le cnam

Habilitation à diriger des recherches

présentée par

Raphaël Fournier-S'niehotta

Maître de Conférences au Conservatoire National des Arts et Métiers
Section CNU 27

Modéliser avec des graphes, pour l'informatique musicale et les systèmes de recommandation

après avis de :

Mme	Armelle Brun	Professeure des Universités	Université de Lorraine
Mme	Florence Levé	Professeure des Universités	Université de Picardie Jules Verne
M.	Camille Roth	Directeur de Recherche	CNRS

et soutenue publiquement le 10 septembre 2024 devant le jury formé par :

Mme	Armelle Brun	Professeure des Universités	Université de Lorraine
M.	Vincent Guigue	Professeur	AgroParisTech
Mme	Safia Kedad-Sidhoum	Professeure des Universités	CNAM Paris
Mme	Florence Levé	Professeure des Universités	Université de Picardie Jules Verne
M.	Philippe Rigaux	Professeur des Universités	CNAM Paris
M.	Camille Roth	Directeur de Recherche	CNRS

À Marie, Aurélien et Clara,

Remerciements

Une Habilitation à Diriger des Recherches est un travail individuel, mais la recherche qui précède est collective. Tous les travaux qui seront présentés dans ce manuscrit sont le fruit de collaborations avec des collègues, permanents ou non, français et étrangers.

Je remercie tout d'abord Armelle Brun, Florence Levé, et Camille Roth qui me font l'honneur d'être les rapporteurs de mon travail, et Vincent Guigue qui a accepté de participer au jury, aux côtés de Philippe Rigaux, le garant de cette habilitation à diriger des recherches.

Je remercie chaleureusement ceux qui ont été les guides de mon parcours professionnel, Matthieu Latapy et Philippe Rigaux, mes mentors scientifiques. Travailler à leurs côtés a toujours été une chance.

Je remercie pour leur travail toutes les personnes avec qui j'ai « fait » de la science. Pour certaines, nous sommes devenus co-auteurs parce que nous avons eu la chance que nos travaux soient publiés. J'aimerais distinguer particulièrement Tiphaine Viard, Lionel Tabourier, Nicolas Travers, Victor David, Pedro Ramaciotti, Francesco Foscarin, Yannis Karmim et Tiange Zhu. Avec d'autres, nous avons partagé des séances de réflexion qui sont devenues des rapports de projets, des morceaux de manuscrits de thèse, des rapports de stage, voire rien de tout cela, et ce n'est pas grave : ces moments ont au moins contribué à créer et échanger de la connaissance, à nous permettre d'apprendre et de mieux comprendre le monde, ce qui est l'essentiel.

Des collègues du CNAM sont devenus des c(n)amarades, des amis, je les remercie grandement pour les moments passés ensemble : Josselin Noirel, Chloé Mimeau, Nicolas Audebert, Simon Marié, Amélie Danlos, Rafik Abdesselam, Clément Rambour, Camilla Fiorini et Gérald Kembellec. Bien entendu, d'autres collègues très appréciés ont certainement été oublié·e·s : je les prie de me pardonner.

Je remercie aussi les personnels administratifs qui contribuent à ce que nos laboratoires de recherche et départements d'enseignement fonctionnent.

Je remercie toutes celles et ceux qui m'ont soutenu, d'une manière ou d'une autre, durant la période d'écriture de ce manuscrit. Je salue au passage Lana Del Rey, Korn, Queens of the stone age, TV on the Radio, Cansei de Ser Sexy, Ennio Morricone et Eminem, mes partenaires d'écriture réguliers.

Enfin, ma vie ne serait pas aussi belle sans Marie, Aurélien et Clara, mais quelques mots dans ce manuscrit ne sauraient suffire pour les remercier.

Toutes mes excuses à ceux que j'ai pu oublier. S'ils m'ont lu jusque-là, ils méritent mes remerciements.

Table des matières simplifiée

Introduction 1

I Informatique musicale 5

1 Introduction 7

2 Représenter la musique notée 21

3 Rechercher de l'information 31

4 Produire des partitions 43

5 Conclusion et perspectives 53

II Systèmes de recommandation et graphes 57

1 Introduction 59

2 Diversité des recommandations 73

3 Apprendre sur des graphes, pour recommander 85

4 Conclusion et perspectives 95

Annexes 97

Bibliographie 99

Résumé en français 117

English abstract 119

Table des matières

Introduction **1**

I Informatique musicale **5**

1 Introduction **7**
1.1 Brève histoire de l'informatique musicale 7
1.2 État de l'art 9
1.2.1 Éléments sur la notation musicale 10
1.2.2 Représentations numériques de données musicales symboliques 11
1.2.3 Qualité et gestion de données 14
1.2.4 Recherche d'information 15
1.3 Positionnement et contributions 16

2 Représenter la musique notée **21**
2.1 Modéliser le contenu musical 22
2.1.1 Modèle de données 22
2.1.2 Grammaires et descripteurs de contenu 25
2.2 Une algèbre de partitions 27
2.3 Conclusion 30

3 Rechercher de l'information **31**
3.1 Extraire les caractéristiques musicales 32
3.1.1 Mélodie 33
3.1.2 Rythme et parole 35
3.2 Indexer 36
3.3 Rechercher 36
3.4 Classer les résultats 37
3.4.1 Similarité par blocs de rythme 37
3.4.2 Classement final 39
3.5 Implémentation et interface graphique 39
3.5.1 Architecture 39
3.5.2 Interface 40
3.6 Conclusion 40

4 Produire des partitions **43**
4.1 Interroger et manipuler des (ensembles de) partitions 44
4.1.1 Schémas 45
4.1.2 Définition des requêtes 45
4.1.3 Évaluation des requêtes 46
4.2 Comparer des partitions 47
4.2.1 Arbres de ligatures, arbres de n-olet 48
4.2.2 Comparaisons inter-mesures et intra-mesure 49
4.2.3 Visualisation de différences 50
4.3 Conclusion 51

5 Conclusion et perspectives **53**
5.1 Contributions 53
5.2 Visibilité et diffusion 54
5.3 Perspectives 55

II Systèmes de recommandation et graphes **57**

1 Introduction **59**
1.1 Les systèmes de recommandation et les réseaux complexes 59
1.1.1 La recommandation 59
1.1.2 La science des réseaux 61
1.2 Positionnement et contributions 62
1.3 État de l'art 65
1.3.1 Recommandation 65
1.3.2 Modèles de graphes 67
1.3.3 Recommander avec des graphes 68

2 Diversité des recommandations **73**
2.1 Un modèle de graphes pour évaluer la diversité 74
2.1.1 Définitions 74
2.1.2 Les *vraies* diversités 75
2.1.3 Marches aléatoires et Heterogeneous Information Networks 77
2.2 Diversités individuelle et collective 77
2.3 Expériences 80
2.4 Conclusion 83

3 Apprendre sur des graphes, pour recommander **85**
3.1 Entraînement des modèles *GNN* en recommandation 86
3.1.1 Problématique 86
3.1.2 Optimisation directe avec le rang 87
3.1.3 Échantillonnage d'exemples négatifs 88
3.1.4 Protocoles expérimentaux inductif et transductif 89

3.2 Expériences 89
3.2.1 Protocoles 90
3.2.2 Résultats 91
3.2.3 Analyses quantitative et qualitative, du modèle 92
3.3 Conclusion 93

4 Conclusion et perspectives 95
4.1 Contributions 95
4.2 Perspectives 96

Annexes 97

Bibliographie 99

Résumé en français 117

***English abstract* 119**

Introduction

— Pourquoi ne dites-vous rien, commissaire ?
— J'essaie de réfléchir.
— Et ça donne quoi ?
— Rien. Mais je suis habitué.

Fred Vargas[1]

Ce mémoire d'Habilitation à Diriger des Recherches présente les travaux de recherche que j'ai effectués au Cnam depuis 2014. Durant cette période, j'ai mené des travaux autour de deux thématiques principales, la modélisation de partitions musicales symboliques, et des modèles de graphes, en particulier pour les systèmes de recommandation.

J'ai commencé mon parcours dans la recherche en stage de Master en 2009, dans une équipe étudiant les graphes de terrain (ou « réseaux complexes », en anglais *complex networks*), qui se trouve aux confins de la physique, de la biologie, de l'informatique et des mathématiques. J'ai prolongé ce stage par un doctorat centré autour de la fouille de données provenant de moteurs de recherche [Fou12].

C'est en post-doctorat, à l'université Paris XIII[2] que j'ai commencé l'étude des systèmes de recommandation. J'ai poursuivi dans ce champ depuis lors, souvent avec des approches reposant sur des graphes (une approche originale dans la communauté), d'abord sans apprentissage, puis avec. Avec mon recrutement dans l'équipe Vertigo au Cnam en 2014, j'ai diversifié mes thématiques de recherche, collaborant avec les membres de l'équipe sur l'informatique musicale.

Contexte de recherche

Dans cette section, je détaille les projets de recherche auxquels j'ai participé depuis 2014, ainsi que les sources de financement qui m'ont permis de faire de la recherche. Si l'obtention d'argent

[1] Fred Vargas. *L'homme aux cercles bleus.* Paris : Viviane Hamy, 1991.
[2] Aujourd'hui, Université Sorbonne-Paris-Nord.

« par projets » n'est pas le seul moyen à notre disposition pour mener nos recherches à bien, c'est aujourd'hui un élément central de l'exercice de notre métier. La diversité de ces projets, leur nombre, les montants de financement et les durées, tout cela explique en partie, et structure, le cheminement de ma carrière de chercheur.

La bibliothèque de partitions Neuma : projets Munir et Polifonia

L'équipe Vertigo, sous la houlette de Philippe Rigaux, avait mené avant mon arrivée un projet ANR, appelé Neuma, ayant abouti à la mise en place d'une bibliothèque numérique de partitions musicales du même nom[3]. Nous maintenons toujours la plateforme, en assurant le développement et l'hébergement.

Nous avons, dès mon arrivée, été candidats pour trouver des sources de financement de travaux autour de cette plateforme, que ce soit auprès de l'ANR avec un consortium élargi, ou plus modestement pour des bourses de thèses au sein du laboratoire. Si les demandes de doctorats n'ont pas été fructueuses, nous avons fini par obtenir un financement ANR avec le projet franco-québecois MuNIR[4], puis un financement européen, le projet Polifonia[5].

