Projet Big Data

Le rap français dans tous ses états

CARDONNE Thomas, KREMER Nathan, M'BARKI Sabri
M1 - ISCE

efrei PARIS
Ecole d'ingénieurs du numérique

Introduction

Pour ce projet de big data, nous avons décidé de nous intéresser au rap francais en nous focalisant sur l'analyse des paroles des rappeurs se situant dans les tendances actuelles. Nous allons dans ce rapport détaille la manière dont nous avons extrait des données pour ensuite les traiter et les analyser afin d'en tirer certaines conclusions.
Nous avons utiliser plusieurs outils dédiés au big data le tout dans un environnement en Python. Tout le code et les datasets que nous avons créés sont disponibles sur le repository suivant https://github.com/deverlabs/m1-bigdata ou bien dans le dossier parent de cette archive.

Récupération des données

Rappeurs

Pour récupérer les données nécessaires à notre étude nous avons utilisé deux API de deux acteurs du monde de la musique : Spotify et Genius.
Le code relatif à cette récupération de données se situe dans le sous dossier "*scrapper*". Pour le faire fonctionner, il faut tout d'abord copier le fichier d'environnement ".example.env" et le dupliquer en un fichier appelé ".env". Se sont les variables d'environnement utilisées par le programme.

Il faut y insérer:

- Une clé d'API Spotify
- Une clé d'API Genius
- Et le nombre maximum d'artistes dont on souhaite récupérer les musiques

Une fois cela fait on peut lancer le programme avec la commande python getsongs.py.

Voici un aperçu de la sortie de l'application :

```
Get rappers from :
-  PVNCHLNRS
-  Fresh Rap

! Limit reached !

** Found 10 rappers **
PNL
Ninho
Niska
RK
Djadja & Dinaz
Dabs
Jok'air
Zola
Diddi Trix
Columbine
****

# Progress : 1/10

* Scrapping songs of PNL *
- DA
- Le monde ou rien
- Naha
- Jusqu'au dernier gramme
- Onizuka
- Au DD
- Bené
- À l'ammoniaque
- Oh Lala
- J'suis QLF

# Progress : 2/10

* Scrapping songs of Ninho *
- Dis moi que tu m'aimes
- De l'autre côté
- Mamacita
```

Voici comment fonctionne ce programme

Dans un premier temps, nous allons fetcher un certain nombre de playlist rap francais contenues sur Spotify afin d'en récupérer le nom des artistes. Nous allons également faire plusieurs autres requêtes à l'API spotify afin d'obtenir ces informations :

- La popularité de l'artiste (0-100)
- Son nombre de followers
- Sa photo de profil

Toutes ces données sont rangées sous forme d'objet. On insérer également un tableau dont la clé est "songs" qui servira plus tard a stocker toutes les musiques de l'artiste en question.

Exemple:


```
{
  "PNL": {
    "popularity": 83,
    "image": "https://i.scdn.co/image/5183e0a003fb8679a63fbdac311df70c79bc9851",
    "followers": 1753753,
    "songs": []
  }
}
```


Une fois le nombre d'artistes désirés obtenu, nous pouvons passer à la deuxième étape, récupérer leurs musiques.

Paroles

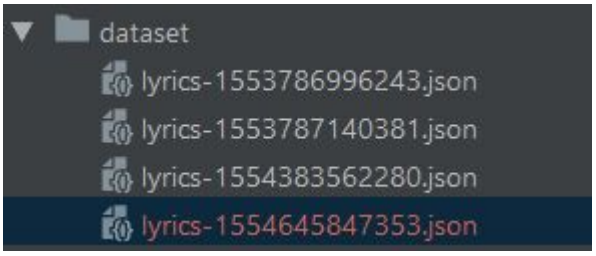
Nous allons donc utiliser le peu d'endpoints disponibles sur l'api de Genius, célèbre site collaboratif répertoriant des millions de paroles de diverses artistes.
On commence par récupérer le top 10 chansons de l'artiste que l'on souhaite

Exemple:

r = requests.get("https://api.genius.com/search?q=PNL, headers=headers)

Cette requête va retourner un objet contenant en autres le titre de la chanson ainsi que le lien permettant d'accéder aux paroles de cette musique sur le site Genius.
Le problème, aucun endpoint permet d'obtenir les paroles de la musique. On a donc improviser et créer un fonction permettant de tirer les paroles depuis le code HTML de Genius.

