

Rapport de Modal : Exploration de réseaux de similarité musicale

CSC_43M02_EP

Arthur Fournier, Arthur Buis

24 mai 2025

[Lien du dépôt Github](#)

TABLE DES MATIÈRES

1	Introduction	2
2	Récupération des données	3
2.1	Sources de données	3
2.2	Choix de la structure de données	4
2.3	Artistes et collaborations	4
2.3.1	Recherche d'artistes et filtrage	4
2.3.2	Filtrer des données extraites	5
2.3.3	Récupération des featurings	5
2.4	Récupération des paroles	6
3	Construction des graphes	7
3.1	Graphe des collaborations	7
3.1.1	Nettoyage du graphe	7
3.1.2	Statistiques du graphe	7
3.1.3	Clustering	8
3.2	Graphe de similarité Last.FM	9
3.3	Graphe des embeddings	12
4	Interprétation des résultats	14
4.1	Analyse approfondie du graphe de collaboration et des communautés	14
4.2	Retranscription de l'évolution du rap à travers le temps	15
4.3	Mise en relation avec les deux autres graphes	16
4.3.1	Graphe de similarité Last.FM	16
4.3.2	Graphe des embeddings	17
4.4	La popularité et le nombre de followers Spotify caractérisent-ils aussi bien l'importance des artistes ?	18
5	Conclusion	20
A	Annexes	21

1

INTRODUCTION

L'univers de la musique, particulièrement aujourd'hui avec le règne des plateformes de streaming, constitue un véritable puits de données. Chacune de nos écoutes est tracée et permet aux plateformes de construire des algorithmes de recommandation des plus affutés. Mais sans utiliser une quantité de données aussi colossale, peut-on établir des similarités artistiques ?

La musique constitue un terrain riche pour l'analyse de réseaux et de contenu textuel, car les artistes y collaborent fréquemment, formant des structures communautaires denses. En particulier dans le rap, la collaboration est devenue banale avec l'ère du streaming, voire nécessaire pour toucher de nouveaux auditeurs. Il semble ainsi plus accessible de classer les rappeurs, liés par ces collaborations, que de classer des groupes de rock ou des chanteurs de variété.

En tant que passionnés de musique et d'analyse statistique, nous avons donc choisi d'effectuer un projet de cartographie du rap français. Ce travail a pour but de mettre en avant les collaborations entre artistes du rap francophone au moyen de graphes, d'en extraire des structures pertinentes (communautés, noeuds à forte centralité), puis d'analyser le contenu lexical des textes des artistes à l'aide d'outils de traitement automatique du langage naturel (NLP). À partir des embeddings des paroles, nous explorerons les similarités sémantiques entre artistes et construirons des représentations thématiques des clusters identifiés.

2 RÉCUPÉRATION DES DONNÉES

2.1 SOURCES DE DONNÉES

Commençons par les sources de données que nous avions à notre disposition avec le géant du streaming Spotify, qui propose une API qui permet de rechercher des artistes et d'obtenir des informations relatives à leur genre musical, en plus de statistiques comme leur popularité ou encore leur nombre de followers.

Ensuite, nous avons utilisé le site web Genius [1]. Genius est une plateforme collaborative de paroles de chansons, qui fournit également des métadonnées sur les artistes, notamment une liste presque exhaustive de leurs collaborations. Cette source a été essentielle pour l'extraction des textes de chansons et pour construire notre graphe de collaborations, fondé sur la présence conjointe d'artistes dans des morceaux. Contrairement à Spotify, elle est open source et très complète, grâce au dévouement des utilisateurs qui contribuent régulièrement à la maintenir à jour, en particulier dans la catégorie du rap. La plateforme propose également une API qui permet d'effectuer des requêtes sur le site Genius via un programme python, et d'extraire des données sur les artistes sans avoir à effectuer de scraping. Au vu de l'importance des données Genius dans notre projet, nous avons décidé d'exclure de nos graphes tout artiste n'apparaissant pas sur ce site. La quasi-totalité des artistes d'intérêts y est bien présente, donc cela nous a simplement permis d'éliminer quelques noeuds problématiques dans notre graphe.

Pour pouvoir comparer nos résultats en fin de projet avec la similarité obtenue en se basant sur les statistiques de streaming, nous utilisons aussi l'API de Last.FM. Last.FM est un traiteur d'écoute, qui recense chaque titre écouté sur diverses plateformes grâce à un mécanisme de "scrobbing". Il permet notamment d'obtenir une liste d'artistes similaires à un artiste donné, ainsi qu'un score de similarité basé sur les comportements d'écoute des utilisateurs. La base de données Last.FM compte plus de 300 milliards d'écoute recensées à ce jour, donc nous pouvons attendre de leur algorithme de similarité une précision assez pointue.

Enfin, nous avons utilisé la base de données musicales MusicBrainz. De format assez similaire à Wikidata, elle ne prend pas en charge les requêtes SPARQL, mais peut être interrogée à l'aide d'une bibliothèque Python dédiée, et constitue une base collaborative orientée vers la précision bibliographique. Toutefois, elle s'est avérée assez incomplète et difficilement exploitable dans le cas particulier du rap français, en raison d'un manque de données structurées pour les artistes francophones récents ou émergents. Nous l'avons finalement utilisée seulement pour étendre notre liste d'artistes étudiés.

2.2 CHOIX DE LA STRUCTURE DE DONNÉES

Avant de nous lancer dans la collecte de données sur le web, il nous faut définir la structure de données qui sera utilisée. Nous avons comme ambition de récupérer plusieurs centaines d'artistes et de construire pour chacun un embedding de 300 paramètres que nous détaillerons dans une prochaine sous section. Pour stocker une telle quantité de données nous avons opté pour l'utilisation d'une base de données MongoDB qui est de type NoSQL. Bien qu'elle utilise un format de type JSON, contrairement à des fichiers CSV ou JSON elle est utile pour des données hétérogènes (ce qui est le cas avant que nous ne mettions certaines contraintes pour filtrer nos données). MongoDB est également très puissant et possède de nombreux outils utiles permettant notamment des requêtes rapides et évite de lire un fichier ligne par ligne ou de tout charger en mémoire. Un exemple de ces données est disponible en [figure 1](#).

2.3 ARTISTES ET COLLABORATIONS

La première étape de notre projet a consisté à identifier un ensemble pertinent d'artistes et à extraire l'ensemble des collaborations entre ces artistes. Pour cela, nous avons combiné des méthodes de recherche à partir de différentes sources, des filtres de pertinence ainsi qu'un raffinement manuel des données.

2.3.1 • RECHERCHE D'ARTISTES ET FILTRAGE

Pour la recherche d'artistes, nous avons d'abord utilisé une recherche par genre à partir de l'API Spotify, ce qui nous a permis d'identifier un premier ensemble de 300 artistes. Après avoir commencé à travailler avec cet ensemble initial, nous avons décidé d'élargir notre base avec MusicBrainz. La recherche via la bibliothèque Python de MusicBrainz peut fournir jusqu'à 800 résultats, mais elle souffre d'un bruit important : de nombreux doublons, ou bien des artistes peu pertinents pour notre étude, comme des artistes n'appartenant pas au milieu du rap, ou des artistes trop peu actifs, qui donneraient des noeuds isolés dans le graphe, sans valeur ajoutée analytique. Nous avons donc mis en place une étape de filtrage.

