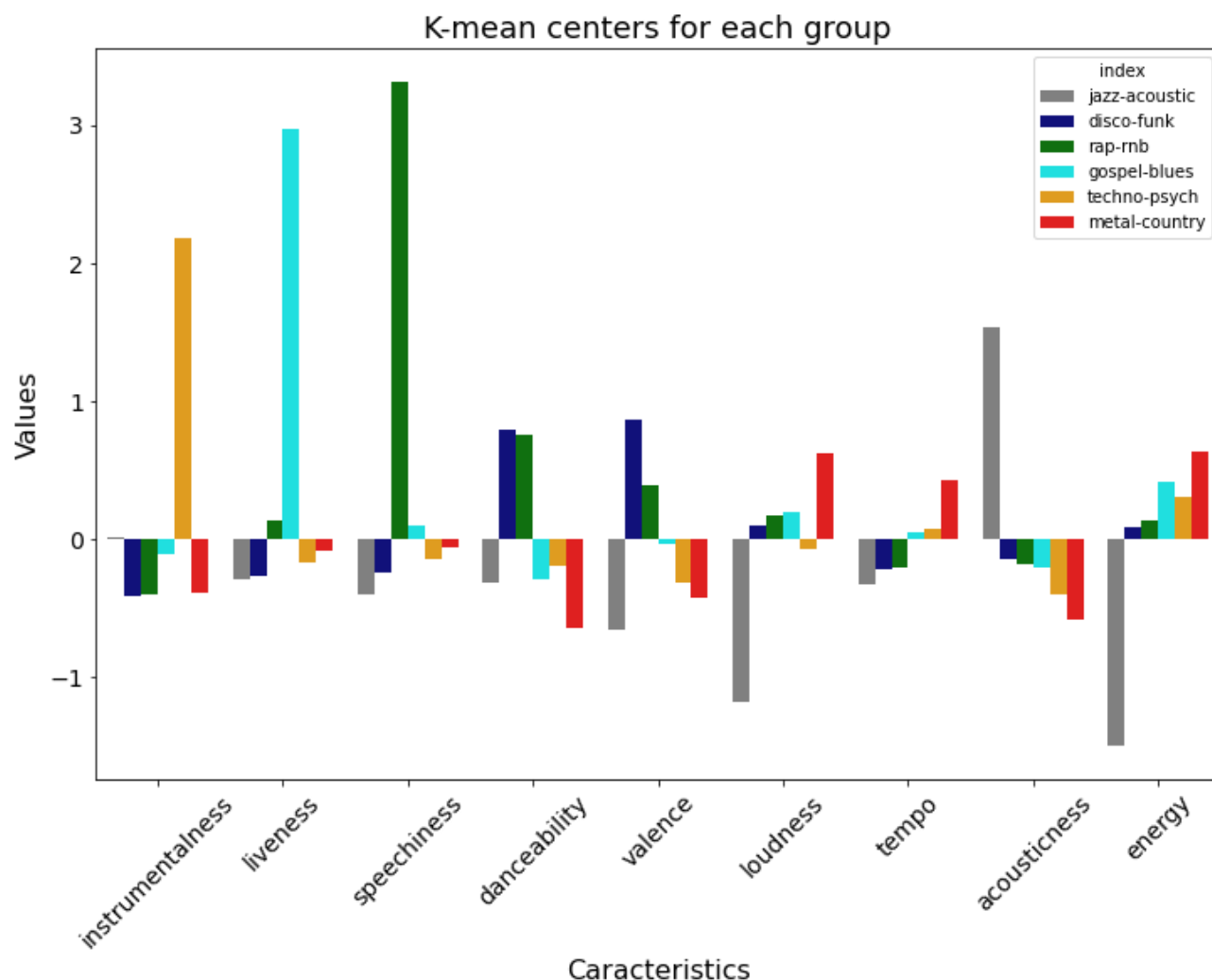
L'affichage du résultat sous forme de « Wordcloud » vaut mieux qu'un long discours :

Wordclouds for 6 groups of music



Certains groupes ont des caractéristiques assez différentes des autres, mais dans chaque groupe il y a quand même un ou deux styles qui semblent dominants, ce qui peut être suffisant pour un système de recommandation par affinité musicale. Par exemple, le groupe 2 est orienté rap/rnb, alors que le groupe 1 est plutôt disco/funk, ce qui semble cohérent.

