Rapport final IFT 3150

Moteur de recherche de musique s’appuyant sur les émotions

Table des matières

Contexte et Problématique

La recherche de morceaux de musique sur des plateformes comme YouTube et Spotify repose sur l'utilisation de mots-clés. Bien qu'efficaces lorsque le titre ou l’auteur sont connus à l'avance, les mots-clés rendent difficile la recherche fondée sur d’autres caractéristiques telles que les émotions. Non seulement les émotions sont difficiles à traduire en mots, mais, en plus, les pièces musicales sont actuellement rarement associées à des émotions.

Ce projet vise à développer une nouvelle forme de moteur de recherche fondée sur l'exploration visuelle de l'espace émotionnel des morceaux de musique.

Les objectifs sont de développer une représentation visuelle d'un espace émotionnel et un algorithme capable de classer des morceaux de musique dans cet espace et, ensuite, concevoir un système de recherche visuelle basé sur cet espace, système qui aidera les utilisateurs à chercher et à découvrir des morceaux de musique.

État des lieux de la recherche de musique

La meilleure recherche émotionnelle de musique actuellement disponible de façon commerciale est celle de Spotify. Celle-ci s'appuie sur le moteur d'extension de playlist de la plateforme : quand l'utilisateur recherche un mot clé émotionnel que Spotify considère comme exploitable, il génère ce que l'interface appelle un *Mix*. Cette playlist est générée à la volée en utilisant comme racine un petit nombre de morceaux de la bibliothèque de l'utilisateur, dont les caractéristiques émotionnelles correspondent exactement à l'adjectif recherché. Ensuite, en utilisant le moteur d'extension de playlist, on rajoute des morceaux jusqu'à atteindre une taille satisfaisante. Bien que très rapide, cette méthode n'est pas idéale car elle se contente d'utiliser et de construire sur ce que l'utilisateur a déjà écouté, et ne s'appuie pas de façon effective sur l'état émotionnel recherché.

Les autres plateformes comme Tidal, Youtube, Apple Music, etc. se contentent de comparer les termes de recherches aux titres, noms d'artistes ou paroles des chansons, ce qui est encore plus éloigné de ce que nous cherchons.

De la recherche sur la visualisation et l'exploration de bibliothèque musicale à déjà été faite. Par exemple, Park et al. (2024)[1] nous montrent les limites de la recherche de musique s'appuyant uniquement sur le texte. On peut aussi mentionner Pohle et al. (2007)[2] qui proposent d'organiser les morceaux d'une collection par genres sur un gradient linéaire et ensuite enrouler ce gradient dans un cercle explorable avec un bouton rotatif, compatible avec l'interface des premiers IPod. Knees et al. (2007)[3] proposaient de parcourir une librairie musicale en la visualisant sous la forme d'un archipel, transposé dans un monde 3D que l'utilisateur pouvait parcourir à l'aide d'une manette de console de jeu vidéo.

---

[1] Park, J., Shin, H., Oh, C., & Ha Young Kim. (2024). *"Is Text-Based Music Search Enough to Satisfy Your Needs?" A New Way to Discover Music with Images*. https://doi.org/10.1145/3613904.3642126

[2] Pohle, T., Knees, P., Schedl, M., Pampalk, E., & Widmer, G. (2007). "Reinventing the Wheel": A Novel Approach to Music Player Interfaces. *IEEE Transactions on Multimedia*, *9*(3), 567–575. https://doi.org/10.1109/tmm.2006.887991

[3] Knees, P., Schedl, M., Pohle, T., & Widmer, G. (2007). Exploring Music Collections in Virtual Landscapes. *IEEE Multimedia*, *14*(3), 46–54. https://doi.org/10.1109/mmul.2007.48

Travail réalisé

Étude de l'espace émotionnel

Constitution du dataset

Pour pouvoir travailler sur ce projet, il était évidemment impossible de simplement extraire ma librairie personnelle de musique. Aussi tentant que cela aurait été, utiliser uniquement des morceaux de musiques que j'appréciais aurait introduit dès le début un bien trop grand biais.

Il a donc été décidé de faire appel à un *dataset* libre de morceaux de musique. Après quelques recherches, je me suis tourné vers la *Free Music Archive*[4,5,6], qui est un jeu de morceaux sous licence CC[7] initialement conçu pour l'apprentissage machine, mais qui présentait, par rapport à d'autre *datasets,* l'avantage significatif de proposer des morceaux complets. Malheureusement, lesdits morceaux étaient conservés dans un fichier zip de 879 Go, ce qui le rendait impossible à décompresser sur ma machiner personnelle. Il a cependant été possible, grâce à l'aide d'un des administrateurs système du département[8], de procéder à la décompression sur les serveurs du département, dans un stockage distant, avec la possibilité d'y accéder par SSH.