Pour filtrer la liste d'artistes extraite, nous avons couplé plusieurs critères : la présence d'informations suffisantes (identifiants Spotify et Genius au minimum), un nombre de followers minimal sur Spotify, et l'existence d'au moins une collaboration connue sur Genius (Attention le nombre de followers Spotify diffère du nombre d'auditeurs mensuels affiché sur la plateforme). Ce premier filtrage, que nous avons modifié au fur et à mesure du projet, nous a permis de converger vers une base de données très complète de 477 artistes. D'autres filtres dont nous discuterons dans une partie ultérieure ont été ensuite mis en place lors de l'étape de construction du graphe

2.3.2 • FILTRER DES DONNÉES EXTRAITES

Pour un nombre non négligeable d'artistes, la recherche de l'identifiant Genius à partir de leur nom échoue. Cela est principalement dû à l'algorithme de recherche de Genius, qui privilégie fortement les artistes très populaires dont le nom est proche de la requête, au détriment d'artistes moins connus dont le nom est identique à la requête. Or, disposer de l'identifiant Genius des artistes importants est absolument essentiel dans notre projet : c'est cet identifiant qui permet d'extraire les paroles et les collaborations, et donc de faire apparaître l'artiste dans notre graphe.

Nous avons donc consacré beaucoup de temps à ajuster les paramètres et les stratégies de la fonction de recherche (comme la normalisation des noms, l'ajout de tags de genre musical ou le contrôle manuel des premiers résultats). Malgré ces efforts, certains cas irréductibles subsistent, incluant pourtant quelques figures notables du rap français (comme Rim'K ou Sexion d'assaut, qui ont des apostrophes dans leur nom). Pour traiter ces cas, nous avons écrit un programme qui permet de sélectionner manuellement l'artiste correct parmi les suggestions de recherche Genius, pour s'assurer de leur intégration dans notre base (fonction `fail_update_mongo` dans `update.py`).

```

_id: ObjectId('68318ba10792b9082a6ec932')
name : "Soprano"
id_spotify : "2RJBv9wXbW6m539q9N0fW1"
followers : 3696547
popularity : 65
id_genius : 1431
url_genius : "https://genius.com/artists/Soprano"
id_mb : "e52815f0-0db9-49b2-acb7-1af6b2dfa936"
embedding : Array (300)
  0: -0.7093960642814636
  1: 0.30433839559555054
  2: 0.6110067000512212

```

FIGURE 1 – Exemple de données stockées sur MongoDB. followers et popularity sont issus de Spotify, id_mb est l'identifiant musicbrainz et _id l'identifiant dans notre base MongoDB

2.3.3 • RÉCUPÉRATION DES FEATURINGS

Pour récupérer les collaborations, nous avons utilisé l'API Genius pour extraire toutes les collaborations de chaque artiste. Une étape de filtrage a alors été appliquée pour ne conserver que les collaborations impliquant au moins deux artistes déjà présents dans notre base, en prenant soin d'ajouter une et une seule fois chaque featuring, en relevant au fur et à mesure les identifiants Genius de chaque titre ajouté. Nous avons identifié avec cette méthode environ 16739 featurings [figure 14 et figure 15 en Annexe], dont 7822 restants après l'étape de filtrage : une quantité largement suffisante pour obtenir un graphe pertinent !

2.4 RÉCUPÉRATION DES PAROLES

Puisque l’API Genius donne un accès assez privilégié aux paroles des morceaux de chaque artiste, nous avons tenté d’élargir le champ de notre étude à une dimension sémantique.

Pour exploiter les textes de manière fine, nous nous sommes appuyés sur un modèle d’embedding spécialisé : un modèle Word2Vec entraîné spécifiquement sur des textes de rap français, nommé *Word2Bezbar-large* [2], développé par l’équipe Rapminerz en 2024. Il convertit les données d’entrée dans un espace d’arrivée de dimension 300, et capture les particularités lexicales, stylistiques et culturelles du rap francophone. Trouver cet outil nous a largement motivé à essayer notre seconde approche de clustering par analyse sémantique.

Dans notre approche, l’idée était d’insérer une arête entre deux artistes seulement si la similarité cosinus de leur embedding dépassait un seuil donné. Le poids de l’arête serait alors proportionnel à cette similarité cosinus. Nous discutons plus en détail cette approche dans la sous section 3.3.

3 CONSTRUCTION DES GRAPHES

3.1 GRAPHE DES COLLABORATIONS

Commençons par le graphe des collaborations, représentant chaque artiste par un noeud et les liant par une arête s'ils ont collaboré ensemble, avec un poids égal au nombre de collaboration. Nous utilisons pour cela la bibliothèque python *NetworkX* qui génère un graphe à partir de nos artistes, que nous nettoyons avant l'utilisation d'algorithmes de clustering. Nous visualisons ensuite notre graphe avec le logiciel *Gephi*, qui propose de nombreux outils facilitant sa lecture et les calculs de ses propriétés.

3.1.1 • NETTOYAGE DU GRAPHE

Pour nettoyer le graphe et éviter de traiter des noeuds peu pertinents, nous avons testé plusieurs approches :

- **Suppression des nœuds isolés** : Cette méthode s'est montrée très peu efficace, car elle n'enlève par exemple pas les composantes connexes à 2 artistes, mais supprime les artistes en périphérie du cluster principal. Nous ne l'avons donc pas retenue étant donné que l'extraction des données depuis le web nous donnait plusieurs artistes peu connus, parfois loin du milieu du rap que nous en souhaitions pas analyser.
- **Suppression des nœuds de faible degré** : Dans cette méthode nous éliminons les noeuds ayant un degré inférieur à un certain degré k . Elle permet de mieux éliminer les petites composantes connexes, mais au vu de leur forte connectivité, un k assez grand est requis et supprime malheureusement du même coup des artistes importants ayant peu collaboré avec des artistes du cluster principal.
- **Suppression des petites composantes connexes** : Ici, nous éliminons seulement les composantes connexes ayant un cardinal inférieur à un certain entier k . Cette méthode est la plus efficace, elle élimine en pratique toutes les composantes connexes parasites pour $k = 10$, ce qui est notre cas, nous donnant un graphe entièrement connexe d'artistes de rap français.

3.1.2 • STATISTIQUES DU GRAPHE

Notre algorithme calcule également certaines données sur notre graphe, que nous avons affichons sous format matplotlib [[figure 2](#), [figure 3](#) et [figure 4](#)] ou dans le terminal [[figure 14](#) en Annexe] :

- 358 noeuds
- 3955 arêtes

- Un degré moyen de 22.09. Les artistes ont donc une moyenne de 22 collaborations, éventuellement plusieurs fois avec un même artiste.
- Une seule composante connexe

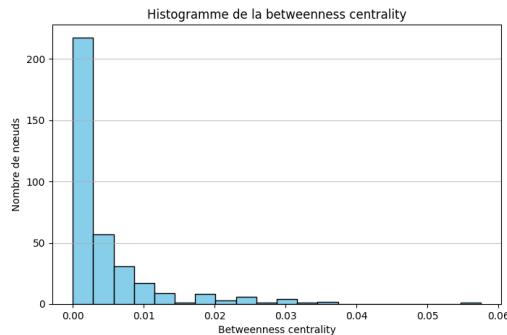


FIGURE 2 – Histogramme de la betweenness centrality

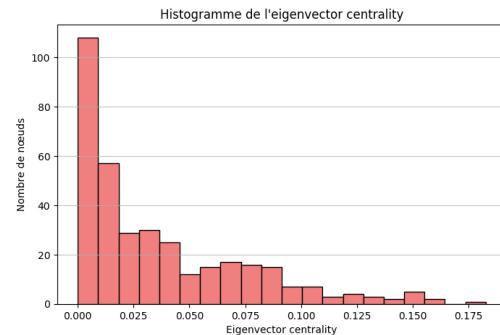


FIGURE 3 – Histogramme de l'eigenvector centrality

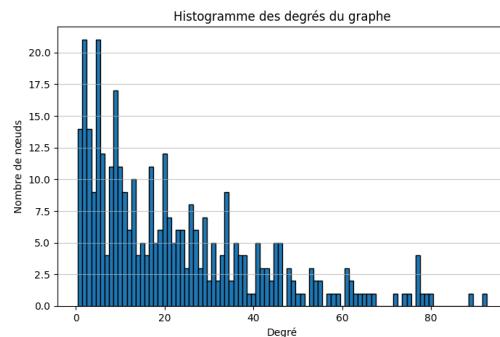


FIGURE 4 – Répartition des degrés dans le graphe

Gephi propose aussi des calculs des statistiques du graphes mais présente les données de manière moins lisible qu'en utilisant les bibliothèques python comme pour les graphes ci-dessus.