Le graphique de répartition ci-dessous montre comment le clustering identifie les groupes selon les caractéristiques musicales (variables d'entrées) :



Certains groupes se démarquent clairement avec une variable précise (par exemple le groupe « 3 : gospel-blues » est fortement basé sur la variable « liveness ») alors que d'autres sont identifiés par une combinaison de variables (le groupe « 0 : jazz-acoustic » se démarque sur trois variables, « loudness », « acousticness » et « energy »).

La méthode de clustering ici permet donc de différencier certaines tendances.

L'une des limites de cette méthode est qu'elle considère que toutes les musiques d'un genre doivent avoir des caractéristiques musicales proches, et cela pourrait s'avérer un peu limitant dans le cas de groupes « borderline » qui expérimentent avec des sonorités très différentes.

Nous constatons cependant que certains résultats sont mal classés, comme par exemple :

	track_id	hashtag	labels
120	67eeca3f067d0496881f4d283f0f626c	acoustic	5
121	0f6c3e42bc32d33710b9e066a14d6374	acoustic	1
123	54d0a6d0c80ffc3856c2729f63363a7b	acoustic	5
125	7d5f7749b21900870377d92d2c8ae0d9	acoustic	5
126	6a5aad35ef90a548ccf1fe7e6daf774a	acoustic	5

Ici plusieurs morceaux de style « acoustic » sont associés au groupe 5 : metal-country. Cela montre que parfois les clusters sont poreux entre différents styles, ou alors qu'au sein de certains styles on peut avoir des caractéristiques musicales très différentes.

Malheureusement ceci a des effets sensibles sur les résultats de recherche de notre système de recommandations. En effet, celui-ci se base sur les morceaux du même groupe (cluster).

Exemples :

Morceau de référence de hashtag « pop », catégorisé dans le cluster « 0 : jazz-acoustic ».

La recherche dans le même cluster renvoie des morceaux de hashtag « acoustic » ou « indie ».

morceau de référence : 844cd2570ddf0729bde295219cb25514

hashtag : ['pop']

label : [0 : jazz-acoustic]

	track_id	hashtag	labels
9	ee67b313db5a7fb1fdff43cc7f305e72	acoustic	0
10	bed12e9de9a2904aa7ee92088bed4096	acoustic	0
17	47919f438a3492115511f89ff25ac059	acoustic	0
19	47919f438a3492115511f89ff25ac059	indie	0
21	dd0657bc29856e0af8f4bbcfad8e0f7c	acoustic	0

Morceau de référence de hashtag « metal », catégorisé dans le cluster « 4 : techno-psych ».

La recherche dans le même cluster renvoie des morceaux de hashtag « acoustic » ou « alternative ».

morceau de référence : d967db99cb358341ec06287b14c485c6

hashtag : ['metal']

label : [4 : techno-psych]

	track_id	hashtag	labels
49	3bb38331d88e6b6dfe627e15db190075	acoustic	4
57	d5fc0e7dd33205244c97bbf13a7fa822	acoustic	4
95	a5452c50af87568b010061c1f0e49d3a	acoustic	4
130	47ba1f3acab95e3cf8d30843abb5032c	alternative	4
138	e1ddf8e992f653333094683896072f95	alternative	4

Test sur un autre jeu de données :