Le projet MuNIR, en collaboration avec l'équipe d'Ichiro Fujinaga (Université McGill), mettait l'accent sur la transcription automatique de partitions, et cela finança mon premier encadrement de doctorat, celui de Francesco Foscarin, de 2017 à 2020. Avec Philippe Rigaux et Nicolas Travers, nous avions au préalable travaillé sur la modélisation de partitions musicales, puis proposé une algèbre de manipulation et un langage d'interrogation reposant sur XQuery, ce que je détaillerai dans le chapitre 2. Le doctorat de Francesco Foscarin était centré sur la transcription automatique de partitions, c'est-à-dire la production d'une partition musicale (représentation lisible par les humains) à partir de documents MIDI. Nous avons collaboré, avec Francesco Foscarin, pour développer des outils conceptuels et logiciels pour comparer efficacement des partitions (comme le `diff` pour le texte). Ces travaux sont présentés dans le chapitre 4, et ils ont été les prémisses des travaux détaillés dans le chapitre 3.

Depuis 2021, dans le cadre du projet Polifonia, j'étudie et propose des méthodes pour la recherche d'information de contenu musicaux symboliques. Nous travaillons, dans le doctorat de Tiange Zhu, à la mise au point d'un moteur de recherche. Le cœur du moteur repose sur une modélisation efficace du contenu musical à partir de *n-grams*, ce qui permet l'intégration dans des moteurs de recherche de texte ainsi que l'interrogation de diverses dimensions de la musique. Le chapitre 3 sera consacré à la présentation de ce travail.

Diversité dans les systèmes de recommandation : le projet Algodiv

Mon second axe de recherche est centré sur la modélisation de graphes (éventuellement dynamiques), appliquée à divers problèmes dont les systèmes de recommandation.

[3]Aujourd'hui, Neuma est hébergée sur HumaNum, TGIR du CNRS. Voir : http://neuma.huma-num.fr/.
[5]Voir https://anr.fr/Projet-ANR-16-FRQC-0007.
[5]Voir https://polifonia-project.eu/.

Dans un premier temps, j'ai étudié les Heterogeneous Information Networks (HIN), une généralisation des graphes. Avec plusieurs collègues, nous avons examiné l'intérêt de ce modèle pour caractériser la *diversité*. Nous avons élaboré une taxonomie aussi exhaustive que possible d'indicateurs, et appliqué ces résultats à des systèmes de recommandation [Ram+21a ; RTF20]. Ces travaux ont été menés dans le projet ANR Algodiv[6] durant lequel j'ai coencadré le post-doctorat de Pedro Ramaciotti-Morales avec Lionel Tabourier.

J'ai également travaillé, lors du post-doctorat de Tiphaine Viard, sur le modèle de graphes dynamiques des flots de liens, pour en examiner diverses propriétés (recherche de cliques temporelles, densités temporelles locales) [Via+18 ; Gau+16], avant d'étudier les possibilités d'améliorer les performances de systèmes de recommandation (en prenant en compte pleinement la dimension temporelle du problème) [VF19a ; VF19b].

Apprentissage et graphes

Depuis 2021, j'encadre le doctorat de Yannis Karmim avec le professeur Nicolas Thome (CNAM puis Sorbonne Université) et nous approchons cette fois le problème de la prise en compte de l'aspect temporel de la recommandation par l'étude de techniques d'apprentissage profond, avec les modèles de *Graph Neural Networks* (GNN) [Kar+22]. Nous avons commencé par une collaboration avec Elias Ramzi autour de l'entraînement des GNN. Nous étudions maintenant les graphes dynamiques, et les architectures Transformers (proches des GNN). Ces travaux poursuivent et rejoignent une ligne de recherche que j'ai également menée ces dernières années, autour du développement et de l'utilisation de modèles de graphes dynamiques ([Gau+16 ; Via+18 ; VF19b ; VF19a]).

J'avais préalablement étudié ces modèles GNN sur un problème de recherche d'information, en partenariat avec l'entreprise Qwant, pour améliorer l'indexation d'images dans un moteur de recherche, à l'aide d'une détection efficace de leurs légendes dans des pages web [Dan+21].

Raisonnement avec des graphes de faits : projet ANR Daphné

Dans un deuxième temps, j'ai mené des travaux sur le raisonnement sur des faits représentés dans des graphes, les liens modélisant les conflits éventuels entre faits. Avec Victor David et Nicolas Travers, nous avons proposé une approche originale par rapport à l'état de l'art pour construire des ensembles de faits cohérents entre eux, en maximisant le nombre et les probabilités des faits conservés, au détriment d'un ensemble de faits laissés de côté. Ce travail trouve son application dans la méthodologie historiographique (faits incertains), mais aussi en systèmes multi-agents (quand ces agents doivent raisonner dans des environnements avec des informations incomplètes). Pour conserver une certaine cohérence et une certaine concision à ce manuscrit, ces travaux qui sont pour l'essentiel encore en relecture ne sont pas présentés ici, le lecteur pourra se référer à [DFT22] ou [DFT23].

[6] Voir : http://algodiv.huma-num.fr.

Organisation du mémoire

Mes deux thèmes de recherche paraissent très différents, et ils le sont, dans une certaine mesure. Quoiqu'ils partagent des problématiques communes, telles que le développement de modélisations originales, j'ai choisi d'organiser ce mémoire en deux parties indépendantes.

Dans la première partie, sur la notation musicale symbolique, nous commençons par exposer, dans le chapitre 1, quelques éléments de contexte, sur la musique et les divers formats de représentation, ainsi qu'un état de l'art. Nous détaillerons ensuite, dans le chapitre 2, des travaux en modélisation du contenu musical, en vue d'en améliorer la manipulation et l'interrogation. Puis, nous verrons dans le chapitre 3, comment une autre modélisation de ce contenu musical le rend intégrable dans un moteur de recherche standard, pour permettre la gestion de grandes collections de partitions. Enfin, le chapitre 4 étudie la comparaison de partitions musicales.

Dans la seconde partie, nous présentons des travaux sur les systèmes de recommandation, avec des approches orientées graphes. Après avoir exposé les principes de la recommandation et l'état de l'art de la partie dans le chapitre 1, nous montrons dans le chapitre 2 que ces modélisations permettent d'accéder à une compréhension particulière des phénomènes de diversité algorithmique. Enfin, nous proposons d'autres contributions dans le chapitre 3, autour des architectures GNN pour la recommandation.

Des chapitres de conclusion terminent chacune des parties en en rappelant les contributions et en exposant des perspectives de recherche.

Note sur la langue. Ce manuscrit est en français, rédigé pour préparer un diplôme français. Cependant, les termes anglais y sont assez fréquents. D'une part, parce que les communications scientifiques internationales sont quasi exclusivement en anglais. Il y a donc des termes dont l'équivalent français n'est pas toujours usité, et je demande l'indulgence du lecteur si j'ai commis des maladresses en les traduisant. Et d'autre part, certaines figures ont été reprises de travaux publiés en anglais, et n'ont pas été traduites.

Première partie

Informatique musicale

Chapitre 1

Introduction

Supposing, for instance, that the fundamental relations of pitched sounds in the science of harmony and of musical composition were susceptible of such expression and adaptations, the [Analytical Engine] might compose elaborate and scientific pieces of music of any degree of complexity or extent.

Ada Augusta King, Comtesse de Lovelace[1]

Sommaire

1.1 Brève histoire de l'informatique musicale **7**
1.2 État de l'art **9**
1.2.1 Éléments sur la notation musicale 10
1.2.2 Représentations numériques de données musicales symboliques 11
1.2.3 Qualité et gestion de données 14
1.2.4 Recherche d'information 15
1.3 Positionnement et contributions **16**

1.1 Brève histoire de l'informatique musicale

La musique[2] et la technique sont liées depuis longtemps, chacune influençant l'autre de diverses façons. Les innovations techniques ont transformé continûment la manière dont la musique est

[1] Luigi Federico Menabrea et al. *Sketch of the Analytical Engine Invented by Charles Babbage, Esq.* Richard et John E. Taylor, 1843.

[2] Si la musique ne se limite pas à l'Occident, nous appellerons dans la suite « musique » et « partitions musicales » ce qui a trait à la *tradition occidentale de la musique* (Common Western notation). Certains des travaux évoqués peuvent s'appliquer également à d'autres traditions musicales, d'autres non.

composée, jouée, partagée, ressentie. D'abord de simples outils faits en matériaux naturels, les instruments sont peu à peu devenus plus complexes. Le phonographe, breveté en 1877 par Edison, a constitué une rupture fondamentale car il a permis que la musique soit enregistrée, puis reproduite ultérieurement, éventuellement loin dans le temps et l'espace. L'industrie musicale que nous connaissons aujourd'hui s'est alors progressivement structurée pour distribuer cette musique enregistrée.

Ada Lovelace mentionna dès 1843, dans son article fondamental sur la machine de Charles Babbage[1], la possibilité de composer de la musique à l'aide ce qui ne s'appelait pas encore ordinateur. Plus tard, au XXe siècle, l'arrivée de l'électronique a conduit des compositeurs comme Schaeffer ou Stockhausen à explorer les possibilités de la musique électroacoustique. En 1957, Max Matthews inventa le langage MUSIC I pour l'IBM 704, ce qui permit à son collègue des Bells Labs, Newman Guttman, de composer « *In the Silver scale* », la première pièce musicale entièrement jouée par un ordinateur. La composition algorithmique est devenue une discipline, avec d'autres langages de programmation et de nombreux logiciels. De la musique de styles très divers a ainsi été créée, allant du classique à l'expérimental. La seconde moitié du XXe siècle a aussi vu la « révolution numérique » chambouler la musique, avec l'avènement d'outils numériques d'enregistrement et de montage, permettant un contrôle inédit sur toute la chaîne de production d'un morceau. Cette possibilité de tout faire sur un ordinateur a démocratisé l'accès à la musique et ouvert de nouvelles pistes pour les artistes. Au-delà des machines elles-mêmes, leur interconnexion via l'Internet a changé comment la musique est distribuée, des premiers réseaux P2P comme Napster aux plateformes mondiales actuelles comme Spotify ou Apple Music. Et l'album, concept lié à un objet physique (disque vinyle, CD ou cassette), devient de moins en moins central, au profit de la granularité de la *piste* (éventuellement groupée en listes de lecture – *playlists*). Sur ces plateformes, le choix des consommateurs est influencé par des systèmes dits de « recommandation », qui ont pour ambition de proposer des morceaux, voire des artistes, qui plairont aux utilisateurs. Ces systèmes de recommandation, dont nous reparlerons dans la seconde partie de ce manuscrit, utilisent l'analyse de données massives, voire l'apprentissage, pour effectuer leurs calculs. Ils peuvent être vus comme issus de travaux académiques et industriels en *recherche d'information musicale*, un champ de recherche connexe à la recherche d'information.