3.1.3 • CLUSTERING

Maintenant que nous avons un graphe propre, nous pouvons lancer un algorithme de clustering, en utilisant la bibliothèque *NetworkX*. Nous nous sommes concentrés sur l'algorithme de Louvain qui donnait des résultats plus pertinents et une partition selon nous plus juste au vu de notre connaissance du genre musical [figure 5].

On y retrouve en **violet** le rap urbain (110 artistes), en **orange** la scène marseillaise (49 artistes), en **bleu** le rap conscient (70 artistes), en **vert** le rap old school (35 artistes), en **jaune** le rap old school ayant fait la transition avec la nouvelle génération et en majorité plus jeune

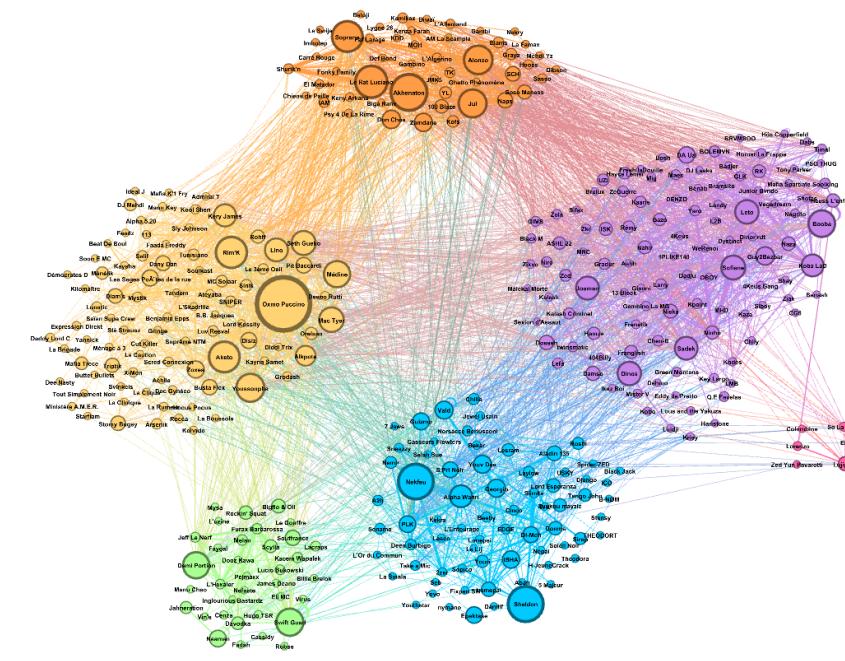


FIGURE 5 – Graphe des collaborations obtenu après coloration des clusters et attribution d'une taille proportionnelle à la betweenness centrality des nœuds sur Gephi.

(88 artistes) et enfin en **rose** quelques artistes alternatifs mis à part par l'algorithme de *Louvain* que nous aurions inclus dans le cluster bleu (6 artistes) [figure 6].

En terme d'affichage nous avons choisi de laisser les clusters séparés dans l'espace et de ne pas appliquer de spatialisation : à cause de la forte inter-connectivité des différents clusters, nous obtiendrions un graphe avec des clusters fortement entremêlés nous empêchant de bien distinguer les artistes qui sont dans leur périphérie.

3.2 GRAPHE DE SIMILARITÉ LAST.FM

Dans le but de pouvoir comparer les clusters de notre graphe de collaborations avec des données d'écoute utilisateur réelles, nous avons construit un graphe dont les arêtes seraient pour chaque artiste les artistes les plus similaires selon le score de similarité Last.FM (introduit en section 2.1).

Pour obtenir un graphe contenant suffisamment d'information sans être trop dense, nous avons décidé de rechercher pour chaque artiste ses 20 artistes les plus similaires selon le score Last.FM, et d'ajouter des arêtes vers les artistes parmi ces 20 qui sont bien dans notre base de données. Pour les artistes les plus populaires, on obtient le plus souvent un degré supérieur à 20, mais pour les moins connus ce degré descend jusqu'à 0. On nettoie une nouvelle fois le graphe en retirant les composantes connexes de taille inférieure à 5, car les noeuds isolés sur le