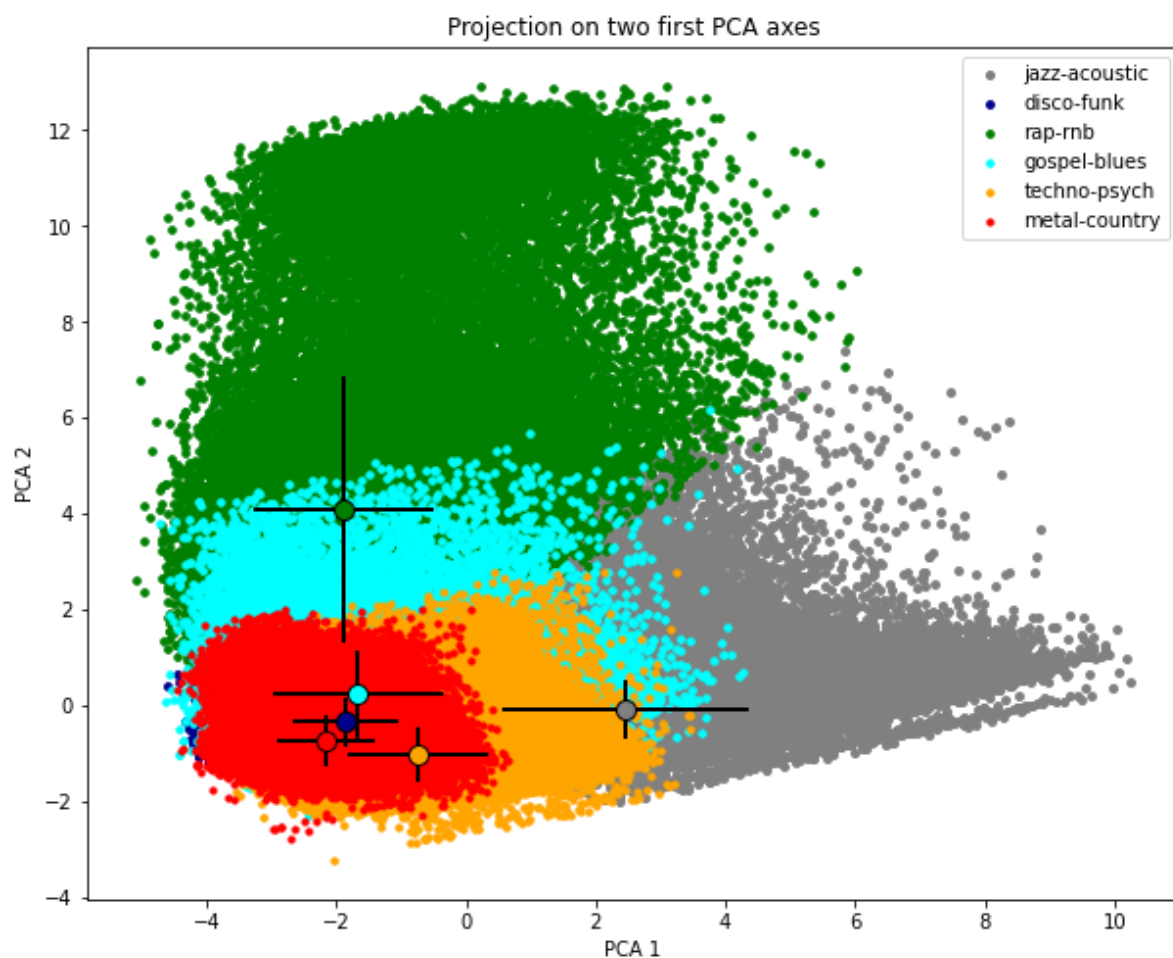
Pour vérifier les résultats différemment, nous avons récupéré un jeu de données de pistes musicales différent, il s'agit de Spotify 1.2M+ Songs :

<https://www.kaggle.com/datasets/rodolfofigueroa/spotify-12m-songs>

Ce jeu de données contient plus d'un million de chansons. Il contient les mêmes caractéristiques musicales et est donc compatible avec notre modèle, mais en plus il contient le nom des artistes et des groupes de chaque morceau (mais pas de hashtags).

En appliquant notre modèle à ce jeu de données, nous pouvons par exemple vérifier que les morceaux d'un cluster ont tendance à être issus de groupes proches musicalement.

L'analyse en PCA montre une nouvelle fois que certains groupes se démarquent déjà même avec seulement 2 dimensions principales :



Nous avons regardé comment le classement catégorise certains groupes de musique par exemple, mais il reste difficile de tout évaluer.

Par exemple, en regardant toutes les entrées de l'artiste Will Young, nous obtenons :

	name	album	artists	labels
36	Love The One You're With	Fridays Child	['Will Young']	1
37	Your Game	Fridays Child	['Will Young']	1
38	Stronger	Fridays Child	['Will Young']	1
39	Leave Right Now	Fridays Child	['Will Young']	1
40	Love Is A Matter Of Distance	Fridays Child	['Will Young']	0
41	Dance The Night Away	Fridays Child	['Will Young']	1
42	Very Kind	Fridays Child	['Will Young']	1
43	Free	Fridays Child	['Will Young']	1
44	Going My Way	Fridays Child	['Will Young']	0
45	Out Of My Mind	Fridays Child	['Will Young']	1
46	Friday's Child	Fridays Child	['Will Young']	1

Cet artiste est donc essentiellement classé dans le cluster 1 (Disco-Funk) mais certains morceaux sont dans le cluster 0 (Jazz-Acoustic). En réalité, cet artiste est plutôt du style « Pop », mais aucun de nos clusters a été identifié comme ayant ce style majoritaire.