Ces travaux académiques en informatique musicale se sont précisément structurés au début du XXIe siècle, à la croisée de la musicologie, de l'informatique et des sciences de la documentation. La conférence ISMIR (*International Society for Music Information Retrieval*), créée en l'an 2000, est un élément central de la communauté. La discipline est désignée par les expressions « recherche d'information musicale » ou « recherche en informatique musicale » (en anglais, *Music Information Retrieval* ou *Music Information Research*, la seconde étant plus générale que la première). Elle concerne l'étude des données numériques liées à la musique, c'est-à-dire la collecte et l'organisation de données musicales compatibles avec des traitements informatisés, ainsi que le développement de représentations et de méthodologies d'analyse spécifiques [Gir+14 ; MF06].

La recherche d'information musicale est donc au cœur de ce que l'on appelle les *humanités numériques* (*Digital Humanities* [SSU08]), les champs d'étude qui adoptent des méthodes et d'outils numériques en sciences humaines et sociales. Le rapport entre les sciences sociales et le numérique a une histoire longue et complexe, qui peut être articulée en trois mouvements : *Litterary and Linguistic Computing* (1950-1980), *Humanities Computing* (1980-1994) et *Digital Humanities* (depuis l'arrivée du Web) [Bur20][3]. La première période est marquée par le développement de modèles quantitatifs et d'analyses statistiques, la deuxième par la formalisation (conception de programmes, *Text Encoding*

Initiative). Le troisième temps, celui de la *numérisation*) des humanités, commence avec l'arrivée du Web qui transforme toute la recherche en sciences sociales : on passe d'outils pour la recherche à un environnement au sein duquel s'effectue la recherche, et à des artefacts nativement et massivement numériques, que l'on cherche à gérer (à l'aide de bibliothèques numériques).

Cette proximité accrue entre informatique et sciences sociales a progressivement, au-delà des outils, apporté des questionnements méthodologiques. La préservation d'artefacts physiques est devenue une préoccupation majeure, qui s'est peu à peu dotée des techniques de *gestion des données*. Dans la plupart des disciplines, on a cherché à dépasser cela, pour viser une *re-représentation* et une bonne *manipulation des données*, afin d'en révéler des propriétés qui ne sont pas apparentes sous la forme première de ces artefacts. Les perspectives ouvertes par l'approche numérique sont très discutées, des thuriféraires qui imaginent qu'on fera des découvertes impossibles sans elle, à des plus modérés qui pointent l'absence (et le besoin) de structuration du mouvement et de ses méthodes [Gui19 ; Ber15 ; DM15]. Néanmoins, les humanités numériques constituent aujourd'hui une approche de recherche active, dans laquelle des musicologues s'inscrivent, tels Christophe Guillotel [Gui20] et (certains de) ses collègues de l'IReMus. Les informaticiens ont, eux, « [investi] le champ des sciences sociales sans se revendiquer des « humanités numériques » et [soulevé] des questions humanistes pertinentes » [Gui19], et l'interdisciplinarité est aujourd'hui une position de recherche très fertile, en musique comme dans d'autres domaines.

La recherche d'information musicale partage nombre d'enjeux et de méthodes avec la *recherche d'information* classique. Les enjeux *scientifiques* de la discipline s'articulent donc d'abord autour de la question de **données**, suivie immédiatement de la question de la **représentation des informations**. En vue de manipuler et d'extraire de nouvelles informations, les **méthodologies de traitement des données** et, dans une moindre mesure, **l'évaluation de ces traitements** sont également des questions structurantes de la discipline. Des enjeux *technologiques* sont aussi au cœur des préoccupations, avec l'ambition d'obtenir des outils d'organisation, de stockage, de découverte, de recherche, et d'analyse d'informations musicales. Enfin, remarquons que la musique est un phénomène socio-culturel, et divers enjeux de cet ordre sont également importants dans le domaine, comme l'étude des comportements individuels et collectifs liés à la musique (influences, multiculturalisme, etc.). Cependant, nous n'aborderons que marginalement les enjeux de cet ordre dans ce document, de même que ceux directement relatifs à la perception de la musique.

Nous détaillons, ci-dessous et dans les chapitres suivants, les défis afférents à ces axes de travail, l'état de l'art et les réponses que mes travaux ont contribué à apporter à la discipline.

1.2 État de l'art

Nous commençons par expliquer la nature et l'origine des données symboliques, avant de détailler leur représentation informatisée, à l'aide de formats de fichiers ou de langages. Nous aborderons ensuite le traitement de ces données, avec une attention particulière sur les possibilités d'interrogation de corpus, et la comparaison de tels documents.

[3] On retrouve, dans le premier chapitre de [SSU08], un découpage similaire, avec un début en 1950, un tournant avec l'arrivée du Web, et deux périodes à la place du *Humanities computing*, appelées « *Consolidation* » et « *New developments* ».

1.2.1 Éléments sur la notation musicale

Les sons sont physiquement caractérisés par leur fréquence, que l'on appelle en musique la *hauteur*. Les humains entendent des fréquences approximatives de 20 à 20 000 Hz et peuvent, même sans entraînement, déterminer facilement si deux sons ont la même hauteur ou si leurs fréquences ont un rapport spécifique (par exemple, double) : deux sons avec un rapport de fréquence $\frac{1}{2}$ sont séparés d'une *octave*, et deux hauteurs éloignées d'une ou plusieurs octaves sont perçues comme similaires (consonance). La musique occidentale utilise depuis plusieurs siècles le système dit *tempéré*, qui divise une octave en 12 notes régulièrement espacées, selon la formule : $f_p = 2^{1/12} f_{p-1}$. L'écart entre deux hauteurs s'appelle un intervalle, il est défini par un rapport de fréquences. L'intervalle le plus faible, le demi-ton, est défini par $\frac{f_p}{f_{p-1}} = 2^{\frac{1}{12}}$. Avec 8 octaves de 12 demi-tons, il y a 96 niveaux de fréquences possibles dans la partie audible du spectre [Kla06].

Un ensemble de hauteurs qui sont à $n \in \mathbb{N}$ octaves d'intervalle constitue une *classe de hauteur* (*pitch class*, en anglais), généralement indiqué par un nombre de $[0, 12[$ ou par une lettre. On distingue en particulier, 7 classes de hauteur (0, 2, 4, 5, 7, 9, 11), désignées par une lettre : 'C', 'D', 'E', 'F', 'G', 'A', 'B' (en français : 'Do', 'Ré', 'Mi', 'Fa', 'Sol', 'La', 'Si'). Les noms des autres classes sont obtenus en ajoutant le symbole $\sharp$ à la lettre de la classe de hauteur la plus à gauche ou en ajoutant le symbole $\flat$ à la lettre de la classe de hauteur la plus à droite. Par exemple, F$\sharp$ et G$\flat$ correspondent à la classe de hauteur 6, ils sont dits enharmoniques [CC05]. Il suffit de spécifier la fréquence d'un son de référence pour pouvoir calculer la fréquence de toutes les hauteurs. Le choix le plus courant est de fixer la classe de hauteur 'A' (le La) à 440Hz.

La notation occidentale de la musique est organisée temporellement selon une pulsation périodique, appelée le *temps*. La fréquence des battements est appelée le *tempo* et est exprimée en battements par minute (BPM). Le rythme est organisé de manière hiérarchique : les temps sont subdivisés en unités égales (demi-temps, tiers de temps) ou assemblés en unités plus longues comme les *mesures*. D'autres divisions récursives sont utilisées, générant une hiérarchie appelée *structure métrique*. Musicalement, la première unité d'une division est, en général, considérée comme plus importante que les autres [CCM63].

Pour noter la musique, on doit donc organiser des éléments *mélodiques* et des éléments *rythmiques*. Les partitions musicales utilisent aujourd'hui une grille de 5 lignes, appelée *portée*. Cette grille peut être vue comme une portion de la grille complète qui contiendrait toutes les hauteurs possibles, et dont la position relative dans cette grille complète dépend d'un symbole initial, la *clef* (*cf.* figure 1.1). La représentation du temps se fait avec la *signature* : un rationnel ($4/4$, sur la figure 1.1) donne l'unité de temps et le découpage en *mesures*. Sur une portée, la notation est discrétisée. Une *note* indique la production attendue d'un son, avec un symbole encodant deux informations essentielles : la *fréquence* du son que l'on souhaite voir jouer et sa *durée* (relative au *temps*).

Les *polyphonies* combinent plusieurs voix, comme le montre la figure 1.2. La synchronisation des sons entre voix se traduit graphiquement par un alignement vertical. Le premier son est ainsi une combinaison de 3 notes, C3, C5 et E5 (de bas en haut). Les deux notes du dessus ont la même durée (une noire), la 3e (la basse) est une ronde (qui se synchronise avec 4 notes des voix du dessus).

Outre les éléments musicaux, les partitions contiennent également des éléments graphiques, qui constituent des conventions pour les interprètes : nous verrons dans la suite que ces éléments ont guidé les premières représentations informatisées, ce qui a quelque peu limité leur possibilités.

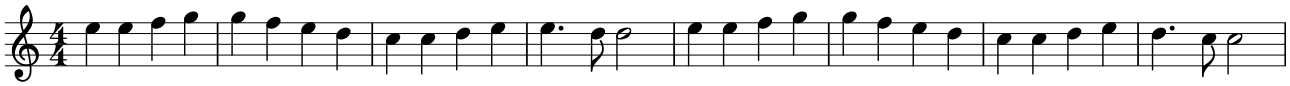


Figure 1.1 : *Ode à la joie, extraite de la 9e symphonie de Beethoven.*

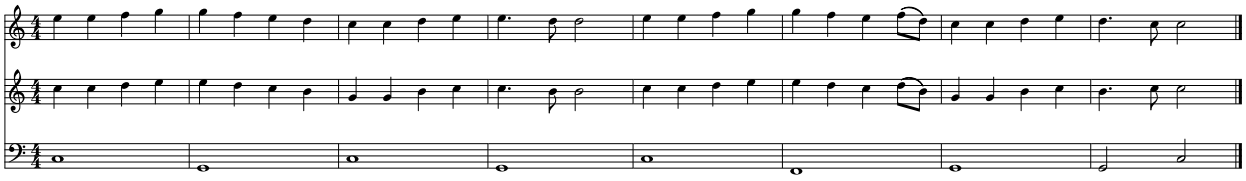


Figure 1.2 : *Ode à la joie, avec trois instruments.*

1.2.2 Représentations numériques de données musicales symboliques

Avec l'apparition et le perfectionnement des ordinateurs durant le XXe siècle, on a progressivement utilisé ces machines pour représenter l'information musicale sous forme symbolique. Les éditeurs de partitions ont peu à peu employé des outils de PAO[4] dédiés pour élaborer les documents à imprimer, les compositeurs ont utilisé des programmes et logiciels pour écrire leurs œuvres. Les partitions numériques peuvent également résulter d'un processus de transcription, automatique ou manuelle, d'une performance. Avec les synthétiseurs électroniques, on peut obtenir des partitions *explicites* ou *implicites* (issues de la représentation logicielle interne de l'outil). Dans la tradition musicale occidentale classique, les interprètes ont une certaine liberté pour jouer la partition « prescrite », que ce soit pour le rythme, des ornements voire la mélodie (une cadence).