Un *script bash* a ensuite été mis au point dans le but de créer un sous-*dataset* à partir des morceaux de FMA. Celui-ci était déjà subdivisé en plusieurs blocs (156) contenant chacun des morceaux du *dataset*.

Afin de garantir l'équilibre entre les morceaux avec paroles et sans paroles lors de la construction du nouveau *dataset*, je me suis appuyé sur des approches décrites dans la section suivante.

Paroles et finalisation du dataset

L'obtention des paroles et la garantie de l'équilibre entre les morceaux sans paroles et avec paroles s'est révélée être un des obstacles les plus complexes et chronophages à surmonter durant ce projet de recherche.

Initialement, l'objectif était simplement de déterminer si une jointure entre FMA et une base de données de paroles de chansons libres était possible. D'un point de vue théorique, il suffisait de comparer l'index des deux bases et de prendre en

---

[4] Defferrard, M., Benzi, K., Vandergheynst, P., & Bresson, X. (2017). FMA: A Dataset For Music Analysis. *arXiv:1612.01840 [Cs]*. https://arxiv.org/abs/1612.01840
[5] Free Music Archive sera abrévié en "FMA"
[6] https://github.com/mdeff/fma
[7] Creative Commons
[8] Remerciements à Raouf Bencheraiet pour son aide précieuse.

note les lignes qui correspondaient. Cette tâche s'est révélée être significativement plus complexe dans la pratique, et ce pour plusieurs raisons.

Dans un premier temps, il faut donner du contexte à cette jointure. L'IFPI[9] dénote sur son site l'existence de plus de 150 millions[10] d'ISRC[11]. De tous les datasets que j'ai exploré, le plus grand (LRCLIB[12,13]) contenait entre 3 et 5 millions d'entrées, en fonction de la stratégie de déduplication choisie. Concrètement, tous les des *datasets* libres que j'ai pu trouver se sont révélés trop petits.

L'aspect chronophage de la question venait du fait que pour vérifier l'hypothèse du manque de couverture, il a fallu exécuter des requêtes SQL entre les index des bases de paroles et FMA. Les bases de données étaient mal indexées et s'exécutaient sur un disque externe connecté par USB à cause de leur taille, menant à des temps d'exécutions dépassant les 15 minutes pour des jointure relativement simples.

Dans le cas du FMA, l'index n'existait tout simplement pas au format SQLite, nécessitant donc de le créer à partir de l'index CSV grâce à la librairie python sqlite-tools[14].

Le logiciel DataGrip de JetBrains a permis de rendre la gestion de ces longues requêtes plus simple, grâce à la possibilité de faire fonctionner plusieurs SGBDs en parallèle de façon isolée. Les logiciels open-source Xan[15], Visidata[16] et Tablecruncher[17] ont aidé dans la manipulation des fichiers CSV de très grande taille.

Compte tenu de l'insolvabilité du problème, il a été nécessaire de demander, par l'intermédiaire de M. Lafontant, du financement pour obtenir une clé pour l'API[18]

---

[9] Fédération internationale de l'industrie phonographique
[10] https://isrc.ifpi.org/
[11] L'ISRC (International Standard Recording Code) est un identifiant unique pour chaque morceau de musique enregistré auprès d'un label ou d'une maison de disque et qui permet la gestion internationale des droits d'auteurs. C'est le seul standard de numérotation des morceaux qui soit complètement agnostique en matière de plateforme et qui puisse être considéré comme universel. Bien que sa couverture s'arrête aux morceaux couverts par les droits d'auteurs via des labels ou des maisons de disques (excluant les morceaux en CC0 ou auto-financés), il fournit une très bonne estimation du nombre de morceaux en circulation dans le monde à un instant donné.
[12] tranxuanthang. (2026). *GitHub - tranxuanthang/lrclib: LRCLIB server written in Rust with Axum and SQLite3 database*. GitHub. https://github.com/tranxuanthang/lrclib
[13] LRCLIB. (2026). In *Lrclib.net*. https://lrclib.net/
[14] simonw. (2026). *GitHub - simonw/sqlite-utils: Python CLI utility and library for manipulating SQLite databases*. GitHub. https://github.com/simonw/sqlite-utils
[15] https://github.com/medialab/xan
[16] https://www.visidata.org/
[17] https://tablecruncher.com/
[18] https://docs.musixmatch.com/getting-started

de la compagnie bolonaise Musixmatch[19] , qui sert déjà de *backend* à un grand nombre de très gros acteurs de l'industrie de la musique, dont Spotify, Tidal, Amazon et Apple Music[20].