Cela montre les limites de l'évaluation de notre analyse.

Peut-être que davantage de clusters permettraient de différencier davantage les styles. Il se peut toutefois que certains styles soit absent des hashtags (ou moins présents) et donc l'analyse en World Cloud ne les fait pas ressortir, ce qui rend l'évaluation délicate.

En conclusion :

Cette méthode de clustering semble est intéressante car elle peut être appliquée facilement à n'importe quelle base de données de musiques, mais elle reste difficilement interprétable. Il faudrait sans doute pouvoir la tester directement dans un vrai système de recommandation pour voir la pertinence des propositions.

○ Modèle 3 : Recommandation par similarités sur le champ lexical

Dans cette seconde approche, nous nous sommes basés sur le champ lexical associé à chaque morceau de musique, c'est-à-dire leurs hashtags. L'avantage de cette approche est qu'elle est potentiellement plus riche, puisque le champ lexical peut être très précis et s'enrichir avec de nouveaux termes en permanence. En revanche, elle implique que chaque nouveau morceau de musique que l'on voudrait mettre dans le modèle soit déjà renseigné par des hashtags. Cela peut être problématique si aucun utilisateur n'a renseigné ce genre d'information par exemple, et conduirait à l'exclusion systématique de ces morceaux par le système de recommandation.

Un même morceau (track_id) peut contenir plusieurs hashtags, chaque hashtag génère une entrée. Nous avons tout d'abord regroupé tous les hashtags d'un même morceau dans une seule entrée (en supprimant les autres entrées).

track_id	hashtag		track_id	hashtag
0000e47c1207e2c637a44753a713456f	rock	}	0000e47c1207e2c637a44753a713456f	rock pops 70s music
0000e47c1207e2c637a44753a713456f	pops			
0000e47c1207e2c637a44753a713456f	70s			
0000e47c1207e2c637a44753a713456f	music			
0000e47c1207e2c637a44753a713456f	music			

Nous avons ensuite passé le champ hashtag ainsi constitué via un filtre stopwords en le personnalisant afin de supprimer aussi les hashtags sans intérêt : 'listenlive', 'class95', 'kiss92', 'music', '6music', '234radio', 'jammin105', 'radio2', 'audio5cafe', 'radio', 'bbc6music', 'tweetlink', 'myplaylist', 'tbfmonline', 'vkscrobber', 'wzbt', 'nowlistening', 'spotify', 'onlineradio', 'tunein', ...

nettoyage des mots clés : hashtag -> hashtag_cleaned

track_id	hashtag	hashtag_cleaned
0000e47c1207e2c637a44753a713456f	rock pops 70s music	rock pops 70s

C'est ce champ hashtag_cleaned qui sera utilisé pour notre recherche de similarités.

Après nettoyage des NaN, le dataset de travail contient 2 variables et 111 201 entrées :

	track_id	hashtag_cleaned
0	00003213fb3d4959f42e9157b0eda0a5	newchristianmusic
1	0000e47c1207e2c637a44753a713456f	rock pops 70s
2	0001dc79946a42fbc837c044be0bdbbc	apt
3	000248c97c5991b9900360aca97d9879	musicislife disturbed
4	00027df4d0bd64108624757fe4cbfe76	rock hardrock
...
111196	fffd0d291f4addb6c4640a0f50d22f3f	hitmusic chicagomusic hitparty
111197	fffd293d9450783348d5f9d169ed8abe	hopes
111198	fffd997ff184ba9929358a77644fff6c	musicislife
111199	fffd00857154771de0d8479d8341e1e	stonerrock doometal
111200	ffef37bec01701ce5f548fddb26ad8b	musicislife

111201 rows × 2 columns

L'algorithme employé pour rechercher les similarités entre les textes composés de hashtags est la vectorisation « term frequency, inverse document frequency » TF-IDF.

Cet algorithme a été développé à l'origine pour améliorer les moteurs de recherches.

TF : « Term Frequency » représente la densité du mot, c'est-à-dire le nombre de fois où le mot est présent dans le document analysé.