Les documentalistes et chercheurs en informatiques ont ainsi proposé des formats de fichiers et des langages pour organiser, interroger et explorer des collections de documents musicaux.

Formats de fichiers

De nombreux formats de stockage et d'échange de données symboliques musicales ont été développés, nous allons restreindre notre exposé aux plus pertinents pour notre propos.

Le format MIDI, pour Musical Instrument Digital Interface (MIDI) représente les entrées et sorties sonores d'un instrument numérique (comme un synthétiseur), ainsi que la synchronisation de certains canaux de sorties. Quoique proche d'instruments produisant du son, ce format n'encode pas le son comme un signal, mais comme un ensemble d'événements discrets, avec de nombreux paramètres (hauteur, instruments, appui/relâchement d'une touche de piano, par exemple). Les fichiers MIDI peuvent contenir des métadonnées (signature temporelle, tempo, tonalité), et sont régulièrement utilisés comme un substitut facile à manipuler pour les partitions musicales. Mais ils sont plus limités : on doit inférer les débuts et fin de notes (à partir d'événements *note-on* et *note-off*),

[4]PAO : Publication Assistée par Ordinateur.

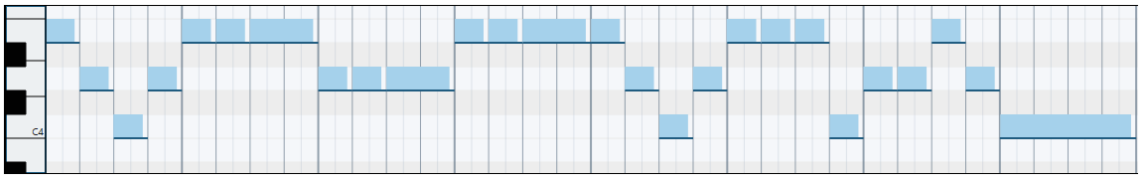


Figure 1.3 : *Un* piano-roll*, une représentation des données MIDI.*

on ne dispose pas de représentation pour les ligatures, les clefs, les silences ; enfin, il n'y a pas de différence un 'D' dièse et un 'E' bémol ou de séparation entre les différentes voix.

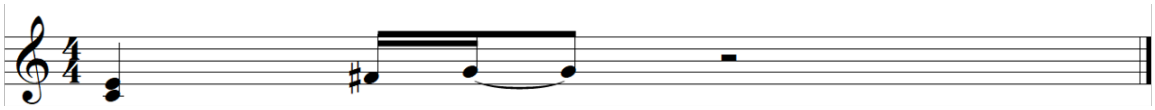
Ce format n'a pas été normalisé, il est initialement le fruit d'accords entre marques de synthétiseurs qui visaient l'interopérabilité de leurs appareils. Il ne repose donc pas sur une modélisation formelle préalable de la notation musicale.

Il existe également des formats de représentation textuels de la musique, comme ABC [OWA10], Musedata [Hew97], Humdrum **Kern [Hur93 ; Hur97], LilyPond [NNc16], Guido [Hoo+98] et MusicTeX [TME93]. Ils ont l'avantage d'être faciles à lire et à créer, un éditeur de texte suffit. Chaque format a des avantages et inconvénients pour parvenir représenter des éléments complexes tout en préservant la lisibilité. Mais ils montrent tous leurs limites lorsque la taille ou la complexité de la partition augmente trop. Les formats les plus récents, créés au début du XXIe siècle, sont dérivés d'XML : MusicXML [Goo01] et MEI [Rol02]. Ils font le choix de perdre un peu de la compacité de la représentation de leur prédécesseurs textuels, pour gagner en expressivité et complétude. Outre le contenu musical, ils incorporent des éléments graphiques de rendu. Leur principal avantage est de permettre l'accès (pour le programmeur) aux composants de la partition, avec un niveau de détail élevé. La figure 1.4 présente une comparaison visuelle entre MEI et MusicXML. MEI permet un accès à chaque élément de la partition avec un identifiant unique (id), et il est plus concis. MusicXML est davantage utilisé que MEI, notamment parce que les logiciels numériques commerciaux de notation l'utilisent (alors que seul Sibelius propose MEI). MEI a été pensé pour conserver et analyser des œuvres, il dispose de notations complémentaires à la *Common Western Notation* (notation mensurale, neumes, etc.).

Autres représentations

Ces formats de fichiers fournissent une réponse aux besoins de *conserver* et *transmettre* l'information musicale et, dans une certaine mesure, au besoin de l'utiliser. Cependant, ils n'offrent pas toutes les possibilités d'accéder aux éléments de la partition et à sa structure. Comme nous le détaillerons dans la section 1.3, la *modélisation fine du contenu musical* est donc fondamentale.

La littérature du domaine distingue plusieurs « niveaux » de caractéristiques musicales : on appelle usuellement « caractéristiques de bas niveau » les notes, et comme caractéristiques de second niveau (ou de haut niveau) les termes utilisés pour décrire la musique (genre, tonalité, rythme, harmonie). Les définitions des niveaux ont varié au fil des années, des auteurs, et des avancées techniques. Les caractéristiques de haut niveau sont en général plus complexes à représenter (et à extraire des données), et l'on fait donc en général un compromis entre la simplicité de la représentation et la complexité des caractéristiques disponibles.



MEI

```
...
<measure xml:id="measure-0000000232735099" right="end" n="1">
    <staff xml:id="staff-0000001963390883" n="1">
    <layer xml:id="layer-0000000738522925" n="1">
        <chord xml:id="chord-0000001488662809" dur.ppq="4" dur="4" stem.dir="up">
            <note xml:id="note-0000000224872045" oct="4" pname="c" />
            <note xml:id="note-0000001902316584" oct="4" pname="e" />
        </chord>
        <beam xml:id="beam-0000001404430694">
            <note xml:id="note-0000002146957621" dur.ppq="1" dur="16" oct="4" pname="f" stem.dir="up">
                <accid xml:id="accid-0000000545435759" accid="s" accid.ges="s" />
            </note>
            <note xml:id="note-0000000588407389" breaksec="1" dur.ppq="1" dur="16" oct="4" pname="g" stem.dir="up" />
            <note xml:id="note-0000001646010113" dur.ppq="2" dur="8" oct="4" pname="g" stem.dir="up" />
        </beam>
        <rest xml:id="rest-0000000011007280" dur.ppq="8" dur="2" />
    </layer>
    </staff>
    <tie xml:id="tie-0000000227382387" startid="#note-0000000588407389" endid="#note-0000001646010113" />
</measure>
...
```

MusicXML

```
...
<measure number="1" width="983">
  <note default-x="87">
    <pitch>
      <step>C</step>
      <octave>4</octave>
    </pitch>
    <duration>4</duration>
    <voice>1</voice>
    <type>quarter</type>
    <stem default-y="-5">up</stem>
  </note>
  <note default-x="87">
    <chord/>
    <pitch>
      <step>E</step>
      <octave>4</octave>
    </pitch>
    <duration>4</duration>
    <voice>1</voice>
    <type>quarter</type>
    <stem>up</stem>
  </note>
  <note default-x="308">
    <pitch>
      <step>F</step>
      <alter>1</alter>
      <octave>4</octave>
    </pitch>
    <duration>1</duration>
    <voice>1</voice>
    <type>16th</type>
    <accidental>sharp</accidental>
    <stem default-y="5">up</stem>
    <beam number="1">begin</beam>
    <beam number="2">begin</beam>
  </note>
  <note default-x="393">
    <pitch>
      <step>G</step>
      <octave>4</octave>
    </pitch>
    <duration>1</duration>
    <tie type="start"/>
    <voice>1</voice>
    <type>16th</type>
    <stem default-y="5">up</stem>
    <beam number="1">continue</beam>
    <beam number="2">end</beam>
    <notations>
      <tied type="start"/>
    </notations>
  </note>
  <note default-x="477">
    <pitch>
      <step>G</step>
      <octave>4</octave>
    </pitch>
    <duration>2</duration>
    <tie type="stop"/>
    <voice>1</voice>
    <type>eighth</type>
    <stem default-y="5">up</stem>
    <beam number="1">end</beam>
    <notations>
      <tied type="stop"/>
    </notations>
  </note>
  <note default-x="613">
    <rest/>
    <duration>8</duration>
    <voice>1</voice>
    <type>half</type>
  </note>
  <barline location="right">
    <bar-style>light-heavy</bar-style>
  </barline>
</measure>
...
```

Figure 1.4 : *Un exemple simple d'encodage d'une mesure en MusicXML et MEI.*

Au-delà des formats de fichiers, diverses autres approches, plus ou moins abstraites, ont été proposées, pour représenter le contenu musical. De nombreux travaux ont proposé des ontologies de représentation symbolique de la musique, comme [PG23], [RDM18] ou [Rai+07]. Des auteurs ont tenté d'apporter à l'informatique musicale des travaux sur les séries temporelles [Ada+04], pour prendre en compte la dimension temporelle de la musique. Les travaux de Paul Hudak [Hud15], et d'autres modèles de calculs formels adaptés à la programmation musicale [Bal96 ; Jan+13 ; FOL12 ; Fob+13] font figure de pionniers. Euterpea [Hud15], basé sur la programmation fonctionnelle (avec Haskell) a un système de typage très riche, et les opérateurs polymorphes d'un langage fonctionnel offrent de larges possibilités de génération d'objets musicaux complexes et d'une étude formelle de leurs propriétés. Dans le même esprit, le T-Calculus [Jan+13] vise une modélisation concise et cohérente de flux musicaux représentés sous forme de *tiled streams* et synchronisés par un opérateur de *tiled group*. En composition musicale, on trouve des approches utilisant des opérateurs "musicaux" et visant la représentation interactive de partitions avec INScore [FOL12 ; Fob+13]. On pourrait

également citer OpenMusic [Ass+99] ou SuperCollider [McC02] parmi les langages et frameworks de programmation musicaux les plus populaires.