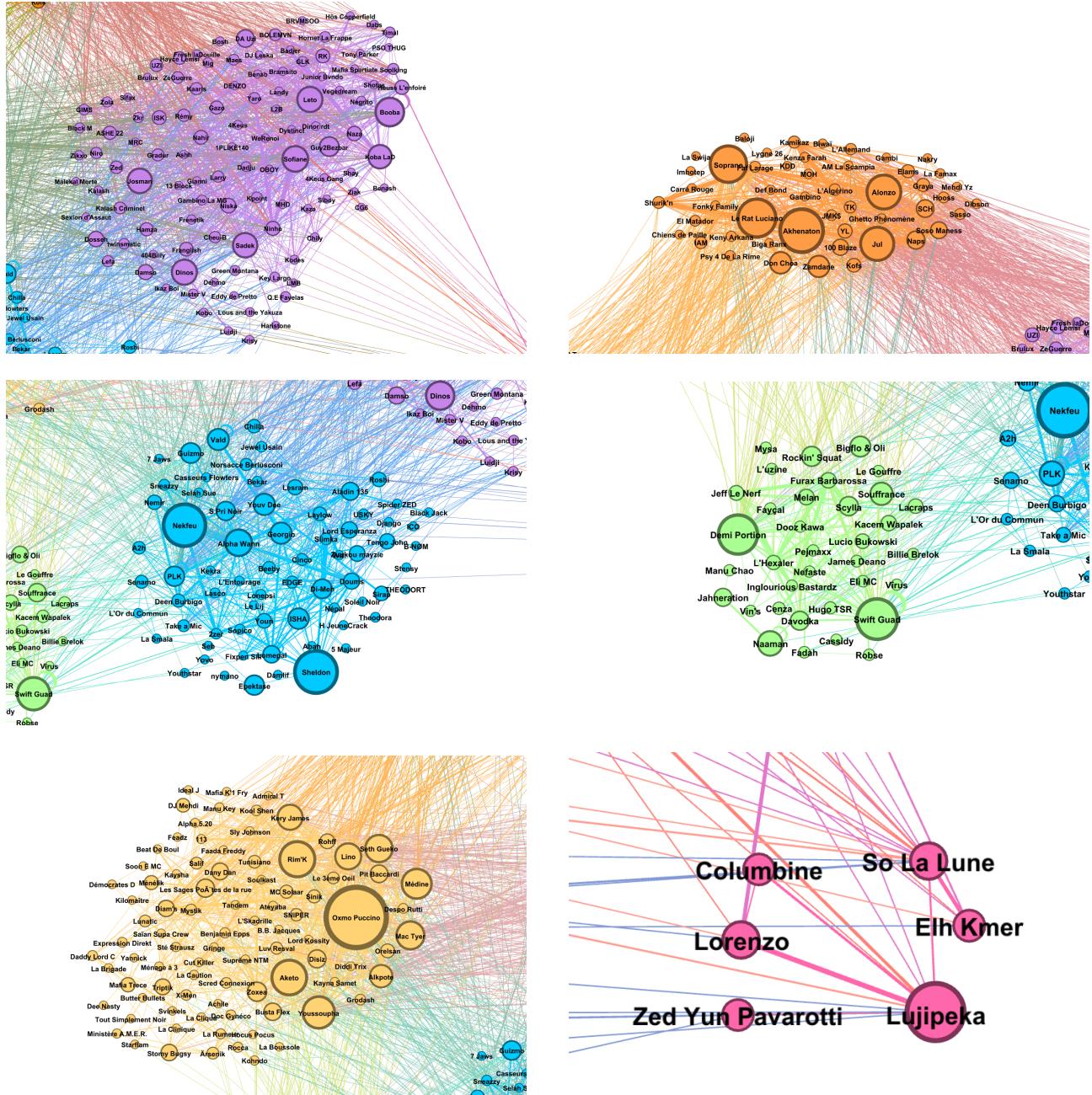


FIGURE 6 – Zoom sur les 6 différents clusters avec une taille proportionnelle à la betweenness centrality

graphes des collaborations et sur le graphe Last.FM ont de bonnes chances d'être les mêmes.

Le graphe obtenu, dont les clusters sont une nouvelle fois calculés avec l'algorithme de Louvain, et avec une taille de noeuds proportionnelle à leur betweenness centrality, est en figure 7.

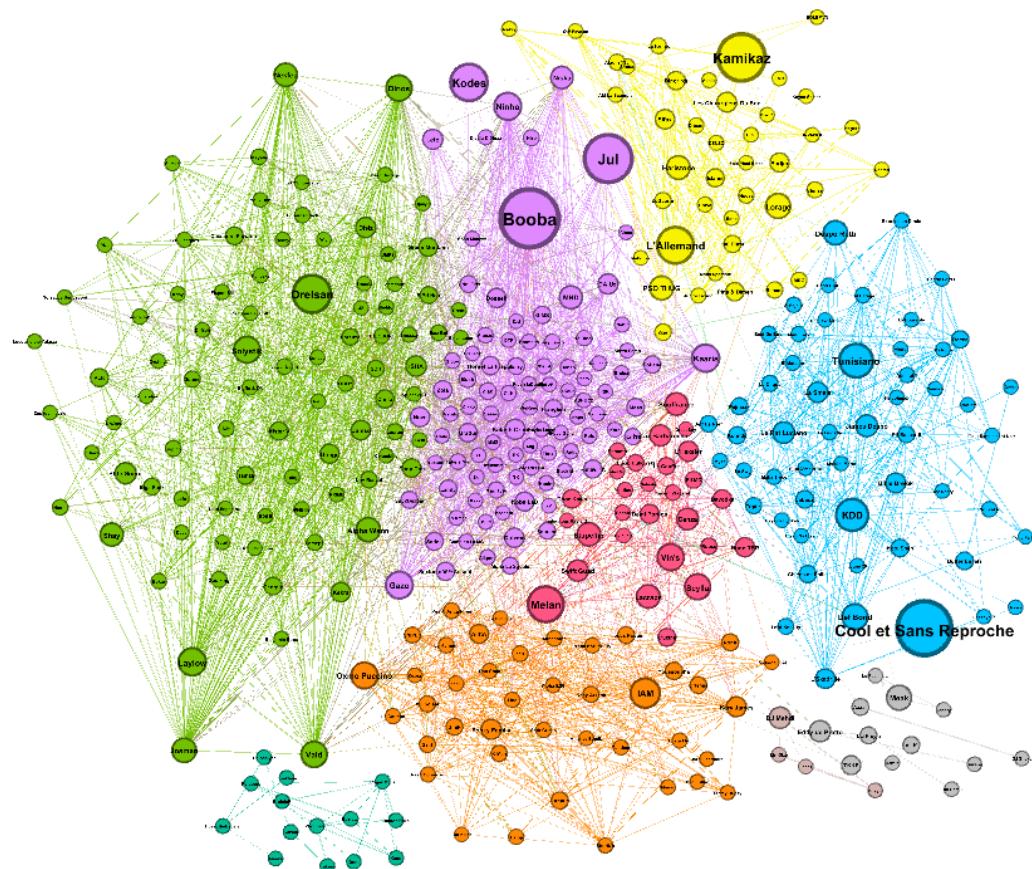


FIGURE 7 – Graphe de similarité Last.FM

Nous y retrouvons plus de clusters que dans notre premier graphe : les clusters **vert** et **violet** peuvent être identifiés respectivement comme des versions élargies de celui du rap conscient et celui du rap urbain que nous avons observé dans notre premier graphe. Le cluster **orange** est quand à lui assez proche de notre cluster "old school", tandis que les autres clusters sont assez difficiles à analyser : notre connaissance de ce genre musical nous permet difficilement d'y mettre un nom. Nous discuterons plus en détail de l'analyse de ce graphe dans la sous-section **4.3.**

3.3 GRAPHE DES EMBEDDINGS

Pour le graphe des embeddings, nous utilisons la stratégie abordée dans la sous section 2.4 : pour chaque couple d'artistes, insérer une arête entre eux si la similarité cosinus de leurs embeddings dépasse un seuil fixé, et donner à cette arête un poids proportionnel à la similarité cosinus. L'objectif de cette approche est que deux artistes soient connectés s'ils abordent des thématiques ou emploient un vocabulaire similaire dans leurs morceaux.

Lors du calcul des similarités cosinus, nous observons la distribution de similarités suivante :

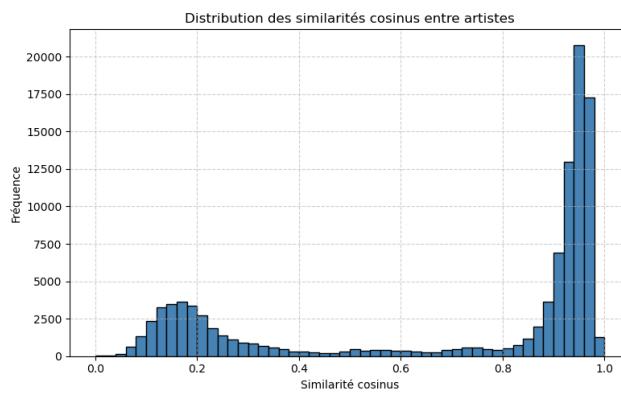


FIGURE 8 – Distribution des similarités cosinus entre artistes de notre base de données

La majorité des scores de similarité sont élevés, supérieurs à 0.9. Cela laisse présager qu'il est difficile de discriminer les artistes du rap francophone en se basant sur l'embedding que nous avons construit. Afin d'éviter d'avoir un graphe trop dense nous avons décidé d'utiliser un seuil élevé, pour ajouter seulement les arêtes en cas de très forte similarité. Nous avons augmenté progressivement le seuil jusqu'à la valeur 0.98, qui permet d'obtenir un graphe lisible et un nombre de clusters exploitable. Afin de conserver une hiérarchie des scores de similarité, nous avons appliqué l'opération suivante pour calculer le poids des arêtes :

$$\text{Poids} = \frac{\text{similarité} - \text{seuil}}{1 - \text{seuil}}$$

Cette opération normalise les scores de similarité supérieurs au seuil dans l'intervalle [0, 1], pour obtenir la distribution de poids d'arêtes disponible en [figure 9](#)

Comme pour les graphes précédents, nous procédons à un nettoyage du graphe : nous enlevons les composantes connexes de taille inférieure à 5, ce qui retire cette fois un grand nombre de sommets : on n'en conserve que 275, soit près de 40% des noeuds sont retirés. Le graphe obtenu est disponible en [figure 10](#).