IDF : « Inverse Document Frequency » mesure le nombre de documents dans le corpus étudié qui contiennent un terme donné, rapporté à l'ensemble des documents analysés.

Ainsi l'algorithme TF-IDF permet d'obtenir le poids (évaluation de la pertinence d'un mot) selon deux facteurs : la fréquence de ce mot dans le document (TF) et le nombre de documents contenant ce mot (IDF) dans le corpus étudié.

TF-IDF

TF-IDF is a measure of originality of a word by comparing the number of times a word appears in a doc with the number of docs the word appears in.

$$TF-IDF = TF(t, d) \times IDF(t)$$

Term frequency (TF) is the number of times term t appears in a doc, d .

Inverse document frequency (IDF) is the inverse of the number of documents containing the term t .

$$IDF(t) = \log \frac{1 + n}{1 + df(d, t)}$$

where n is the number of documents, and $df(d, t)$ is the document frequency of the term t .

Vectorisation TF-IDF :

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer().fit_transform(df_vector['hashtag_cleaned'])
```

Nous utilisons ensuite cette matrice de vecteurs pour rechercher des similarités entre les éléments (comme nous le ferions avec une matrice de vecteurs CountVectorizer).

Le volume des données ne nous permet pas d'utiliser la fonction `cosine_similarity`, nous avons utilisé `linear_kernel` qui est similaire à `cosine_similarity`, mais plus adapté aux larges volumes de données.

Comparaison entre le 1^{er} terme (index = 1) `tfidf[1:2]` et les autres termes de la matrice TF-IDF :

```
from sklearn.metrics.pairwise import linear_kernel
hashtag_similarities = linear_kernel(tfidf[1:2], tfidf).flatten()
```

`hashtag_similarities` contient alors une matrice de poids indiquant la similarité entre les termes.

De cette matrice, nous extrayons les index des 5 plus grands résultats :

```
related_docs_indices = hashtag_similarities.argsort()[::-5:-1]
related_docs_indices

>> array([    1, 83847, 30054, 44206], dtype=int64)
```

Ceci nous permet de connaître les index des termes les plus similaires avec le terme recherché (index 1). Forcément l'index 1 est le meilleur résultat, suivi d'autres index 83847, 30054, 44206.

Comparaison des hashtags des index trouvés par le modèle

```
index = 1, track_id = 0000e47c1207e2c637a44753a713456f
hashtag_cleaned = rock pops 70s
```

```
index = 83847, track_id = c0a3db38bd9f8f8c44ed0c2d0e1b60af
hashtag_cleaned = rock pops 70s
```

```
index = 30054, track_id = 45138db5d3e84a423b12650a0f65d4a6
hashtag_cleaned = rock 70s pops
```

```
index = 44206, track_id = 65929a3541868bf88b85bdc16e86e1d0
hashtag_cleaned = rock pops 60s
```

Les résultats sont satisfaisants en termes de similarité de hashtags.

Malheureusement le jeu de données ne permet pas de comparer les noms des morceaux, nous ne disposons que des `track_id`. Cela dit la pertinence des résultats avec les noms des morceaux dépendrait fortement de la pertinence des mots clés renseignés par les utilisateurs, ce ne serait peut-être pas plus parlant.

Nous décidons de généraliser cette recherche via une fonction

```
...
    recherche nb morceaux similaires à track_id
    via la colonne qui regroupe les hashtags
...
def similar_tune(track_id, nb):
    # recherche de l'index du track_id dans le dataframe
    df_index = get_index_by_trackid(track_id)