1.2.3 Qualité et gestion de données

Les améliorations des formats de stockage et des logiciels d'édition numérique de partitions (Sibélius, Finale, etc.) ont accompagné et favorisé la création de partitions numériques. Il y a essentiellement deux processus de production de ces documents : la gravure et la transcription. La gravure, manuelle, relève de « l'artisanat », c'est un travail généralement long et coûteux, pratiqué par des professionnels comme des particuliers, pour éditer des partitions de compositeurs vivants ou non [Mat22]. La transcription automatisée offre donc une possibilité d'obtenir plus rapidement des données. Les sources sont principalement de deux natures : des fichiers MIDI (via des enregistrements sur instruments numériques, par exemple) ou des fichiers images, comme ceux conservés par Gallica[5]. La transcription d'images est axe de recherche à part entière, apparenté à la vision par ordinateur, que l'on appelle Optical Music Recognition (par calque avec Optical Character Recognition pour le texte). Nous ne détaillons pas ici l'état de l'art (on pourra se reporter à [Reb+12] ou [SF20] pour des *surveys* relativement récents), mais nous pouvons remarquer qu'il y a une proximité avec le travail que nous avons mené dans la thèse de Francesco Foscarin sur la transcription [Fos20] : comme à partir de formats MIDI, transcrire des partitions à partir d'images nécessite une réflexion éclairée par la musicologie sur la partition produite. Pour le format MIDI, s'il existe aujourd'hui divers algorithmes de transcription automatique, ce n'est pas un sujet clos, tant il est délicat d'aboutir à un résultat satisfaisant en terme de contenu musical comme de rendu graphique. La thèse de Francesco Foscarin [Fos20] présente les défis et des contributions significatives dans ce domaine.

Une transcription automatisée est régulièrement un *préalable* à une gravure manuelle, un point de départ. La qualité des données est fondamentale, à plusieurs titres [Fos+18a]. Outre le côté évidemment délétère de conserver des données fausses, des données erronées sont également plus difficile à interroger (puisque ne correspondant pas aux requêtes que les utilisateurs sont susceptibles de poser). Une étape de correction des des erreurs (de saisie ou de transcription), est donc indispensable pour aboutir à des fichiers de sortie MEI ou MusicXML les meilleurs possibles. La possibilité de valider un document XML au regard d'un schéma [W3C04] ne permet d'éliminer que certaines catégories d'erreurs.

Ainsi, il est important de disposer d'outils de comparaison de partitions, par exemple une partition en cours d'élaboration avec une partition de référence. Pour le texte, de tels outils et algorithmes de comparaison existent depuis plus de quarante ans (le `diff` Unix [HM76 ; Hec78], par exemple). Ils identifient les différences entre deux contenus relativement proches, par exemple deux versions d'un même texte, au niveau des lignes ou des caractères Leur sortie, normalisée, est une liste de différences, appelée *patch*. Si l'on dispose d'un fichier et du *patch*, on peut reconstituer le second. On peut également produire une *fusion* des deux fichiers, si les changements sont indépendants.

Pour la musique, nous l'avons dit, la notation comporte plusieurs niveaux (contenu musical, rendu graphique, par exemple), ce qui rend la création d'un tel outil assez délicat [Fos+18a ; CR18] :

[5]Exemple : https://gallica.bnf.fr/ark:/12148/bpt6k8733549/f1.item

on ne peut appliquer directement les outils pour le texte ou le XML aux arborescences musicXML. Les divers styles d'encodage musicXML compliquent la tâche [KB07]. Beaucoup de propositions existantes se concentrent sur le contenu musical, délaissent le rendu graphique, et reposent sur des distances d'éditions [MS90 ; All+09 ; Lem00 ; CD17 ; MS18], éventuellement sur des arbres [Riz10 ; Ber+11].

Les partitions numérisées ont été rassemblées en corpus, par diverses institutions et groupes. Elles ont d'abord été conservées dans des bases de données classiques, puis des bibliothèques dédiées sont apparues, parmi lesquelles Neuma [Neuma], IMSLP [LLC06], MuseScore [BVB02], RISM [Sou52] ou Musipedia [Typ97]. Celles-ci peuvent utiliser des métadonnées (compositeurs, interprètes, dates et lieux, etc.) ainsi que des données sur le contenu des œuvres (mélodies, rythmes). Comme les autres bibliothèques, elles peuvent mettre à profit des standards ou normes, comme Dublin Core, ou des modèles comme les spécifications fonctionnelles des notices bibliographiques (Functional Requirements for Bibliographic Records, FRBR). Les bibliothèques de partitions ont des objectifs qui dépassent la seule conservation de documents musicaux : navigation organisée dans des collections (par époque, compositeur, genre), analyse musicale, partage collaboratif, recherche par le contenu, recherche par similarité [Rig+12]. Une tentative de transposer les principes des bases de données à la gestion de partitions a été présenté dans [FR10], et on a étudié comment XQuery pourrait être utilisé directement sur le format MusicXML, concluant que ce langage était trop générique [GSD08].

1.2.4 Recherche d'information

La recherche d'information musicale est proche de la recherche d'information pour le texte. Les deux problématiques centrales sont la « recherche de documents », qui consiste à rassembler les documents d'une collection qui correspondent à une requête formée de mots (ou d'un document-exemple), et la classification, dont l'objectif est d'assigner à un document une ou plusieurs classes. La recherche de similarités entre documents peut être vue comme connexe à cette tâche de classification. Appliquer les techniques du texte à la musique suppose d'être capable de trouver des équivalents satisfaisants à la notion de mots, ce qui s'avère plus aisé en notation symbolique qu'avec des contenus audio.

Le développement de moteurs de recherche dédiés au contenu musical est un sujet relativement émergent, notamment faute de besoin : il existait peu de grandes collections de musiques numérisées auparavant. Un article pionnier de Typke *et al.* a recensé les systèmes de base pour la recherche d'information musicale [TWV05], suivi quelques années plus tard par celui de Nanopoulos *et al.* ([Nan+09]). Dès 2011, une structure d'indexation reposant sur des n-grams a été décrite dans [Con+11] et un *Music Ngram Viewer* a été développé par l'entreprise Peachnote [Vir11]. L'idée de diviser les séquences musicales en n-grams avait été proposée dans des travaux antérieurs [Dow99 ; NO04 ; DR04 ; CJ06], mais pas pour de l'indexation. Ces dernières années, quelques projets de mo-

teurs de recherche ont été décrits dans des articles de recherche, comme Probado [Tho+12], Vocalsearch [Par+08][6]ou Modulo7 [San16].

Un moteur de recherche ne serait pas très utile sans un accès efficace aux résultats : les plus intéressants doivent apparaître en premier, les moins pertinents être relégués pour une consultation ultérieure éventuelle. On doit donc disposer d'un *classement* en fonction de la *pertinence* des documents vis-à-vis de la requête. Pour classer des documents, il faut savoir identifier ceux qui sont les plus proches, donc les plus *similaires* à une requête donnée. Dans la communauté MIR, de nombreuses études ont été consacrées à la similarité musicale au cours des dernières décennies [Ber+04 ; AP02 ; Zha+09 ; Bog+11], et un peu plus récemment [Cas+08 ; Bel11 ; Sil+16]. Cependant, trouver une « bonne » mesure de similarité reste une tâche difficile, car les similarités perçues dépendent fortement des goûts individuels, de la culture et de l'expérience de l'utilisateur [JDE07 ; ERP17].

1.3 Positionnement et contributions

Mes travaux en recherche d'information musicale ont été influencés par mon parcours antérieur et celui de mes collègues, venant des bases de données et de la recherche d'information. J'ai eu une approche originale, consistant à rester dans un sous-domaine spécifique, celui de la musique en notation *symbolique*.

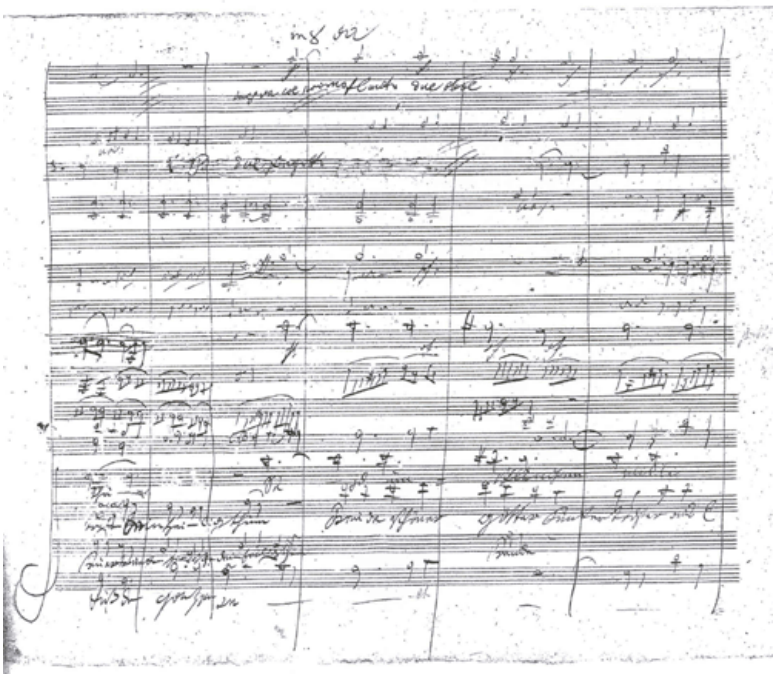


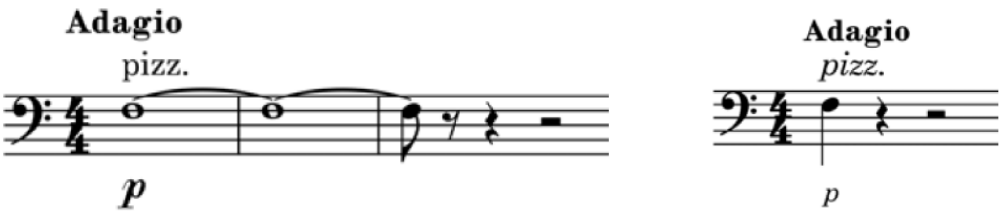
Figure 1.5 : *Partition autographe de la Symphonie n° 9 en ré mineur, op. 125 (4e mvt.), de Ludwig van Beethoven.*

[6]Vocalsearch propose plusieurs modalités de recherche, symbolique et non symbolique, dont le « Query-by-humming », c'est-à-dire de rechercher en fredonnant. C'est un champ de recherche actif côté audio, on pourra consulter [KM11] pour une synthèse sur le sujet.

On aborde la musique par une représentation intermédiaire, la partition musicale. C'est un objet culturel, un document qui sert à communiquer les caractéristiques musicales du son à (re)produire, pour un lecteur ou un interprète, quelques jours ou quelques centaines d'années après la composition. Comme le dit justement et joliment Jawher Matmati, « une partition de musique est la projection d'un phénomène sonore temporel sur un support physique et spatial : la page » [Mat22].