Cette clé API était liée au plan le moins cher (et néanmoins onéreux) de Musixmatch, qui est très restreint vis-à-vis du nombre d'appels, notamment ceux à l'*endpoint* pour les paroles, pour lequel le besoin était le plus grand.

Il a donc fallu procéder à la création d'un algorithme optimisé qui permettrait de créer mon sous-*dataset*[21] à partir des informations de cette API. Le principe était d'utiliser un appel de recherche dans la base (dont je disposais d'une quantité généreuse) et d'exploiter ensuite les informations auxiliaires de cet appel, notamment une certaine valeur booléenne qui permettait de déterminer si le morceau était instrumental ou non. A partir de cet appel il était aussi possible de récupérer le commontrack_id, qui est l'identificateur du morceau à l'intérieur du système de référencement de Musixmatch. Il ne doit pas être confondu avec le Spotify ID, l'Apple Music ID, ou l'ISRC précédemment mentionné.

Si nous avions la garantie que le morceau contenait des paroles, il suffisait alors de déterminer si son inclusion dans le bloc actuel était pertinente et procéder, si nécessaire, à la récupération des paroles *via* l'*endpoint* dédié.

```
Pour chaque morceau d'un bloc:
        le morceau a-t-il des paroles ?
                si oui :
                        As-t-on déjà 5 morceaux avec paroles pour ce bloc?
                                si non :
                                        ajouter le morceau à micro-FMA
                                si oui :
                                        ignorer le morceau
                si non :
                        As-t-on déjà 5 morceaux sans paroles pour ce bloc?
                                si non :
                                        ajouter le morceau à micro-FMA
                                si oui :
                                        ignorer le morceau
Passer au bloc suivant
```

Une fois MicroFMA construit avec ses 1560 morceaux et les paroles récupérées, il a été possible de passer à l'étape suivante de construction de ma base de vecteurs émotionnels.

Construction de la base de vecteurs

Équipés des données collectées durant l'étape précédente, il était à présent temps de créer nos vecteurs. La stratégie choisie consistait à construire des

[19] https://www.musixmatch.com/
[20] https://about.musixmatch.com/business/customer-stories
[21] Baptisé pour l'occasion "MicroFMA"

macro-vecteurs à partir des sous-vecteurs sémantiquement distincts. Trois pipelines de construction parallèles ont été utilisés.

Métadonnées

L'index principal de FMA, et par extension de MicroFMA, regorgeait de métadonnées très utiles pour les morceaux contenus. On pouvait y trouver des informations sur l'année, le genre, la position géographique ou le morceau avait été créé, en plus d'informations plus conventionnelles comme son titre, le nom de l'album, ou le nom de l'artiste. L'agrégation de ces métadonnées a formé le premier des trois sous-vecteurs de notre macro-vecteur, pour chaque morceau du *dataset*.

Vectorisation des morceaux Music2Emotion

Music2Emotion[22,23] est un modèle de *machine learning* permettant d'obtenir à partir d'un morceau de musique une série de coefficients allant de 0 à 1, pour une série de 56 propriétés émotionnelles prédéfinies[24]. Music2Emotion était aussi en mesure de fournir, en même temps que les coefficients émotionnels, la *valence* et l'*arousal*[25] du morceau.

La principale difficulté dans l'exploitation de ce modèle venait du fait qu'il utilise nécessairement une vieille version de python (3.12), L'opération de *build* (qui utilisait Meson) échouait si la version de *setuptools* était supérieure à 82, à cause de l'emploi de fonctions obsolètes.

Le traitement des morceaux a pris environ une demi-journée sur un système avec un GPU Nvidia RTX5070 12GB et 32GB de RAM.
Deux morceaux n'ont pas pu être traités à cause d'une erreur de nature inconnue entre le codec MPEG MP3 et la section du kernel CUDA chargé de l'ingestion des morceaux.