    # matrice de similarité
    hashtag_similarities = linear_kernel(tfidf[df_index:df_index+1], tfidf).flatten()

    # résultats similaires (index)
    similar_tunes_idx = hashtag_similarities.argsort()[::-nb-1:-1]

    # conversion au format dataframe (index, track_id, hashtag)
    idx_fnd = []
    trk_fnd = []
    hsh_fnd = []

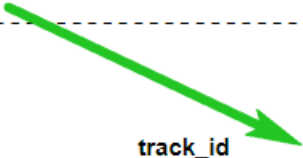
    for res in similar_tunes_idx:
        tune_found = get_trackid_by_index(res)
        hashtags_found = get_hashtags_by_index(res)
        idx_fnd.append(res)
        trk_fnd.append(tune_found)
        hsh_fnd.append(get_hashtags_by_index(res))
    return pd.DataFrame({'idx' : idx_fnd, 'track_id' : trk_fnd, 'hashtags' : hsh_fnd})
```

Cette fonction permet, à partir d'un morceau « track_id » d'obtenir « nb » morceaux similaires avec les infos suivantes : index (dans la matrice de similarité), track_id, hashtags.

Exemple : 10 morceaux similaires au morceau « c0a3db38bd9f8f8c44ed0c2d0e1b60af »

morceau : [83847] : track_id = c0a3db38bd9f8f8c44ed0c2d0e1b60af
hashtags: rock pops 70s

résultats:

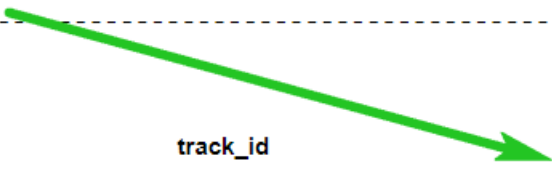


	idx	track_id	hashtags
0	1	0000e47c1207e2c637a44753a713456f	rock pops 70s
1	83847	c0a3db38bd9f8f8c44ed0c2d0e1b60af	rock pops 70s
2	30054	45138db5d3e84a423b12650a0f65d4a6	rock 70s pops
3	44206	65929a3541868bf88b85bdc16e86e1d0	rock pops 60s
4	24288	37bc24b5c16cc1f91a1049d6ebed410b	pops 00s
5	71372	a4196978d57dd77ce85628bc86cb27d1	rock 70s
6	60588	8b4146285e3ca27693c3b1cf81851e1c	rock hardrock 70s
7	43678	646a1e0d1b8ceb54ce87f017d4357071	rock hardrock 70s
8	1786	03e508ca8989f648f305283100388b6c	rock hardrock 70s
9	102600	ec3ac9eb8e3df884700b76c70c252b5f	rock hardrock 70s

Cependant, nous voyons que lorsque le hashtag est trop isolé les résultats ne sont pas cohérents.

```
morceau : [ 93933 ] : track_id = d801089f2dea97bb0ea5b95c875465e0
hashtags: juevesmeloso
```

résultats:

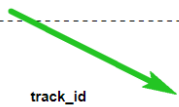


idx	track_id	hashtags
0 93933	d801089f2dea97bb0ea5b95c875465e0	juevesmeloso
1 37069	5545075b7f3a0bae8d14c9468b79abe4	streetstyleradio homegrownradio ryanlocker sav...
2 37058	553e1e757b2d176c973b353e075a3b1d	metalcore
3 37059	553e4cb281ad399347db86d2bde112db	bristol listeningnow 90smusic bbr bass
4 37060	553e4f0e9dde8a757e21ce87f7ae8902	love funklove 1rockhour nocturnalclassicswithl...

Et parfois il y a énormément de hashtags renseignés pour un même morceau, sans rapport entre eux, par conséquent les résultats sont mauvais :