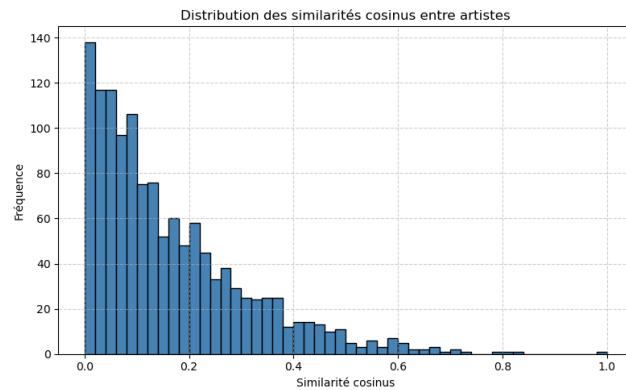


FIGURE 9 – Distribution de poids des arêtes dans le graphe d’embedding

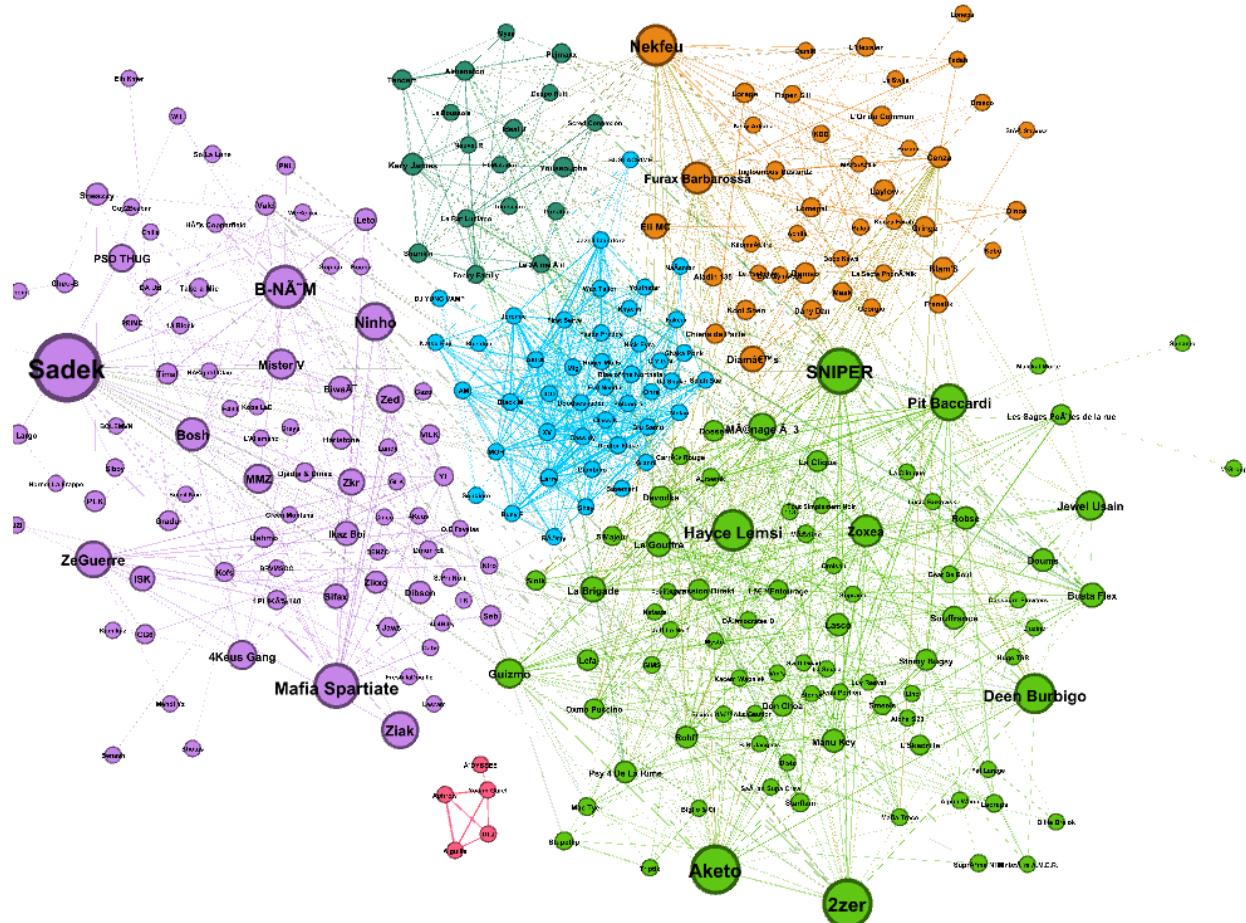


FIGURE 10 – Graphe des embeddings obtenu après coloration des clusters et attribution d'une-taille proportionnelle à la betweenness centrality des nœuds sur Gephi

4 INTERPRÉTATION DES RÉSULTATS

4.1 ANALYSE APPROFONDIE DU GRAPHE DE COLLABORATION ET DES COMMUNAUTÉS

Le graphe des collaboration obtenu est très satisfaisant et retranscrit bien les caractéristiques des artistes de chaque clusters. Les titres assignées aux différents clusters dans la partie précédente sont pertinents :

- Cluster **violet** : Parmi les multiples courants du rap français, le cluster violet correspond à un sous-genre spécifique, souvent qualifié de "rap mainstream" et dans une moindre mesure de "rap de rue", qui occupe une place centrale. Il est incarné par des artistes comme Booba, Koba LaD, Kaaris, Sofiane ou encore Leto.
- Cluster **orange** : Un autre courant majeur du rap français n'est autre que la scène marseillaise contemporaine, portée par des artistes comme Jul, SCH, Naps ou encore Alonzo. Ce sous-genre se distingue par une esthétique sonore solaire, rythmée et dansante, avec des influences variées.
- Cluster **bleu** : Ce cluster plutôt hétérogène correspond à un courant du rap français qui se concentre autour d'artistes comme Nekfeu, PLK, Alpha Wann, Vald, Laylow ou S.Pri Noir, qui incarnent une scène souvent qualifiée de rap conscient. Ce sous-genre se distingue par une grande diversité stylistique et une attention particulière portée aux textes.
- Cluster **vert** : Le rap incarné des artistes comme Swift Guad, Lacraps, Hugo TSR, Bigflo & Oli, Demi Portion et Lucio Bukowski pourrait être interprété comme les lyricistes portant une grande importance à l'écriture de leurs textes.
- Cluster **jaune** : scène qu'on pourrait appeler "rap old school" des années 90/2000, portée par Oxmo Puccino, Kery James, Alkpote, Seth Gueko ou encore Rim'k. Elle comporte également des artistes plus contemporains ayant beaucoup collaboré avec la scène des années 2010 et 2020 voire issus de cette même scène comme Orelsan, Gringe, Luv Resval ou encore B.B. Jacques (Révélation 2022 de Netflix : Nouvelle École).
- Cluster **rose** : Enfin le dernier cluster correspond à des artistes plus alternatifs qui auraient totalement eu leur place dans le cluster bleu mais qui ont été séparés par l'algorithme *louvain*, sûrement à cause une faible quantité de collaboration avec des artistes de d'autres clusters (à l'exception de So La Lune) et d'une très forte connection intra-cluster, notamment entre Lorenzo, Lujipeka et Columbine car les deux premiers faisaient parti du troisième qui est un collectif artistique rennais.