Traitement des paroles

Pour les paroles, la façon la plus logique d'en stocker vectoriellement le sens était de générer un *embedding*. « Les modèles d'embedding sont des réseaux de

---

[22] Kang, J., & Herremans, D. (2025). Towards Unified Music Emotion Recognition across Dimensional and Categorical Models. In *arXiv.org*. https://arxiv.org/abs/2502.03979
[23] Music2Emotion sera dorénavant abbrevié en M2MO
[24] Liste (en langue originale): *action, adventure, advertising, background, ballad, calm, children, christmas, commercial, cool, corporate, dark, deep, documentary, drama, dramatic, dream, emotional, energetic, epic, fast, film, fun, funny, game, groovy, happy, heavy, holiday, hopeful, inspiring, love, meditative, melancholic, melodic, motivational, movie, nature, party, positive, powerful, relaxing, retro, romantic, sad, sexy, slow, soft, soundscape, space, sport, summer, trailer, travel, upbeat, uplifting*.
[25] La *valence* et l'*arousal* sont les deux coordonnées d'un système orthonormé de représentation des émotions : Le Système Circumplex de l'Affect, théorisé par James Russell (*cf* note 35 ci-dessous). Ce sont des valeurs quantifiées qui ne sont pas calculées algorithmiquement par analyse de fréquence mais par des modèles d'apprentissage machine spécialisés.

neurones basés sur des *transformers*, qui transforment les morceaux de documents (c'est-à-dire les passages de texte) en représentation numérique ou vecteur. Les contenus ayant une signification similaire seront mappés à une représentation similaire dans l'espace latent [...] »[26] (IBM, 2024).

Pour ce projet, il a été décidé de se porter sur le modèle Qwen3-embedding-0.6B[27] tel que mis à disposition par la librairie Transformers de la plateforme HuggingFace[28]. La raison de ce choix était la capacité de ce modèle à générer des *embeddings* de dimensionnalité arbitraire, ce qui a permis a la base de données de garder une taille raisonnable.

Pour cette application spécifique, il a été décidé de paramétrer le modèle pour générer des *embeddings* à 256 dimensions, car cette valeur semblait offrir un bon compromis entre compacité et distinctivité.

Agglomération

Tous ces sous-vecteurs sémantiques ont été agglomérés dans une base de données sous la forme d'un CSV monolithique. Il a été argumenté qu'il était possible de placer les vecteurs dans une base SQLite, mais le gain en rapidité marginal (considérant le nombre de morceaux) et l'augmentation de la complexité d'exploitation ont rapidement contribué à exclure cette option. Le format CSV est simple, universellement reconnu par l'essentiel des outils et langages et parfaitement capable de supporter la charge de lecture-écriture. Charge d'autant plus petite que l'entièreté du fichier allait pouvoir être chargé en RAM sans passer par un *pagefile*, sa taille ne dépassant pas les 6 MB.

Visualisation

Pour mieux comprendre les données qui venaient d'être produites, il était important de produire un certain nombre de visualisations. Le problème principal venant du fait qu'il fallait projeter un espace à plusieurs centaines de dimensions vers un espace bidimensionnel.

---

[26] *Qu'est-ce qu'un embedding ? Un guide des modèles d'embedding textuel | IBM*. (2026, March 19). Ibm.Com. https://www.ibm.com/fr-fr/think/architectures/rag-cookbook/embedding

[27] Zhang, Y., Li, M., Long, D., Zhang, X., Lin, H., Yang, B., Xie, P., Yang, A., Liu, D., Lin, J., Huang, F., & Zhou, J. (2025). Qwen3 Embedding: Advancing Text Embedding and Reranking Through Foundation Models. In *arXiv.org*. https://arxiv.org/abs/2506.05176

[28] https://huggingface.co/Qwen/Qwen3-Embedding-0.6B

En s'inspirant du travail de Chevalier et al.[29] , il a été établi que la meilleure stratégie serait de travailler avec plusieurs vues. En IHM[30], une vue est une façon spécifique de représenter des données. En groupant plusieurs vues différentes, on offre à l'utilisateur la possibilité d'étudier les données selon plusieurs perspectives, d'une façon similaire à ce qui est proposé par Elmqvist et al. (2008)[31]

Distances relatives dans un graphe sous contrainte

Considérant que l'objectif de ce projet de recherche était de proposer une façon d'explorer l'espace émotionnel en musique, il semblait intéressant d'étudier comment nos vecteurs musicaux étaient répartis les uns par rapport aux autres dans cet espace espace.

Pour ce faire, la proximité de chaque vecteur l'un par rapport à l'autre a été analysée et stockée dans une matrice, qui a servi d'entrée à un moteur de graphe sous contrainte. De la sorte, la distance relative entre chaque morceau était traduite par la distance relative entre les points correspondants sur la visualisation bidimensionnelle.