```
morceau : [ 104841 ] : track_id = f16e9029afe5e148d2dffcf88a55603b
hashtags: hiltonhead savannah progressiveradio toulouse sanjose phoenix word beatbc ontheair winnipeg belfast brunswick freshly
squeezed yourmusic aracityradio rock itunes metal bringinitback manchester rheims orleans kamloops pittsburgh anchorage takeove
rtext 2002年の音楽 radiocidadeoficial tocandonacidade fuerzamartes cancióndeviernes cancióneldía musicislife nanaimo liverpool
köln lille tacoma fragradio gaming foofighters turnitup turnup rockon tocando radio98rock brazil atacalaradio allmylife tsgonai
r rcbs
```

résultats:



idx	track_id	hashtags
0 104841	f16e9029afe5e148d2dffcf88a55603b	hiltonhead savannah progressiveradio toulouse ...
1 35674	51f31c2eb4aa9fbcb9f6ddc6f716416	savannah hiltonhead ontheair belfast brunswick...
2 65829	9774a311af1b0e64e092e50f4a35ec2a	worldclassmusic savannah hiltonhead auckland b...
3 108216	f92726d6d470a7f911257132863bdf5c	progressiveradio savannah hiltonhead orleans l...
4 83168	bf1468972cc9e5360fb881d9c6bceda5	progressiveradio savannah hiltonhead buffalo o...

○ Conclusion sur les méthodes de clustering

Les deux approches permettent d'avoir des résultats intéressants, mais c'est surtout le champ lexical (modèle 3) qui permet les meilleures recommandations de morceaux.

Mais cette méthode est très dépendante des entrées utilisateurs, et nous avons constaté que le jeu de données contient énormément de hashtags inutiles (qui n'ont aucun sens réel et qui ne sont utilisés qu'une fois), ce qui représente un bruit énorme pour le modèle.

Pour avoir un modèle efficace, il faudrait d'abord trouver une méthode pour réduire le nombre de hashtags et ne garder que ceux qui ont du sens -ou regrouper ceux qui ont un sens proche sous un même hashtag- ce que nous n'avons pas réellement eu le temps de faire ici.

On peut dire que le clustering basé sur le champ lexical reste une approche logique et sans doute très efficace pour un système de recommandation musical, mais qu'il nécessite d'être couplé à un système de traitement du langage pour être sûr de pouvoir limiter la dimension des hashtags. Il faut aussi penser à la façon dont de nouveaux morceaux peuvent être intégrés au système (surtout s'ils ne contiennent pas déjà de hashtags spécifiques).

Eventuellement, un système hybride qui combine à la fois les variables de caractéristiques musicales et le champ lexical des morceaux pourrait apporter des résultats encore meilleurs...

Enfin, nous pourrions envisager un système de recommandation qui ne se base pas forcément sur du clustering mais directement sur les similarités musicales / lexicales. Par exemple, si un utilisateur écoute un morceau, on pourrait chercher dans la base de données les morceaux les plus proches musicalement (ou dans leur champ lexical) et proposer aléatoirement quelques morceaux parmi les plus proches. Cela aurait l'avantage d'être un système évolutif (puisque à chaque nouvelle écoute le modèle irait chercher de nouveaux morceaux par similarité) et ne nécessiterait pas forcément d'apprentissage (il n'y aurait pas clairement de groupes/clusters à définir). Néanmoins le temps nous a manqué pour explorer une telle approche.

Description des travaux réalisés

Répartition de l'effort sur la durée et dans l'équipe

Le déroulé de notre projet ne fut pas linéaire.

L'équipe était composée initialement de 3 personnes, nous tenions des points réguliers (~hebdomadaire) sur l'avancée du projet, les idées (et les tâches que nous nous attribuions), les démonstrations de tests, ...

Une 4^{ème} personne nous a rejoint peu avant Noël et a participé à nos réunions.

Mais passé Noël nous nous sommes retrouvés à deux (Nicolas & Lionel) pour poursuivre le projet, constituer les notebooks, fichier de code, et le rapport. Il semblerait que nos deux collègues aient abandonné la formation, ce qui a généré un accroissement de charges pour ceux qui restaient.

Bibliographie

The Best Document Similarity Algorithm: A Beginner's Guide

<https://towardsdatascience.com/the-best-document-similarity-algorithm-in-2020-a-beginners-guide-a01b9ef8cf05>

Définition du TF*IDF

<https://www.definitions-seo.com/definition-du-tfidf/>

TF-IDF Vectorizer scikit-learn

<https://medium.com/@cmukesh8688/tf-idf-vectorizer-scikit-learn-dbc0244a911a>

TF-IDF and similarity scores

https://goodboychan.github.io/python/datacamp/natural_language_processing/2020/07/17/04-TF-IDF-and-similarity-scores.html

Difficultés rencontrées lors du projet

La première difficulté a été de trouver la méthode adaptée à ce genre de problème (régression, clustering, vectorisation, ...). La seconde est liée au dataset, avec beaucoup de hashtags inutiles, et qui ne contient malheureusement pas les noms des morceaux de musique, ce qui limite grandement l'interprétation que l'on peut faire des résultats.

Bilan & Suite du projet

Ce projet nous a permis de découvrir différentes approches pour une même problématique, sur la base d'éléments appris durant la formation, et d'autres pistes que nous avons exploré sur les conseils de notre chef de cohorte.

Malgré la piètre qualité du dataset nous avons su mettre à profit les connaissances apprises.

Le processus par itération, complété des conseils de notre chef de cohorte, nous ont permis de progresser pas à pas.

Ainsi les enseignements de DataScientest sont devenus concrets, et nous pourrions continuer d'explorer ces pistes plus en détail par la suite.