Bien que le clustering soit très satisfaisant, il existe tout de même une petite poignée d'artistes assignés à un cluster qui ne semble pas leur correspondre au mieux. C'est le cas du cluster rose, ou de Bigflo & Oli (cluster vert), qui ont beaucoup plus collaboré avec des artistes

conscients bien qu'ils rentrent tout à fait dans la case des lyricistes, ou encore de Luidji et Krisy (cluster violet), qui offrent un style plus doux et introspectif, caractéristique du "rap conscient" (cluster bleu). Cela semble être en partie dû à la nature du graphe qui est fortement connecté et au rap en lui même qui a pour valeur de casser les codes et donc les barrières entre sous genres. Il en résulte des barrières difficiles à définir qui participent à la richesse du rap français. C'en est même une marque de fabrique pour certains artistes qui mettent en avant leur style mélangé venant de plusieurs genres.

4.2 RETRANSCRIPTION DE L'ÉVOLUTION DU RAP À TRAVERS LE TEMPS

Quand on joue avec les différentes représentations du graphe, notamment en jouant avec la taille de nœuds proportionnelle soit à la betweenness centrality soit au degré, on observe deux dynamiques différentes pour les clusters jaune et violet, correspondant respectivement au "rap old school", plus ancien, et au "rap mainstream", plus jeune [figure 11].

L'affichage des nœuds selon le degré est assez révélateur de la forte augmentation du nombre de feats lors des 2 dernières décennies : les nœuds à gauche du cluster jaune ont un degré plus faible que ceux de droite, révélant un faible nombre de collaboration par artiste. Bien que ce ne soit pas très visible en zoomant sur ces clusters, la vue globale nous montre que les nœuds à gauche du cluster jaune ont en moyenne un degré plus faible que ceux du cluster violet. Les artistes à droite du cluster jaune correspondent à des artistes qualifiés de "old school" mais plus contemporains que ceux de gauche, car ils ont souvent collaboré avec des artistes émergents : Rim'k, Oxmo Puccino ou Orelsan, déjà évoqués en sont l'exemple.

Ce constat soutient notre affirmation d'introduction affirmant que les collaborations étaient devenues beaucoup plus fréquentes au cours des dernières décennies.

On remarque également que le degré est fortement corrélé à la centralité, ce qui est plutôt cohérent : un artiste ayant fortement collaboré devient un artiste important car il lie les différents sous genres. Des artistes comme Alonzo, Jul ou Leto en sont l'exemple. Cependant certains artistes arrivent à occuper une place centrale avec moins de feats. C'est le cas de Booba (dans le cluster violet) qui a la plus grosse betweenness centrality de son cluster mais qui a beaucoup moins de collaborations que Leto par exemple (cluster violet également) [figure 12].

Le graphe montre également l'évolution stylistique de certains artistes qui sont représentés par plusieurs nœuds. Non pas que le nettoyage des données ait échoué, certains artistes font ou ont fait partie de groupes de musique référencés dans différents clusters qu'eux. C'est le cas de Orelsan et Gringe dans le cluster jaune ("old school") alors que leur duo, les Casseurs Flowteurs, est dans le cluster bleu ("rap conscient"). Il en va de même pour Booba en cluster violet ("mainstream"), qui voit son duo avec Ali (artiste non représenté) représenté en cluster jaune ("old school").

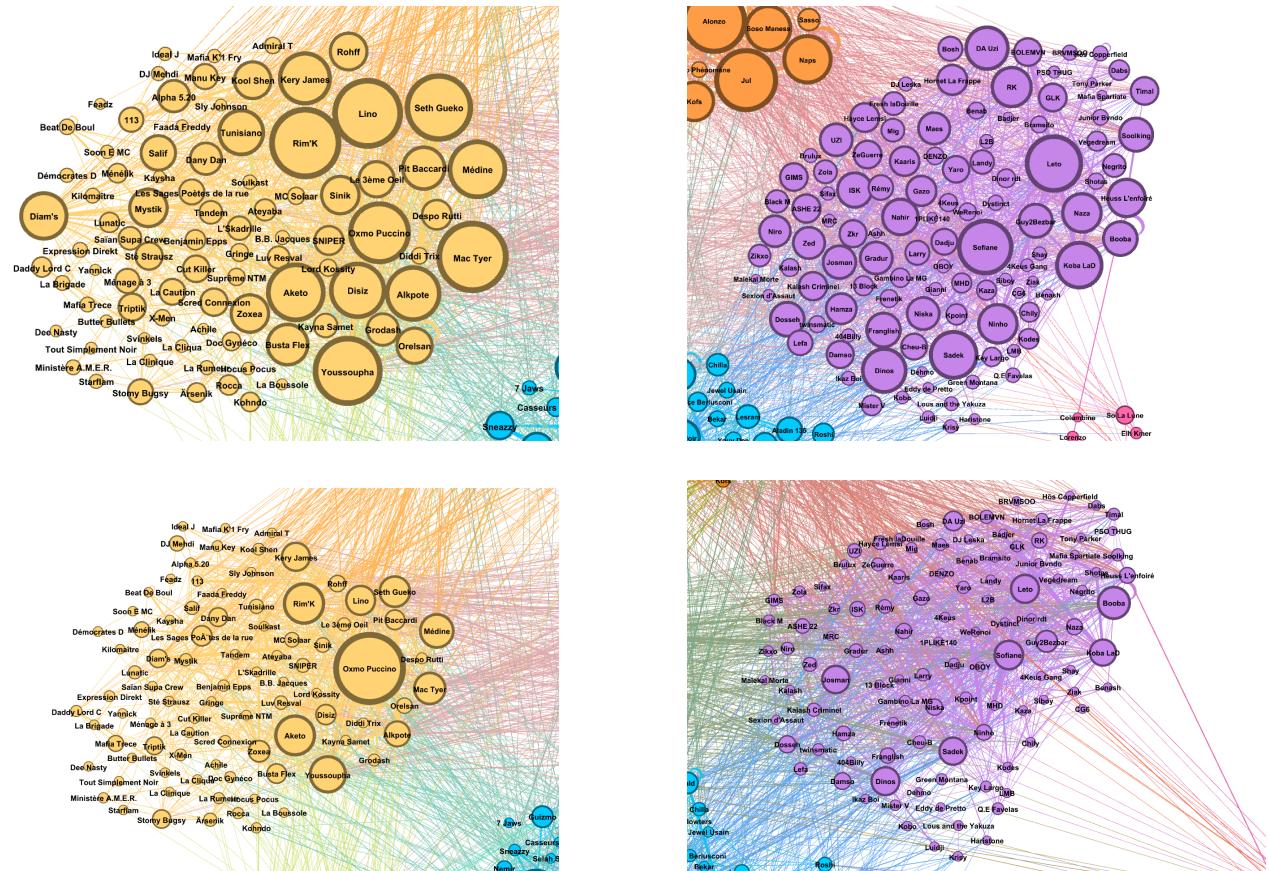


FIGURE 11 – Première ligne : taille proportionnelle au degré. Deuxième ligne : taille proportionnelle à la betweenness centrality. À gauche le "Rap old school" et à droite le "Rap mainstream". Les 4 captures d'écran ne sont pas à la même échelle.