Une fois fait, on insère ces données dans l'objet de l'artiste en question qui au final donnera un fichier contenant tous les lyrics nommé lyrics-{{timestamp}}.json



Voici un aperçu du contenu de ces fichiers:


```
{
  "PNL": {
    "popularity": 83,
    "image": "https://i.scdn.co/image/5183e0a003fb8679a63fbdac311df70c79bc9851",
    "followers": 1753753,
    "songs": [
      {
        "song": "DA",
        "lyrics": "\n\n[Couplet 1 - Ademo]\nMa frappe y'a personne qui l\u2019arr\u00eate\nPenalty j'souris au gardien\nIgo y
      },
      {
        "song": "Le monde ou rien",
        "lyrics": "\n\n[Couplet 1 : Ademo]\nJ'veux du L, j'veux du V, j'veux du G, pour dessaper ta racli\nIgo, on est vou\u0
      },
      {
        "song": "Naha",
        "lyrics": "\n\n[Produit par BBP]\n\n[Couplet 1 : N.O.S]\nMes gouttes de sueur ont l'odeur d'l'Enfer\n\u00c7a r'commer
      },
      {
        "song": "Jusqu'au dernier gramme",
        "lyrics": "\n\n[Couplet 1 : Ademo]\nJe suis \u00e0 91 mille lieues sous la merde\nJ'ai mille eu' sous la semelle\nJ'a
      },
      {
        "song": "Onizuka",
        "lyrics": "\n\n[Produit par BBP & Dolor]\n\n[Couplet 1 : N.O.S]\nJ'fais les ronds, j'fais les ronds, j'fais les ronds
      },
      {
        "song": "Au DD",
        "lyrics": "\n\n[Couplet 1 : Ademo]\nBats les couilles d'l'Himalaya\nBats les couilles, j'vise plus l'sommet\nMon c\u0
      },
      {
        "song": "Ben\u00e9",
        "lyrics": "\n\n[Intro : Ademo]\nBen\u00e9 Ben\u00e9, Ben\u00e9 Ben\u00e9, Ben\u00e9 Ben\u00e9, Ben\u00e9 Ben\u00e9\nB
      },
      {
        "song": "\u00c0 l'ammoniaque",
        "lyrics": "\n\n[Couplet 1 : Ademo]\nOuais, ouais, ouais, ouais, ouais\nOuais, c'est l'd\u00e9sert dans la te-t\u00ea\
      },
```


Nous avons donc fait tourner ce programme pendant plusieurs heures afin d'obtenir un dataset convenable pour notre étude. Nous avons un total de 112 rappeurs et 1188 musiques à analyser

Pas trop lent, mais pas trop vite !

Durant le développement du scrapper, nous avons utilisé le script Python sur nos ordinateurs en limitant le nombre d'artistes récupérés à 10. Afin d'avoir un dataset conséquent, il a fallu augmenter ce nombre. Or, plus on augmente ce nombre, plus notre script effectue de requêtes extérieures et doit traiter de données. Afin d'optimiser un peu le temps de scrapping, nous avons décidé de faire tourner le script sur un serveur hébergé dans un datacenter, et donc d'être plus "proche" des APIs mais surtout de bénéficier de processeurs plus puissants.
Afin de pouvoir faire tourner rapidement notre script Python sur un serveur sans y installer toutes les librairies nécessaires, nous avons opté pour la technologie Docker, qui grossièrement permet de cloisonner notre application avec le reste du système (ie, pour ne pas polluer le système hôte avec les librairies qui ne seront utilisées qu'avec ce projet).

En utilisant des serveurs plus puissants, nous avons touché la limite de requêtes par période de temps (rate-limit) avec les APIs. Pour pallier à ce problème, nous avons implémenté un algorithme permettant lorsque le serveur nous dit de ralentir nos requêtes, d'attendre un certain nombre de seconde afin de recommencer.

Analyse

Pour analyser les données extraites, nous avons utilisé Jupyter permettant de créer des notebooks. C'est une application web très utilisé dans le domaine de la data science.