Sans faire ici une histoire exhaustive de la partition musicale, nous pouvons dire que les versions modernes sont le fruit d'une histoire complexe, longue, multiculturelle, qui a suivi les évolutions des techniques d'impression et les techniques de notation musicales. L'épitaphe de Seikilos est le plus ancien objet physique porteur de notation musicale, datant du I^e^ou IIe siècle : une « chanson » était gravée sur une colonne de marbre placée sur la tombe qu'a fait ériger un certain Seikilos pour son épouse ou son père, près de Tralles (Turquie). Plus tard, au IXe siècle, pour accompagner les mélodies grégoriennes (chant liturgique), des signes (les *neumes*) sont apparus au côté du texte chanté, comme une sorte d'aide mémoire de hauteur et de rythme. À partir du XIIe siècle environ, des carrés et des losanges font leur apparition, pour indiquer des rapports de temporalité : ce sont les ancêtres des têtes de notes modernes. Dans les siècles suivants, cette représentation se *structure*, se *systématise*, le code devient convention.

Une partition musicale est une interface, comme le livre est une interface entre le lecteur et l'auteur. Elle se compose d'un ensemble de signes, des *signifiants*, qui permettent, à celles et ceux capables de les *déchiffrer*, d'accéder au *signifié*, au sens, au *contenu musical*. Ces signes, dont nous avons présenté quelques éléments dans la section 1.2.1, sont organisés sur la page en vue, notamment, de faciliter la compréhension et l'interprétation. Comme pour la typographie et l'impression de textes littéraires, on peut distinguer la « *notation* musicale » de la « *gravure* musicale » : « une musique peut être mal notée mais bien gravée et vice versa » [Mat22] (voir la figure 1.6). En poursuivant l'analogie entre la partition et le texte imprimé, une erreur de notation correspond à une faute de grammaire ou d'orthographe d'un auteur, alors qu'une erreur de gravure est une erreur de représentation, un signe mal formé ou mal positionné. Les éditeurs numériques de partitions modernes permettent d'éviter une partie de ces erreurs de gravure.



(a) *Fragment musical mal noté.* **(b)** *Fragment musical mal gravé.*

Figure 1.6 : *Une musique mal notée (a), mal gravée (b). Exemples créés et publiés par [Mat22].*
(a) La gravure ne contient pas d'erreurs. La valeur rythmique contredit le « pizz. » (pizzicato), il faudrait modifier l'un ou l'autre pour que cet extrait devienne bien noté.
(b) L'indication « pizz. » est écrite en italique qui est destiné aux expressions et non aux instructions de technique. La longueur de la hampe dépasse une octave. La nuance est à la fois, mal centrée et écrite en texte « p » et non en typographie musicale appropriée **p**.

Cette *forme* du document musical est très importante : il faut s'assurer que le lecteur pourra en comprendre le sens, le plus immédiatement possible pour être proche de la pensée de l'auteur. Mais

il faut également éviter une notation inutilement complexe ou inhabituelle. Dans l'introduction de sa « bible » de la notation (« Behind Bars » [Gou11]), Elaine Gould cite une lettre de Gustav Mahler se lamentant auprès de sa femme d'une partition mal éditée, qui lui fait perdre « des heures et des heures » [Mah05].

On devrait pouvoir jouer une œuvre à partir de sa partition, mais ce n'est donc pas si simple en pratique : les biais conventionnels culturels et historiques sont nombreux et, comme toute représentation, toute médiation, la partition est source de *perte d'information*. En revanche, et c'est tout son intérêt, cette représentation apporte, immédiatement, de très nombreuses informations que d'autres représentations (comme l'enregistrement audio) rendent plus complexes, voire impossible, à distinguer (les voix, la métrique, les instruments, etc.), comme nous l'avons dit dans la section 1.2.

La séparation stricte du contenu musical, de la structure de l'œuvre et des éléments de rendu graphique est un sujet de discussion musicologique de longue date (voir par exemple [Pug+12]) : les éléments qui participent à la présentation peuvent, dans une certaine mesure, être considérés comme faisant partie intégrante du message transmis par la notation. Pourtant, être en mesure de modéliser séparément le contenu musical permet de considérer une collection de partitions comme un ensemble d'objets structurés, ce qui présente de nombreux avantages. Avec une structure claire, on peut manipuler efficacement des partitions : les découper, appliquer des fonctions à certains éléments (de transposition, d'analyse harmonique, par exemple), fusionner des portions. On peut également aligner des partitions pour les comparer, effectuer des recherches fines. À plus haut niveau, on peut aussi espérer rendre compatible le langage musical avec des outils standards de gestion de données (comme les SGBDR[7]), qui disposent *déjà* de modèles de données stables et puissants, ainsi que d'une indépendance forte entre les modèles logique et physique.

Afin de pouvoir accéder à cette modélisation du contenu musical, il faut donc le défaire des éléments de forme. On doit chercher une certaine *minimalité* de l'information : on souhaite conserver tout ce qui est indispensable pour la production du son, mais garder le moins possible d'information supplémentaire. Les représentations symboliques traditionnelles comme MEI ou MusicXML, que nous avons présenté dans la section 1.2.2, prennent en compte le contexte graphique de la notation [Pug+12], ce qui oriente leur présentation du contenu et limite d'autant les possibilités de manipulation : les choix d'organisation des arborescences XML compliquent des opérations aussi courantes et simples qu'« extraire une séquence de notes ».

Mes travaux se sont donc focalisés principalement sur trois aspects du traitement des données musicales symboliques : leur **modélisation**, leur **interrogation** et leur **transcription**.

Nous avons d'abord entrepris une étude sur plusieurs années de la représentation du contenu musical. Partant d'un constat de spécialiste de base de données («Querying XML Score Databases: XQuery is not Enough! » [FRT16d ; FRT17]), nous avons proposé une modélisation reposant sur des séries temporelles [FRT16c] et étudié en détail une manière d'avoir un modèle de données à partir du contenu musical [FRT16b].

Si ce travail était indépendant de l'*interrogation* de bases de données musicales, cette dernière a tout de même orienté certains des choix que nous faisions. Nous avons commencé par examiner

[7]SGBDR : Systèmes de gestion de bases de données relationnelles.

la possibilité de transférer le concept d'algèbre, crucial en bases de données, vers les données musicales [FRT16c ; FRT16e], en proposant déjà une réflexion sur la constitution de bases de données musicales reposant sur un format spécifique, MEI [FRT16a]. Nos travaux ont convergé pour aboutir à une représentation arborescente du contenu musical que l'on peut *sérialiser*, ce qui permet de l'intégrer efficacement dans un moteur de recherche standard [Zhu+22].

Mon dernier axe de recherche, la transcription automatique de partitions, concerne la *production* de partitions. Il s'agit d'organiser la création de données *de bonne qualité*, en général un préalable à des travaux d'analyse (et de modélisation), un enjeu en *amont* des précédents. Produire des partitions est en général un processus très coûteux, qui repose sur le travail manuel, pour la composition (l'encodage du contenu musical) comme pour le rendu graphique (la gravure), que ce soit en vue de créer des partitions imprimées sur papier ou lisibles sur écran. Nous avons proposé des algorithmes et des outils pour identifier les différences entre des partitions musicales, ce qui a des applications pour l'analyse musicologique mais aussi pour repérer des erreurs humaines de saisie [FFJ19a ; FFJ19b].

Chapitre 2

Représenter la musique notée

À quoi tu penses ?

Je pense que je ne saurais pas expliquer pourquoi on prononce « orkestre » et pas « kef d'orkestre ».

Hervé Le Tellier[1]

Sommaire

2.1 Modéliser le contenu musical **22**
2.1.1 Modèle de données 22
2.1.2 Grammaires et descripteurs de contenu 25
2.2 Une algèbre de partitions **27**
2.3 Conclusion **30**

Dans ce chapitre, je présente les contributions que j'ai effectuées en représentation structurées de données musicales[2]. Notre approche était tirée de la conception des bases de données, et nous avons cherché à élaborer un *modèle de données* du contenu musical, puis à extraire ce modèle de données des documents musicaux, afin d'en permettre la manipulation. Nous l'avons abordé dans le chapitre précédent, les formats de fichiers et autres représentations pré-existantes, conçues pour des objectifs précis tels que le stockage, ne se prêtent pas à des manipulations complexes.

La Figure 2.1 illustre notre vision :

- la couche basse est une bibliothèque de documents musicaux, dans différents encodages (audio, image, XML).

[1] Hervé Le Tellier. *Les amnésiques n'ont rien vécu d'inoubliable.* Bègles : Le Castor Astral, 2005.
[2] Ces travaux ont donné lieu à plusieurs publications : [FRT17 ; FRT16d ; FRT16e ; FRT16b ; Zhu+22].

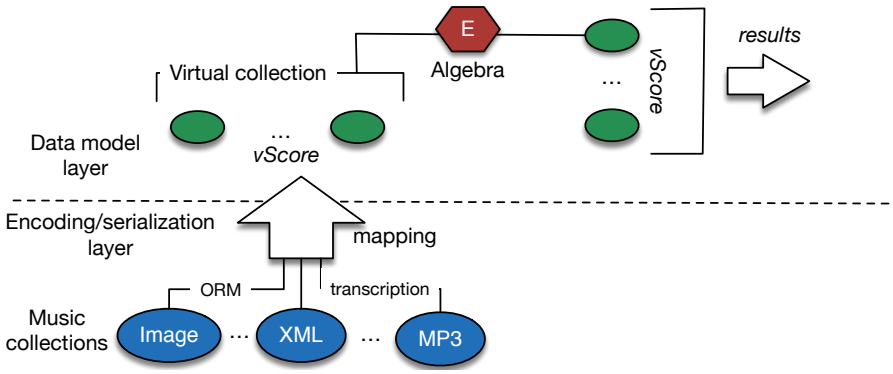


Figure 2.1 : *Schématisation des processus de modélisation.*

- au-dessus, nous faisons correspondre ces documents avec une couche de modélisation dans laquelle le contenu est extrait et structuré. On obtient une abstraction virtualisée (sans matérialisation) des objets musicaux, que nous appelons *vScore*
- une couche haute manipule les représentations des données et les opérations sur celles-ci. Un langage de requêtes avec un ensemble de transformations peut produire d'autres *vScore*(s).

2.1 Modéliser le contenu musical

Après des travaux préliminaires autour de l'idée de représenter le contenu musical comme des séries temporelles, nous avons abouti à un *modèle de données*. Un modèle de données spécifie la nature des éléments contenus dans les données ainsi que les relations qu'ils entretiennent. Nous en présentons ci-dessous les éléments principaux, en illustrant les concept avec le morceau de la Figure 2.2.