Initialement, l'erreur a été commise de travailler avec la distance euclidienne, mais celle-ci fonctionne moins bien quand on travaille avec des vecteurs à dimensionnalité élevée. On a donc effectué une transition vers la similarité cosinus[32], qui a produit une structure très intéressante. Cette structure a été baptisé « *motif* en cirrus » à cause de sa ressemblance avec le nuage d'été du même nom. Un détail très intéressant sur cette structure est qu'elle persiste peu importe la manière dont on retravaille la méthode avec laquelle la similarité cosinus est calculée. En effet, l'approche initiale de calcul, impliquant de travailler sur les vecteurs bruts, s'est rapidement révélée comme limitée et limitante. On s'est donc tourné, après quelques essais, vers la similarité cosinus masquée en bloc[33]. Les blocs proviennent du fait qu'au lieu de traiter les macro-vecteurs comme des entités monolithiques, la similarité était calculée entre les sous-vecteurs sémantiques. Quand on compare un morceau avec paroles avec un morceau qui n'en a pas, on applique un masque sur l'*embedding* des paroles. Ceci afin d'éviter de devoir artificiellement le mettre à zéro i, ce qui aurait pu mener à la création d'un ou plusieurs faux *clusters* à proximité de l'origine.

---

[29] Chevalier, F., Dragicevic, P., & Hurter, C. (2012). Histomages. *Proceedings of the 25Th Annual ACM Symposium on User Interface Software and Technology*, 281–286. https://doi.org/10.1145/2380116.2380152

[30] IHM : "Interactions Humain-Machine"

[31] Elmqvist, N., Dragicevic, P., & Fekete, J.-D. (2008). Rolling the Dice: Multidimensional Visual Exploration using Scatterplot Matrix Navigation. IEEE Transactions on Visualization and Computer Graphics, 14(6), 1141-1148. https://doi.org/10.1109/tvcg.2008.153

[32] *Cosine similarity* en anglais

[33] *Masked Blocked Cosine Similarity* en anglais

Vue « géographique »

La vue géographique est sans doute la plus simple, puisqu'elle se contente de projeter sur une carte les coordonnées géographiques de chaque morceau, tel que présentes dans les métadonnées du FMA. Cette vue à permis de découvrir que certains morceaux présentaient des erreurs de métadonnées, prétendument enregistrés à *Null Island*[34].

Vue projetée avec UMAP

Un des désavantages de la vue en graphe de proximité est que les points représentant les morceaux sont projetés dans un espace non orthonormé, ce qui signifie que nous manquons d'information sur la position absolue des morceaux. Il ne faut cependant pas oublier que nous travaillons dans un espace hyperdimensionnel et qu'il n'est donc pas possible de simplement afficher ces points sur un écran.

Plusieurs techniques existent pour projeter ce genre d'espace vers un espace de dimensionalité plus faible. On peut mentionner PCA, T-SNE et UMAP. De celles-ci, celle qui a été la plus recommandée par la direction de recherche est UMAP.

On a donc utilisé UMAP pour convertir nos informations hyperdimensionnelles en nuage de points dans un espace bidimensionnel, ce qui a produit une nouvelle vue.

Nuage de points Valence *vs* Arousal

La *Valence* et l'*Arousal* ont été mentionnées brièvement dans les sections precedentes. Ces deux métriques appartiennent au Modèle Circumplex de l'Affect théorisé pour la première fois par J.A Russell en 1980 (Russell,1980)[35]. Vidas et al. (2025)[36] indiquent une bonne fiabilité de ces métriques en musique, telles que calculées par Spotify. Bien qu'on n'ait pas accès à l'implémentation exacte de Spotify, il est acceptable de présumer d'un niveau de fiabilité similaire pour les implémentations disponibles au grand public[37], ou, dans notre cas, dans M2MO.

---

[34] *Null Island* est le nom donné au point du globe de coordonnées 0° N, 0° E.

[35] Russell, J. A. (1980). A Circumplex Model of Affect. *Journal of Personality and Social Psychology*, *39*(6), 1161–1178. https://www.researchgate.net/publication/235361517_A_Circumplex_Model_of_Affect

[36] Vidas, D., Nitschinsk, L., Osborne, M., & Rickard, N. (2025). *Validating Spotify's 'Valence,' 'Energy,' and 'Danceability' Audio Features for Music Psychology Research*. https://doi.org/10.31234/osf.io/8gfzw_v2

[37] Plusieurs extracteurs libres existent. Pour ce projet, il a été envisage d'utiliser l'extracteur de la librairie Essentia (https://essentia.upf.edu/) dont une des demonstrations en ligne est n porjet qui rappelle certains aspects du notre.