4.3 MISE EN RELATION AVEC LES DEUX AUTRES GRAPHES

4.3.1 • GRAPHE DE SIMILARITÉ LAST.FM

Discutons désormais des clusters obtenus avec le graphe de similarité Last.FM. Comme expliqué en sous-section 3.2, on retrouve dans le graphe Last.FM des versions enrichies de nos clusters rap urbain et rap conscient. La frontière entre ces deux clusters est ici assez floue, avec de nombreux artistes de chacun des deux clusters que l'on pourrait légitimement classer dans l'autre. Les frontières floues entre les différents clusters sont probablement dues aux habitudes d'écoute des utilisateurs qui ne restreignent pas à un seul et unique genre. Ils ont tendance à écouter des artistes issus de différents sous-genres ce qui met moins en évidence les différents sous genres dans les données Last.FM.

Remarquons aussi que notre cluster "scène marseillaise" a totalement disparu dans le graphe Last.FM : les artistes qui le composent se retrouvent plus dispersés. Et pour cause, peu d'auditeurs n'écoutent que du rap marseillais ou pas du tout de rap marseillais. Si l'appartenance

name	Degré	Betweenness Centrality
Leto	80	0.020125
Sofiane	77	0.020656
Booba	45	0.02398

FIGURE 12 – Comparaison des statistiques de Booba et de Leto

à la scène marseillaise peut être identifiée via les collaborations avec les artistes proches, on ne peut pas la retrouver en se basant sur les goûts musicaux des auditeurs.

Un cluster du graphe Last.FM qui reste assez fidèle à celui du graphe des collaborations est celui de la scène old school. Puisqu'il s'agit d'artistes d'une autre génération, il est compréhensible qu'il touche une population d'auditeurs différente de celle des clusters précédents et donc de pouvoir reconstituer cette scène à partir des écoutes utilisateur.

Pour commenter les clusters dont le sens échappe à notre connaissance du milieu du rap francophone, nous pouvons remarquer que la plupart des artistes qui les composent sont peu populaires, et que l'algorithme Last.FM a donc probablement trop peu de données pour proposer une similarité cohérente. Cette hypothèse peut expliquer ce résultat décevant du graphe Last.FM.

Nous remarquons également que les noeuds de forte betweenness centrality, qui apparaissent en plus gros sur le graphe, sont différents de ceux de notre graphe des collaborations : ici, ce sont les artistes à la frontière entre plusieurs genres ou sous genres (comme Booba, Orelsan ou IAM qui touchent chacun plusieurs générations d'auditeurs) qui sortent du lot.

4.3.2 • GRAPHE DES EMBEDDINGS

Globalement, ce graphe donne des résultats assez décevants. On peut y retrouver des clusters assez proche de certains sous-genres du rap, mais ils sont très incomplets et tout de même assez hétérogènes. On retrouve par exemple une petite partie de la scène old school dans le cluster vert foncé, et une majorité d'artistes du rap conscient dans le cluster orange, mais cela reste très incomplet au vu de la richesse de ces sous-genres. Les autres clusters sont très hétérogènes, certains incluant même des artistes non rappeurs ou non francophones.

Deux grandes pistes d'améliorations sur cette approche peuvent être formulées :

- D'abord, notre méthode de construction de l'embedding de chaque rappeur est assez sommaire : elle ne prend en compte que 5 morceaux donc ne capture pas suffisamment la richesse lyrique de chaque artiste, et faire la moyenne d'embeddings de mots est assez destructeur d'information comme nous l'avons expliqué en section 2.4. Nous avons manqué de temps et de ressources de calcul pour extraire plus de paroles, et de temps de travail pour mettre en place une méthode plus élaborée que le Word2Vec.
- Nous aurions pu espérer de meilleurs résultats avec un algorithme de clustering appliquée directement sur l'espace d'embedding, sans passer par un graphe. Cette approche sortait selon nous du cadre de ce modal, et aurait été plus difficile à représenter graphiquement donc nous avons choisi de ne pas nous y plonger.

4.4 LA POPULARITÉ ET LE NOMBRE DE FOLLOWERS SPOTIFY CARACTÉRISENT-T-ILS AUSSI BIEN L'IMPORTANCE DES ARTISTES ?

Enfin, nous avons essayé d'utiliser les paramètres *Popularity* et *Followers*, tous deux issus des données de Spotify, afin de donner des tailles dépendant de ces paramètres aux noeuds. Cependant nous nous sommes rendus compte que ces données étaient trop similaires d'un noeud à un autre [figure 13, 1^{ère} ligne]. Nous avons alors utilisé une courbe de spline [figure 13, 2^{ème} ligne] afin de rendre ces données plus pertinentes [figure 13, 3^{ème} ligne]. Nous retrouvons grossièrement la dynamique des graphes de collaboration avec comme paramètres de taille le degré ou la centralité, mais avec une importance moindre donnée aux anciens rappeurs se trouvant dans les clusters vert, jaune et un peu dans le cluster orange. Cela pourrait s'expliquer par une forte utilisation des plateformes de streaming par les jeunes générations qui écoutent surtout des artistes récents, alors que ceux qui écoutent des rappeurs plus anciens sont également plus âgés, et les écoutent d'ailleurs souvent grâce aux exemplaires physiques des albums.

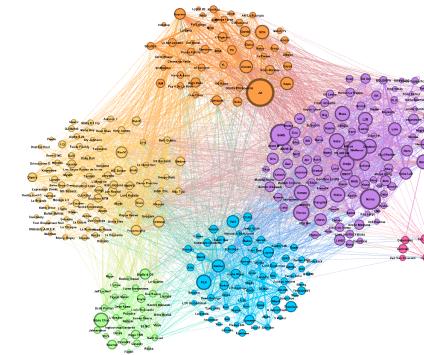
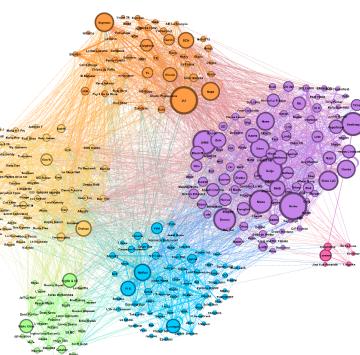
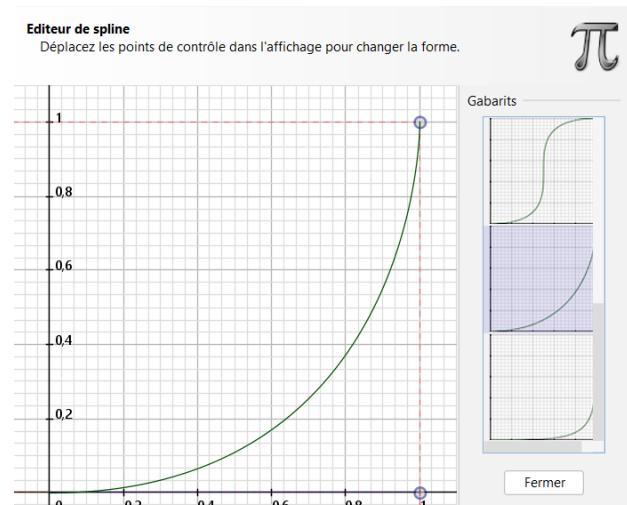
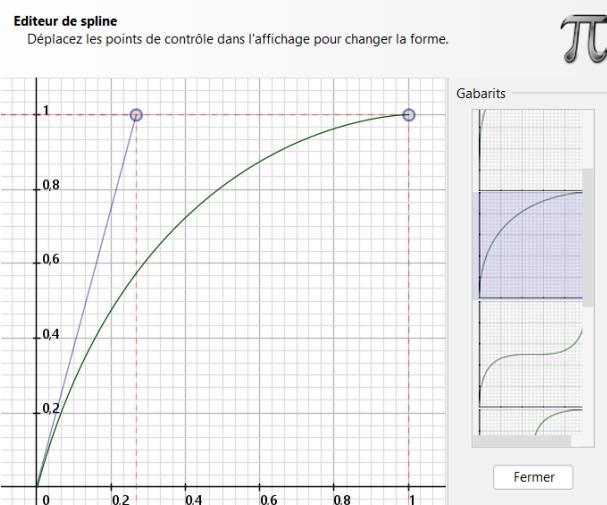
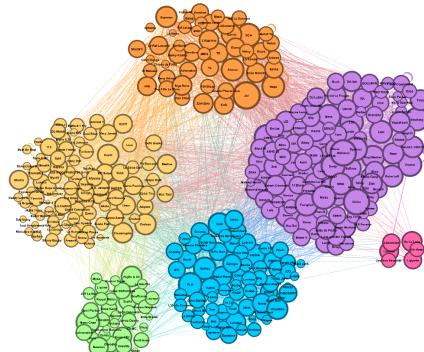
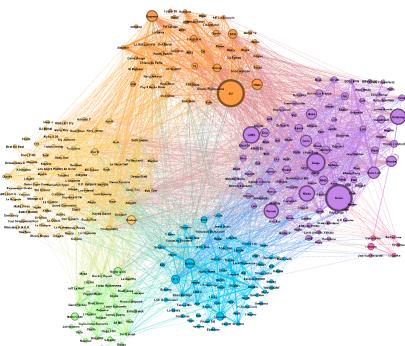