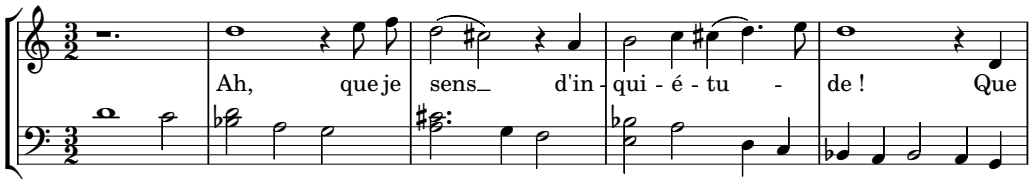


Figure 2.2 : *Air français polyphonique [Des05].*

2.1.1 Modèle de données

Notre modèle de données est centré sur les éléments qui font consensus dans les représentations et encodages de la musique (*cf.* section 1.2.2) : des voix et des événements (musicaux, de durée non nulle).

Valeurs. Notre modèle se compose d'ensembles de constantes, réparties en domaines. Le premier domaine, `Mus`, est celui des sons, identifiés par leur fréquence. Comme indiqué précédemment, il

y a un peu moins de 100 fréquences (notes de A à G dans 8 octaves), avec les altérations éventuelles. Un son peut être représenté par une ou plusieurs notes consécutives, qui peuvent être liées (graphiquement, par un arc).

Définition 1. *Le domaine* Mus *est composé de :*

1. *les symboles de fréquence* $P[a]I$, $P \in \{A, B, C, D, E, F, G\}, a \in \{\sharp, \flat\}, I \in [1, 7]$,
2. *le symbole de pause* r,
3. *un symbole de continuation* _ .

La voix de basse de la Figure 2.2 commence par un D4 et un C4, des sons « atomiques ». Il y a ensuite une combinaison de deux sons, un B♭3 et un D4, en un accord. On définit dsound comme l'union Mus, Mus^2, Mus^3, $\cdots$, Mus^i, $i \in \mathbb{N}$, et on note $< B\flat 3, D4 > \in \mathsf{dpitch}^2$. On dote ce domaine de deux opérateurs :

- l'opérateur Ξ, qui crée un accord à partir de deux notes : $B\flat 3 \; \Xi \; D4 =< B\flat 3, D4 >$
- l'opérateur de transposition $\updownarrow$, qui change la fréquence d'un nombre donné de demi-tons. Par exemple, $C4 \updownarrow 2 = D4$.

Pour compléter le domaine dsound, on a besoin d'un autre domaine pour les paroles, que l'on appelle dsyll, formé de l'ensemble des syllabes. Le seul opérateur dont il est muni est celui de concaténation, $||$. On peut aussi ajouter des domaines arbitraires (par exemple pour intégrer des résultats d'analyse musicologique).

Événements. Un événement e est une valeur v d'un domaine dom pour un intervalle de temps $[t_1, t_2[$.

Définition 2. *Soit* dom *un domaine. Un* événement $e = a_{t_1}^{t_2}, a \in \mathsf{dom}, \mathsf{t}_1, \mathsf{t}_2 \in \mathcal{T}, \mathsf{t}_1 < \mathsf{t}_2$, *représente le fait que la valeur* a *est observée de* t_1 *(inclus) à* t_2 *(exclu). L'intervalle* $[t_1, t_2[$ *est la* durée *de l'événement, et on note* $\mathcal{E}(\mathsf{dom})$ *l'ensemble des événements sur* dom.

Quelques exemples d'événements, avec la Figure 2.2 :

- $D5_{12}^{20}$ est un événement de dsound représentant la première note de la première voix. La valeur est un $D5 \in \mathsf{dsound}$, de t_{12} (début de la seconde mesure) à t_{20}.
- 'Ah' $_{12}^{20}$ est un événement syllabique, de valeur $Ah \in \mathsf{dsyll}$, sur la même période que l'autre exemple.
- $< B\flat 3, D4 >_{12}^{16}$ est un événement de dsound (un accord), de [12,16[.

Voix. Les événements sont regroupés en séries temporelles que l'on appelle des voix. Elles constituent la structure centrale dans la description d'un morceau de musique.

Définition 3. *Une voix* v, *de type* ***Voice(dom)****, est une fonction de* $\mathcal{T}$ *vers* $E_v \subset \mathcal{E}(\mathsf{dom})$ *telle que*

$$a_{t_1}^{t_2} \in E_v \Leftrightarrow v(t) = a_{t_1}^{t_2}, \forall t \in [t_1, t_2[.$$

Cette définition exprime la contrainte implicite de non-recouvrement : à chaque moment t, il y a *au plus* un événement e tel que $v(t) = e$. Mais la fonction est partielle, il peut exister des instants t tels que $v(t)$ est non défini $v(t) = \emptyset$. L'ensemble des intervalles entre les événements de E_v définissent une partition temporelle du temps $\mathcal{T}$, qu'on note $\mathcal{P}(v)$ et que l'on appelle le *domaine temporel actif* de v.

$$v_{sopr}(t) = \begin{cases} \emptyset, & t \in [0,12[\\ D5^{20}_{12}, & t \in [12,20[\\ \emptyset, & t \in [20,22[\\ E5^{23}_{22}, & t \in [22,23[\\ F5^{24}_{23}, & t \in [23,24[\\ D5^{28}_{24}, & t \in [24,28[\\ C\sharp 5^{32}_{28}, & t \in [28,32[\\ \emptyset, & t \in [32,34[\\ A4^{36}_{34}, & t \in [34,36[\end{cases} \quad v_{lyrics}(t) = \begin{cases} \emptyset & t \in [0,12[\\ \text{Ah}^{20}_{12}, & t \in [12,20[\\ \emptyset, & t \in [20,22[\\ \text{que}^{23}_{22}, & t \in [22,23[\\ \text{je}^{24}_{23}, & t \in [23,24[\\ \text{sens}^{32}_{24}, & t \in [24,32[\\ \emptyset, & t \in [32,34[\\ \text{d'in}^{36}_{34}, & t \in [34,36[\end{cases} \quad v_{bass}(t) = \begin{cases} D4^{8}_{0}, & t \in [0,8[\\ C4^{12}_{8}, & t \in [8,12[\\ < B\flat 3, D4 >^{16}_{12}, & t \in [12,16[\\ A3^{20}_{16}, & t \in [16,20[\\ G3^{24}_{20}, & t \in [20,24[\\ < A3, C\sharp 4 >^{30}_{24}, & t \in [24,30[\\ G3^{32}_{30}, & t \in [30,32[\\ F3^{36}_{32}, & t \in [32,36[\end{cases}$$

Figure 2.3 : *Les voix de notre exemple, vues comme des séries temporelles (mesures 1 à 3).*

Temps. Le domaine temporel $\mathcal{T}$ est un ensemble discret, dénombrable, ordonné et isomorphe à $\mathbb{Q}$. En musique, un temps a un nombre maximal de subdivisions possibles : dans l'exemple de la Figure 2.2, la signature (3/2) indique que le temps dure une blanche avec 3 temps par mesure. Le temps se divise en 4 au plus, donc l'unité est la croche, et chaque mesure dure 12 unités. On définit une partition temporelle $\mathcal{P} = \{I_1, \cdots, I_n\}$ comme un ensemble d'intervalles ouverts à droite $I_i = [t^i_1, t^i_2[$ tels que $\forall i \neq j \leq n, I_i \cap I_j = \emptyset$. On n'impose pas ici de condition de couverture de l'espace temporel, car les morceaux de musiques contiennent des silences (pause, soupir). Chacune des voix est une partition temporelle. Pour combiner plusieurs voix, on définit la notion de partitions fusionnables :

Définition 4. *Deux partitions temporelles $\mathcal{P}_1 = \{I_1, \cdots, I_n\}$ et $\mathcal{P}_2 = \{J_1, \cdots, J_m\}$ sont* fusionnables *ssi leur union est une partition temporelle, i.e., $\forall i \in [1,n], j \in [1,m]$, soit $I_i \cap J_j = \emptyset$ soit $I_i = J_j$.*

Dans la figure 2.4 (à gauche), $P_1 = \{[0,2[, [4,6[, [8,9[\}$ et $P_2 = \{[2,4[, [6,7[, [8,9[\}$ sont fusionnables. Leur union : $P_1 \cup P_2 = \{[0,2[, [2,4[, [4,6[, [6,7[, [8,9[\}$ est une partition temporelle dont chaque intervalle est dans P_1 ou P_2. En revanche, comme certains intervalles se recouvrent ou sont contenus les uns dans les autres, P_3 et P_4 ne sont pas fusionnables.

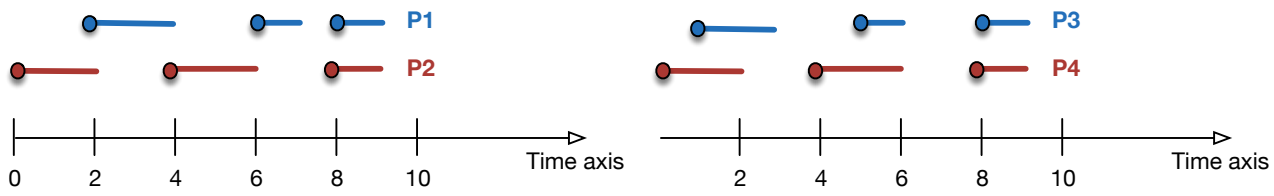


Figure 2.4 : *Partitions fusionnables (gauche) et non fusionnables (droite).*

Partitions. Les voix peuvent être synchronisées. La synchronisation de v_1 de **Voice(dom$_1$)** et de v_2 de **Voice(dom$_2$)**, notée $v_1 \oplus v_2$, est la voix v_3 de **Voice(dom$_1$ $\times$ dom$_2$)**, telle que $\forall t \in \mathcal{T}, e_1 \in \mathcal{E}(\mathsf{dom}_1)$, $e_2 \in \mathcal{E}(\mathsf{dom}_2)$, on a : $v_3(t) = (e_1, e_2) \Leftrightarrow v_1(t) = e_1$ *et* $v_2(t) = e_2$.

On définit ensuite les partitions (abstraites), comme des synchronisations de voix.

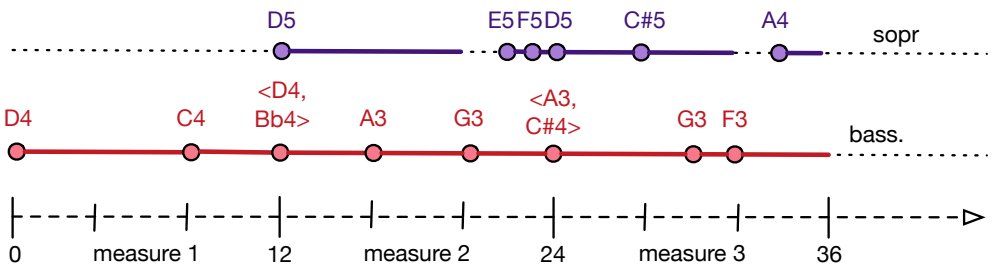


Figure 2.5 : *Synchronisation de voix (mesures 1-3)*

Définition 5. *Une partition (*a score*, en anglais) est un arbre, défini récursivement comme suit :*

- *une voix est une partition,*
- *si $s_1, \cdots, s_n$ sont des partitions, alors $s_1 \oplus \cdots \oplus s_n$ est une partition.*

La partition S de notre exemple (Figure 2.2) est un arbre tel que :

- la partie vocale s_{vocal} résulte de la synchronisation de v_{sopr} et v_{lyrics}, i.e., $s_{vocal} = v_{sopr} \oplus v_{lyrics}$;
- la partition S est la synchronisation de s_{vocal} et v_{bass}, i.e, $S = s_{vocal} \oplus v_{bass}$.