Après avoir lu les données, il a fallu transformer notre dataset JSON en objet Dataframe (pandas). La première problématique a été de convertir les paroles dans un format plus pratique : le texte doit être séparé en un tableau de mots, ou de tokens. Il a été nécessaire de trouver un pattern pour pouvoir séparer les mots dans un tableau.

Ensuite, nous devons filtrer tout ce qui n'est pas désirable et qui pourrait entraver les résultats de nos recherches.
Il faut supprimer ce qui s'appel "stopwords", tous les mots courants utilisés dans la langue française, comme par exemple les "le", "la", "c'est" et qui n'apporte pas d'informations sur le contexte de la phrase.
Pour cela nous utilisons notre propre dataset de mots contenus dans une liste ainsi que le package nltk utilisé pour le natural language.
Une fois les données filtrées, nous allons pour chaque chanson extraire le nombre unique de mots utilisés.
Nous avons utilisé la puissance de la librairie Pandas pour stocker les morceaux et les données associées (paroles, quelques statistiques sur les paroles et les lyrics).
On manipule toutes ces données à l'aide de Pandas et on affiche ensuite les résultats sous forme de tableau :

```
-- Processed stats
> Count songs 1189
> Count raw words : 306107
> Count filtered words : 207170
>  67.67894886428601 % kept
```

Out[17]:

	artist	artist_followers	artist_image	artist_popularity	count_filtered_words	count_raw_words	filtered_tokens	lyrics	name
0	PNL	1697186	https://i.scdn.co/image/5183e0a003fb8679a63fbd...	80	179	288	[moche, espèce, m'aiment, temps, Ciseaux, semb...	\n\n[Couplet 1 - Ademo]\nMa frappe y'a personn...	DA
1	PNL	1697186	https://i.scdn.co/image/5183e0a003fb8679a63fbd...	80	127	206	[souffle, balade, J'nique, l'goût, p'tit, chic...	\n\n[Couplet 1 : Ademo]\nJ'veux du L, j'veux d...	Le monde ou rien
2	PNL	1697186	https://i.scdn.co/image/5183e0a003fb8679a63fbd...	80	139	234	[parlu, merde, J'vois, connais, pète, pue, mai...	\n\n[Produit par BBP]\n\n[Couplet 1 : N.O.S]\n...	Naha
3	PNL	1697186	https://i.scdn.co/image/5183e0a003fb8679a63fbd...	80	212	335	[sentiments, Fleur, jette, l'orage, là-bas, ma...	\n\n[Couplet 1 : Ademo]\nJe suis à 91 mille li...	Jusqu'au dernier gramme
4	PNL	1697186	https://i.scdn.co/image/5183e0a003fb8679a63fbd...	80	105	181	[J'vois, Ouh, gronde, prince, ciel, matin, *kr...	\n\n[Produit par BBP & Dolor]\n\n[Couplet 1 : ...	Onizuka
5	PNL	1697186	https://i.scdn.co/image/5183e0a003fb8679a63fbd...	80	126	204	[crier, polícia, temps, J'roule, J'vois, Bille...	\n\n[Intro : Ademo]\nBené Bené, Bené Bené, Ben...	Bené
6	PNL	1697186	https://i.scdn.co/image/5183e0a003fb8679a63fbd...	80	127	221	[sentiments, temps, vallée, l'désert, qu'hier,...	\n\n[Couplet 1 : Ademo]\nOuais, ouais, ouais, ...	À l'ammoniaque
7	PNL	1697186	https://i.scdn.co/image/5183e0a003fb8679a63fbd...	80	192	303	[connais, Bats, ouhlalala, Paname, laisse, J'm...	\n\n[Couplet 1 : Ademo]\nBats les couilles d'l...	Au DD

Nous avons trouvé intéressant d'analyser la donnée suivante:

Il y a t'il une corrélation entre le vocabulaire des rappeurs et leur popularité ?