La *valence* est perçue comme l'axe représentant la négativité ou la positivité d'une émotion, alors que *l'arousal* en représente l'intensité (ce dernier a été renommé en *Energy* par Spotify).

En traçant un nuage de points avec ces deux axes servant de base orthonormée à notre graphe, on se retrouve avec une figure présentant une apparence très diagonale, ce qui semble être corroboré par certains des résultats de Nandy et al. (2021)[38]. Les morceaux les plus intenses sont ceux qui ont le plus tendance à exprimer des émotions fortement négatives ou positives.

Interactivité

Toutes les représentations, projections et visualisations susmentionnées ont été groupées dans une interface commune en y ajoutant deux fonctionnalités.

La première consiste en l'ajout d'une mécanique impliquant le fait qu'effectuer une sélection dans une des vues reproduit automatiquement cette sélection de morceaux dans les autres vues, leur offrant donc une forme de synchronisation.

La seconde est que, sur pression d'un bouton, l'utilisateur peut donner à l'interface l'autorisation de jouer les morceaux sur lequels il déplace sa souris, permettant ainsi de mieux saisir le contexte émotionnel dans lequel il se trouve.

Axes sémantiques dynamiques

A partir de notre base de vecteurs, contenant des *embeddings* pour les paroles de chaque morceau qui en possède, l'idée est apparue de proposer une visualisation qui permettrait de comparer la distance entre chaque morceau et des concepts choisis par l'utilisateur, et transformés en *embeddings* à la volée.

On a, pour ce faire, utilisé l'implémentation en javascript de la librairie Transformers de la plateforme HuggingFace qui héberge Qwen3-Embeddings-0.6b. La visualisation calcule *l'embedding* des concepts soumis par l'utilisateur et positionne les points de chaque morceau en fonction de la similarité cosinus entre l'*embedding* des paroles et celui de l'axe concerné. On obtient de cette façon les coordonnées de chaque point.

Malheureusement, les résultats de cette visualisation n'ont pas été à la hauteur des attentes. Celle-ci semblait fonctionner (*e.g.* les concepts de « *war* » et « *ambition* » sont plus proches l'un de l'autre que « *love* » et « *sadness* »[39]), mais

[38] Nandy, R., Nandy, K., & Walters, S. T. (2021). Relationship between valence and arousal for subjective experience in a real-life setting for supportive housing residents (Preprint). JMIR Formative Research. https://doi.org/10.2196/34989

[39] À noter que bien que les concepts ici mentionnés soient en anglais, la documentation de Qwen3-Embedding-0.6b indique clairement que le modèle accepte plus de 100 langues.

elle ne nous a pas apporté de nouveau concepts concernant les données sur lesquelles nous travaillions.

Interface

En Interaction Humain-Machine, il est préférable de passer par une analogie avec la vie réelle, que l'on appelle métaphore, afin de ne pas avoir à faire apprendre à l'utilisateur de nouveaux concepts abstraits, sans en avoir absolument le besoin. De façon générale, une métaphore est une transposition d'un concept d'un domaine source familier vers un domaine de destination.
Une métaphore que nous utilisons au quotidien est la métaphore dite « du bureau ». C'est elle qui est à l'origine de l'organisation familière de nos ordinateurs avec des dossiers, contenant des fichiers, que l'on peut ouvrir sur un bureau (appelé *desktop* en anglais).

Au lieu d'essayer d'enseigner un nouveau concept à l'utilisateur (en se souvenant qu'un tutoriel est un aveu d'échec), on s'appuie sur un concept connu et sémantiquement intégré. C'est un peu comme exploiter la librairie standard d'un langage de programmation, mais dans ce cas nous utilisons « la librairie standard » du cerveau humain, acquise au cours de son expérience de vie.

La raison pour laquelle trouver une métaphore pour représenter nos données est si difficile est somme toute assez simple : étant sortis du domaine classique de la musique, nous ne pouvons plus nous appuyer sur ces expériences communes chères à tous les designers d'interface. Il n'y a pas d'autre exemple d'interface physique ou virtuelle réalisant une action similaire pour l'utilisateur, car il n'a jamais eu à réaliser une telle action. Une autre difficulté est que nous travaillons dans des espaces à plusieurs centaines de dimensions. En tenant compte du fait que les êtres humains ont du mal à travailler avec des représentations de dimensionnalité supérieure à deux, on est obligé de compresser l'essentiel de nos représentations sur un plan. De la même façon, on est restreint à un maximum de 5 couleurs, 2 formes si c'est absolument nécessaire et le moins de variations de couleur possible.