FIGURE 13 – À Gauche le paramètre est les *Followers* et à droite c'est la *Popularité Spotify*.
 1^{ère} ligne : tailles proportionnelles aux paramètres. 2^{ème} ligne : application d'une courbe de spline pour mieux représenter les données. 3^{ème} ligne : graphe avec cette courbe de spline

5 CONCLUSION

Au terme de ce projet, nous avons conçu et mis en œuvre plusieurs méthodes d'extraction, de restructuration et d'analyse de données issues de plusieurs sources, afin d'analyser en profondeur le réseau du rap francophone. Les graphes générés ont permis de visualiser des collaborations, d'identifier des communautés artistiques ainsi que de formuler des hypothèses sur les liens thématiques et stylistiques entre les artistes.

Ce projet a été doublement formateur pour nous. D'abord parce qu'il a été l'opportunité d'appliquer et de comprendre plus en détail les algorithmes étudiés en cours, mais aussi parce que nous avons accordé une place centrale à l'élaboration d'affichages graphiques clairs dans le terminal et d'une organisation soignée via *GitHub*, qui ont contribué au bon déroulement de ce projet collectif. En plus de notre expérience d'extraction et d'analyse de données, nous tirons de ce projet de nouvelles compétences de développement informatique, ce que nous nous étions fixé comme objectif en début de période.

Si nous avions eu plus de temps, nous aurions pu aller plus loin dans la collecte et l'analyse des données. Le bpm (battement par minute, tempo de la musique) des sons des artistes, le nombre de featuring par rapport au nombre de morceaux d'un artiste, les dates de sortie des titres (pour détecter les époques) ou encore les thèmes abordés (grâce à l'usage de NLP adaptés) sont autant de données intéressantes qui pourraient donner lieu à de nouveaux graphes et de nouvelles interprétations.

Nous aurions également pu inclure à nos données les producteurs musicaux, renseignés sur le site genius, ce qui nous aurait donné l'influence d'autres acteurs clés du domaine. Les labels et les majors, (difficilement) récupérables depuis l'API Spotify pourraient aussi nous donner d'autres liens entre artistes, car ils poussent souvent leurs artistes à collaborer ensemble. C'est particulièrement le cas du label *Rec 118* où y signés de nombreux rappeurs mainstream comme Ninho, Hamza ou SCH.

En guise d'ouverture, nous aurions apprécié élaborer un algorithme de recommandation en utilisant la structure du graphe (arêtes, centralité des noeuds...) pour recommander des artistes nouveaux à un utilisateur.

A ANNEXES

```
[graph.py] Graph stats:  
Number of nodes: 358  
Number of edges: 3955  
Average degree: 22.09  
Density: 0.0019  
Clustering coefficient: 0.4053  
Connected components: 1  
Is connected: True  
Average clustering coefficient: 0.4053  
[graph.py] -----  
[graph.py] Nodes stats:  
Top nodes by degree centrality: ['Alonzo', 'Jul', 'Leto', 'Soprano', 'Le Rat Luciano', 'Rimâ€™K', 'Sofiane', 'Akhenaton', 'Mac Tyer', 'Lino']  
[graph.py] -----  
[graph.py] Setting clusters using louvain...  
[graph.py] Cluster stats:  
Number of communities: 6  
Number of nodes in each community:  
Community 0: 110 nodes  
Community 1: 35 nodes  
Community 2: 88 nodes  
Community 3: 49 nodes  
Community 4: 6 nodes  
Community 5: 70 nodes  
[graph.py] -----  
[graph.py] Exporting graph to Gephi...  
[graph.py] Graph exported to ./graphs/graph_louvain_del_small_comp.gexf  
PS C:\Users\arthu\Desktop\Obsidian\Arthur Vault\3 - cours\x2\Modal\Modal_Projet> []
```

FIGURE 14 – Statistiques de notre graphe

artists				
Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
73.73 kB	477	225.00 B	1	20.48 kB
featurings				
Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
770.05 kB	17 K	148.00 B	1	397.31 kB

FIGURE 15 – Statistiques de notre base de donnée MongoDB

```

update.py > ...
290     def update_csv_to_mongo(db_name, filename = "./data/db_artists_clean.csv"):
291         """
292             This function updates the data in the MongoDB database.
293         """
294         print(f"[{this_name}] Ouverture de la base de données...")
295         client = MongoClient("mongodb://localhost:27017/")
296         db = client[db_name]
297         collection = db["artists"]
298
299         # Suppression de tous les documents existants
300         collection.delete_many({})
301         print(f"[{this_name}] Lecture du fichier csv...")
302         with open(filename, newline=None) as csvfile:
303             reader = csv.DictReader(csvfile)
304             for row in reader:
305                 collection.insert_one({
306                     "name": row['name'],
307                     "id_spotify": row['id_spotify'],
308                     "followers": int(row['followers']),
309                     "popularity": int(row['popularity']),
310                     "id_genius": int(row['id_genius']),
311                     "url_genius": row['url_genius'],
312                     "id_mb": row.get('id_mb')
313                 })
314         print(f"[{this_name}] Fermeture de la base de données MongoDB...{' *100}'")
315         client.close()

```

FIGURE 16 – Fonction d'extraction des données depuis un fichier csv en une base MongoDB dans le fichier *update.py*

RÉFÉRENCES

- [1] [Genius website](#)
- [2] [Word2Bezbar](#) : Word2Vec Models for French Rap Lyrics
- [3] Lingfei Wu, Ian En-Hsu Yen, Kun Xu, Fangli Xu, Avinash Balakrishnan, Pin-Yu Chen, Pradeep Ravikumar, and Michael J. Witbrock. 2018. [Word Mover’s Embedding: From Word2Vec to Document Embedding](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pages 4524–4534
- [4] [WatiBERT](#) : Fine-Tuned BERT Model for French Rap Lyrics, rapminerz