Pour le savoir, une fois les mots filtrer, nous avons concaténer les résultats par rappeurs:

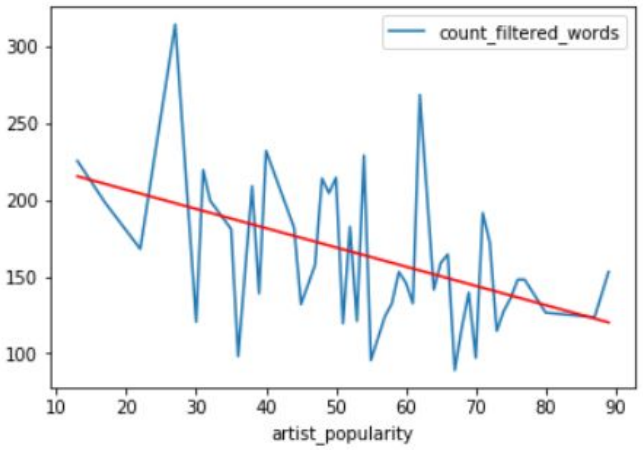
```
In [7]: dfGroupArtist = df.groupby(['artist', 'artist_popularity', 'artist_followers', 'artist_image'])

        # Artist using the most words

        dfGroupArtist.sum().sort_values(by='count_filtered_words', ascending=False)
```

Out[7]:

				count_filtered_words	count_raw_words
artist	artist_popularity	artist_followers	artist_image		
Passi	49	33350	https://i.scdn.co/image/1df8a361f72369740a8f382fe3c0a39f4e04f42a	4365	4498
Akhenaton	50	69426	https://i.scdn.co/image/875785de032a61f1c74fe4ea135bea68a9a95660	3688	5131
Shurik'n	48	51342	https://i.scdn.co/image/678e390fdaf1586d50212620244485e076f64349	3533	5003
Rohff	62	310781	https://i.scdn.co/image/40c0a47e15a25fa800e3d0c0a610f5337652ff2f	3381	4764
Fonky Family	49	88165	https://i.scdn.co/image/5748576cb2f3b52ec7bfef7ab00df03f157e755a	3190	4734
R.E.D.K.	38	4070	https://i.scdn.co/image/46d83fd793525951ea76f5208ceb966f4329f45d	3103	4151
III.	27	1105	https://i.scdn.co/image/815715e76d8554e20d5e4be87e71c159f1b20903	3022	3173
Medine	60	127325	https://i.scdn.co/image/48bb9122675967e8b73a72b1210f7c65a33d7c39	3001	4272
SNIPER	54	152262	https://i.scdn.co/image/ad059b35d870a547ef943a2226786f2601bea431	2938	4314
Ideal J	35	16189	https://i.scdn.co/image/d0a0b5f73992129f921948f41c5ef3961ba14ea9	2674	4128
Orelsan	72	915483	https://i.scdn.co/image/1820263e45a4116112e5a2f028b1873ea0668ea8	2617	3848
Lunatic	48	68566	https://i.scdn.co/image/adbcf4490d83e5a124134ff7b9ef4a8d3efa3301	2469	3634
Kheops	36	12425	https://i.scdn.co/image/580848c0e727629780f3efe35fc7605b387b7976	2456	3116
Zola	65	91349	https://i.scdn.co/image/4685fb04618894c63c863b42e643ece3ef2128f1	2435	2598
Rémy	54	50616	https://i.scdn.co/image/96070824471618ff1d4b7c2a45279cda20884073	2423	2806
La Caution	40	9276	https://i.scdn.co/image/1f901fcba5ac99d0ddd21369bab92db15acfb0fd	2411	3604
Nekfeu	71	754355	https://i.scdn.co/image/4b29b23a7dc7ac866210d93d4a5287c95941696c	2394	3675
Le Rat Luciano	44	15542	https://i.scdn.co/image/0ae036941e6430bfd8c906d280a55f6062c6f855	2359	3738