C'est dans ce contexte que nous nous sommes tournés vers la métaphore du disquaire. En effet, les bacs d'un disquaire permettent de bien retranscrire la proximité des morceaux, et des morceaux mal replacés par d'autres clients serviraient en réalité de portail pour se rendre dans un bac (donc ici un *cluster*) à proximité immédiate.

Pour déterminer nos clusters, nous nous sommes arrêtés sur la méthode des *K-means*. Initialement, la *elbow method* a été utilisé pour déterminer le nombre idéal de clusters. Mais on obtenait cinq *clusters* de plusieurs centaines de morceaux, ce qui ne faisait pas de sens, même dans les cas les plus extrêmes de

notre métaphore. On a donc pris le problème à rebours et imposé une composition de *clusters* avec un maximum de 50 morceaux par bac, ce qui correspond à peu près au maximum qu'on pourrait rencontrer dans la vie réelle.

L'interface présente ces bacs sur une rangée dans laquelle l'utilisateur navigue en scrollant de façon latérale. Pour fureter dans un bac, il lui suffit de scroller dessus dans le sens vertical. Passer sa souris sur un morceau, affiche une jaquette générée à la volée, présentant sur fond gris le titre, le nom de l'album et l'identifiant du morceau dans la librairie FMA (essentiellement utilisé à des fins de débogage). Si la souris reste plus de 500 ms sur un même morceau, et que l'option correspondante est activée, le morceau commence à se jouer.

Les morceaux verts représentent les morceaux « mal replacés » (leur couleur verte les rend plus faciles à identifier dans cette version prototype de l'interface) et suivent les mêmes règles dans tous les bacs : il y en a toujours deux, ils sont placés aléatoirement dans le bac et ils renvoient toujours vers les bacs correspondant aux deux clusters les plus proches du bac de départ.

Différences par rapport aux objectifs initiaux et perspectives

Différences par rapport aux objectifs

Une grande partie des objectifs initiaux de ce projet de recherche ont pu être accomplis, mais, et ce notamment a cause de contraintes de temps et de moyens, pas toujours avec le niveau de profondeur désiré.

Concernant l'espace émotionnel, il aurait probablement été bénéfique de travailler sur un plus grand nombre de morceaux. Avec n=1560 morceaux, il est possible et même probable qu'un certain nombre de comportements à grande échelle nous ait échappé.

Avec plus de temps, il aurait aussi été possible de tester plusieurs compositions de macro-vecteurs, et plusieurs stratégies de générations de sous-vecteurs, typiquement en variant les modèles utilisés ou le nombre de dimensions de leur sortie.

L'interface produite est satisfaisante, mais présente des limitations et reste une exploration de surface de ce qui est faisable avec les données obtenues. Sans contrainte de temps, il aurait été possible de travailler sur plusieurs interfaces, plusieurs métaphores et de réfléchir à plus d'idées originales.

Perspectives

L'approche du design dit autobiographique a été choisie lors de ce projet. En effet, l'idée derrière ledit projet est née d'une insatisfaction personnelle concernant les outils de recherche et de suggestion des plateformes de streaming de musique. Cette stratégie de design a plusieurs avantages, mais présente l'inconvénient majeur de ne pas inclure de perspectives autres que la mienne et celle de mon directeur de recherche.

Avec plus de temps, la prochaine étape aurait été de développer un protocole faire une demande auprès du comité d'éthique pour réaliser une *user-study*. Mais ce processus peut prendre plusieurs mois et nécessite un suivi du projet pendant un période beaucoup plus longue.