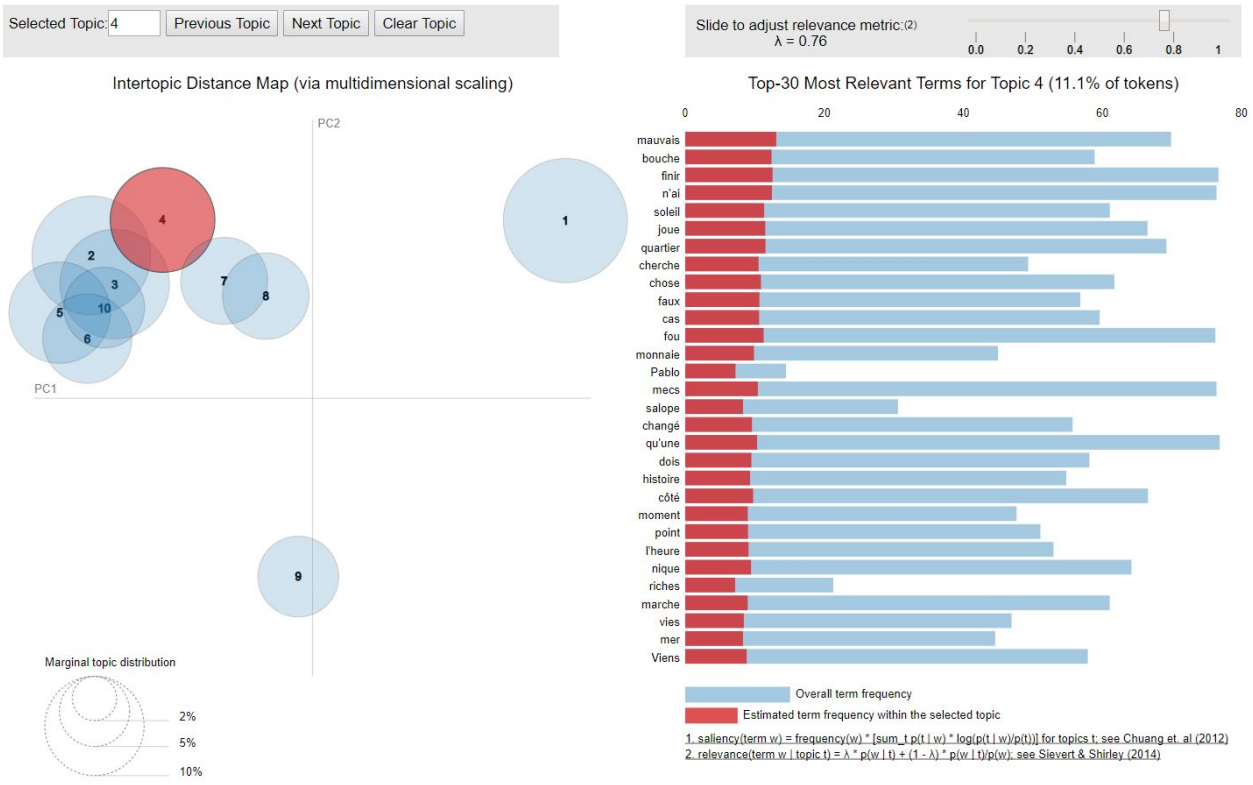
Nous avons ensuite tracé une courbe avec matplotlib montrant la popularité de l'artiste en fonction de son vocabulaire

```
Out[12]: [<matplotlib.lines.Line2D at 0x1a209ceac8>]
```



Nous pouvons clairement voir la tendance que moins l'artiste utilise un vocabulaire varié, plus il est populaire.

Nous avons ensuite effectuer un Topic modelling à l'aide de Gensim permettant d'extraire et de classifier les thèmes abordés par les rappeurs dans leurs paroles :



Pour finir en beauté et avoir un retour visuel sur notre travail, nous avons effectué un WordCloud pour voir les mots les plus utilisés par les rappeurs. Pour cela nous avons utilisé matplotlib et un masque en forme de pistolet pour l'esprit rap.

Dernière donnée analysée, nous avons créer notre propre dataset contenant un grand nombre d'appellations différentes pour diverses drogues. Nous nous sommes servis de ce dataset afin de compter le nombre d'occurence dans les textes afin de voir lesquelles sont le plus citées :

Out[23]:

shit	166
came	61
beuh	59
bang	52
coke	43
weed	43
dope	38
blanche	36
coco	23
défonce	21
charge	19
pot	19
crack	18
douce	17
spliff	17
neige	16
bédo	16
gaz	16
tise	16
jaune	16
pétard	13

Conclusion

Nous avons trouvé très pertinent le fait que le sujet de ce projet était libre, cela nous a permis de nous orienter vers ce qui nous plaisait. Ce fut pour nous une première expérience dans le big data, les outils et les techniques utilisées pour soutirer des informations aux données qui à première vue, n'en contiennent pas.

Nos notebooks extraits de Jupyter sont disponible dans un fichier au format HTML appelé analysis.html