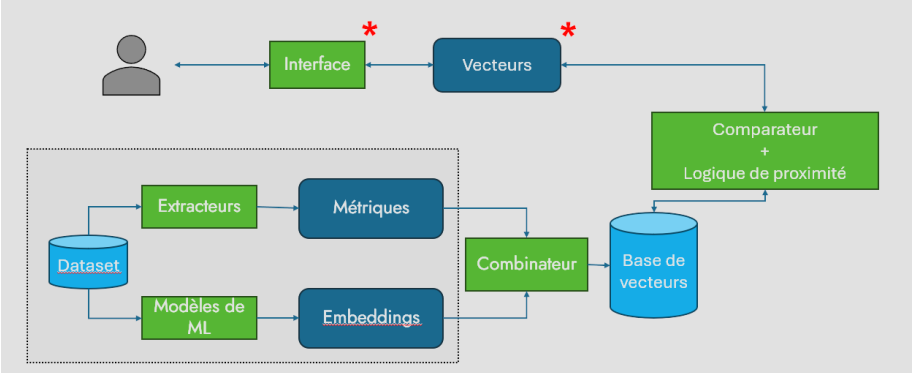
Concernant notre interface, il est possible de l'améliorer de plusieurs façons. Un des points les plus immédiats serait de la relier à un service de statistiques d'écoute comme last.fm[40] et s'en servir pour ajuster la façon dont les bacs sont générés.
On pourrait aussi proposer une vue plus dynamique, en 3D, avec la possibilité de déambuler dans le magasin et d'interagir avec le « propriétaire » qui nous fournirait des recommandations personnalisées. On se retrouverait donc au fur et à mesure des utilisations avec un disquaire qui finirait par recommander à l'utilisateur des titres qui correspondent exactement à ce qu'il recherche tout en lui offrant la possibilité de naviguer dans la boutique a son rythme, au gré des recherches transversales.

Il serait aussi possible d'enrichir visuellement les bacs en générant avec un modèle de génération d'image des jaquettes en fonction de divers paramètres, comme les paroles, le genre ou les émotions.

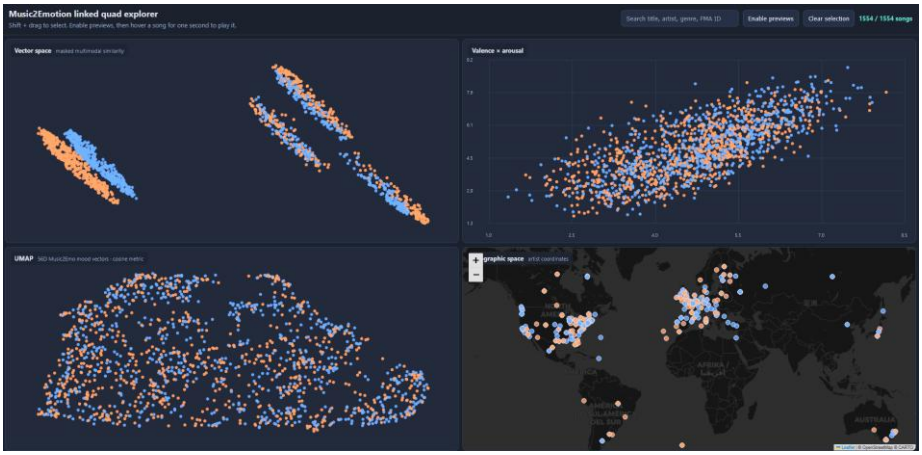
Figures

Ci-dessous : Diagramme illustrant l'architecture générale du projet

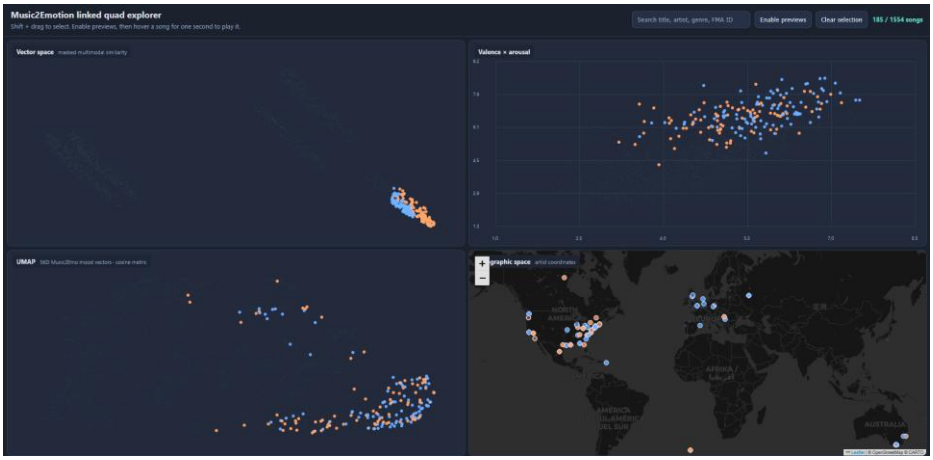
[40] https://www.last.fm/



Ci-dessous : Capture d’écran du tableau de bord principal.



Ci-dessous : Capture d’écran du tableau de bord avec une sélection arbitraire.



Ci-dessous : visualiseur dynamique pour les *embeddings* de paroles.

Ci-dessous : Interface métaphore (1/2)

Ci-dessous : Interface métaphore (2/2)