Les partitions sont *typées*, selon les types qui les composent :

- le type T_v de s_{vocal} est [`sopr` : Voix(`dsound`), `lyrics` : Voix(`dsyll`)]
- le type T_b de S est [`vocal` : T_v, `bass` : Voix(`dsound`)]

Morceau (opus) et corpus. Un morceau (*opus*) est un ensemble de valeurs atomiques (chaînes de caractères, entiers, etc.) ou composites (partitions). Si l'on revient au modèle relationnel des bases de données, un corpus est un conteneur d'objets de même structure, comme une table, et les opus sont les éléments de cette table (des lignes).

2.1.2 Grammaires et descripteurs de contenu

Maintenant que nous disposons d'un modèle de données, nous proposons de lui adjoindre une représentation des documents musicaux. Comme une partition est découpée en intervalles temporels, nous suivons cette organisation intrinsèquement hiérarchique et travaillons avec une structure d'arbre. Nous obtenons *in fine* des *descripteurs de contenu musicaux.*

Grammaires rythmiques. Le rythme d'une partition et ses règles peuvent s'exprimer dans un langage formel, les *Context-Free Grammars*, éventuellement pondérées (*Weighted Context-Free Grammars*) [FFJ19a]. Une grammaire simple suffit à modéliser l'exemple de la Figure 2.6. La grammaire $\mathcal{G} = (V, \texttt{Mus}, R, S)$ est définie avec un ensemble $V = \{S, m, b, q\}$ de symboles non-terminaux, où S est une partition, m une mesure, b un temps (*beat*) et q une croche (*quaver*). Les symboles terminaux sont ceux de `Mus` et R est l'ensemble des règles suivantes :

1. $R_0 : S \rightarrow m|m, S$ (un morceau est une suite de mesures)

2. $r_1 : m \to b, b, b, b$ (une mesure se décompose en 4 temps)

3. $r_2 : b \to q, q$ (un temps se décompose en 2 croches)

4. un ensemble $\mathcal{R}^m$ de règles $R_e^v : v \to e$ telles que $e \in \mathtt{Mus}$ soit un symbole.

La règle R_0 et l'ensemble $\mathcal{R}^m$ déterminent la structure (rythmique) d'un morceau, et forment une grammaire du contenu musical.

Descripteur de contenu musical. À partir de ces grammaires, on peut extraire une représentation arborescente du morceau (un *parse tree*), ce qui nous permet de définir les descripteurs de contenu musicaux et la notion de *séquences de notes.*

Nous illustrons cela avec la Figure 2.6, le début de l'hymne allemand, *Das Lied der Deutschen*, dont la musique a été composée par Joseph Haydn, en 1797.

Définition 6. *Soit $\mathcal{G} = (V, \mathtt{Mus}, R, S)$ une grammaire du contenu musical. Un descripteur monodique du contenu musical est le* parse tree *de G. Les nœuds internes constituent l'arbre de rythme, et les feuilles sont les événements.*

Définition 7. *Soit D un descripteur de contenu. Les valeurs des feuilles forment une suite d'éléments de $\mathtt{Mus}^*$ que l'on appelle séquence de notes de D, on la note $PSeq(D)$.*

Avec la modélisation de la section précédente et cette définition, on définit une partition temporelle.

Définition 8. *Soit $I = [\alpha, \beta[$ un intervalle de temps, et D un descripteur de contenu. La* partition temporelle *$P(I, D)$ de I par rapport à D est définie comme suit. Soit N un nœud de l'arbre de rythme de D.*

1. *Si N n'a pas d'enfants, $P(I, N) = \{I\}$*

2. *Si N est de la forme $N(N_1, \cdots, N_i)$, I est partitionné en n intervalles de taille identiques $s = \frac{\beta-\alpha}{n}$ chacun $P(I, N) = \{I_1, \cdots, I_n\}$ avec $I_i = [\alpha + (i-1) \times s, \alpha + i \times s]$*

Cette partition associe à chaque nœud interne N un intervalle non nul $itv(I, N)$ et une durée $dur(I, N)$. Chaque événement (feuille) couvre l'intervalle de son parent dans l'arbre de rythme.

On adopte la convention suivante : la durée d'une mesure est 1, et un morceau dure n (comme le nombre de mesures). Les durées et intervalles d'un nœud résulte de sa définition par les règles de division ($\frac{1}{2}$ pour une blanche, etc.). On termine par définir la séquence d'événements :

Définition 9. *Soit D un descripteur de contenu, et $[L_1, \cdots, L_n]$ la séquence de notes de D. La séquence $[(L_1, dur(L_1)), \cdots, (L_n, dur(L_n))]$ où on associe à chaque feuille sa durée est appelée* séquence d'événements *de D, notée $ESeq(D)$. Chaque élément de $ESeq(D)$ associe donc un symbole de $\mathtt{Mus}$ à une durée.*

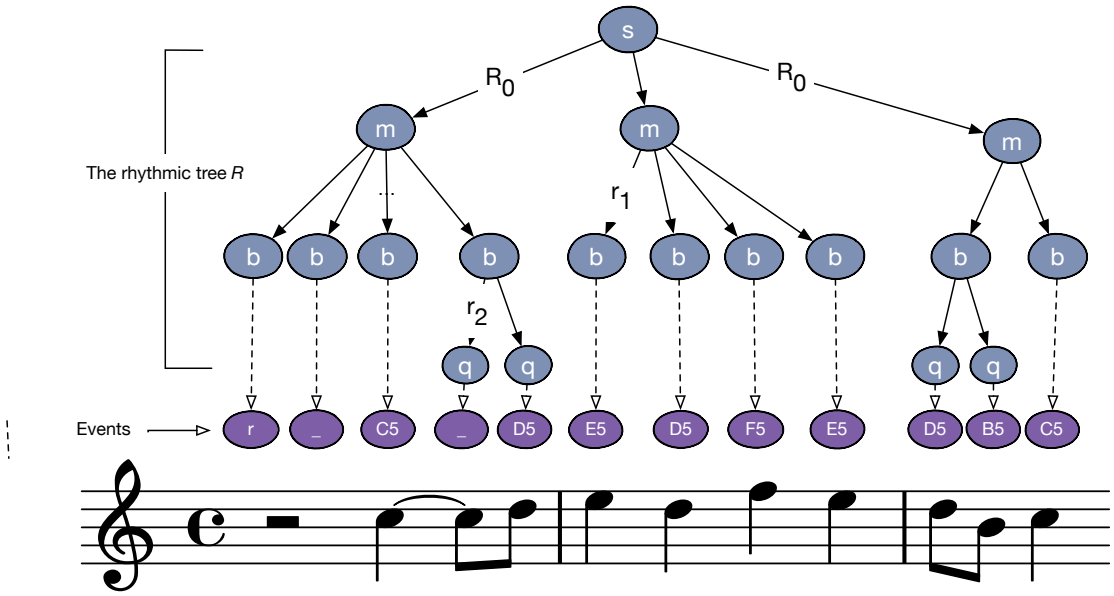


Figure 2.6 : *Le descripteur de contenu de l'hymne allemand, avec ses événements et son arbre de rythme.*

Notre modèle souffre, bien sûr, de quelques limites : on perd parfois une partie du sens en laissant de côté certains éléments de la partition, et quelques caractéristiques musicales rares ne sont pas correctement capturées (par exemple : des accords avec des durées de notes variables). Cependant, nous avons enrichi une représentation relativement classique en musique avec une structure d'arbre. Celle-ci permet d'effectuer diverses simplifications, calculer des mesures de similarité ou déduire des temps forts ou faibles à partir de leur position. Plus généralement, ce *framework* permet d'inférer des *caractéristiques* à partir de descripteurs de contenu en extrayant, transformant, normalisant des aspects spécifiques relatifs au rythme et/ou aux valeurs.

2.2 Une algèbre de partitions

À l'aide de la modélisation que nous avons élaboré ci-dessus, nous proposons une algèbre. Comme nous l'avons indiqué dans le chapitre 1, interagir avec le contenu musical d'une partition est un besoin pour l'analyse musicologique (extraire, combiner des fragments, par exemple), et ouvre aussi la voie à de nombreuses applications.

Notre algèbre prend en entrée des partitions virtuelles (*vScore*, pour *virtual score*) et produit des partitions en sortie, opérant en forme close. Cette algèbre dispose également de la possibilité d'appliquer des fonctions arbitraires. Elle se compose de trois parties : un ensemble de fonctions, des opérateurs, et une extension limitée de l'algèbre relationnelle.

Fonctions

Nous définissons deux types de fonctions, les fonctions temporelles et les fonctions *du domaine* :

- une fonction *temporelle* est une application de $\mathcal{T}$ vers $\mathcal{T}$. On peut s'en tenir aux fonctions linéaires telles que $\tau_{m,n}, m \neq 0$ de la forme : $\tau_{m,n}(t) = mt + n$. En particulier, les fonctions de déformation temporelle (*time warping*) sont utiles, pour étendre ou réduire les durées des événements. Elles sont de la forme $\text{warp}_\text{m} : \mathcal{T} \to \mathcal{T}, t \mapsto mt$. Les fonctions de décalage (*shifting*) de la forme $\text{shift}_\text{n} : \mathcal{T} \to \mathcal{T}, t \mapsto t + n$. peuvent effectuer des translations dans le temps.
- une fonction *de domaine* associe une valeur de $\mathsf{dom}_1 \times \cdots \times \mathsf{dom}_\text{n}$ à une valeur d'un domaine dom_o. Ces fonctions s'appliquent aux événements. Si $e = a_{t_1}^{t_2}$ est un événement de dom, alors
  1. (fonction temporelle) $\tau_{m,n}(e) = a_{\tau_{m,n}(t_1)}^{\tau_{m,n}(t_2)}$
  2. (fonction de domaine) si f est une fonction de domaine sur dom, alors $f(e) = f(a)_{t_1}^{t_2}$

Des fonctions externes (*user-defined fonctions* ou UDF) peuvent également être définies. Une fonction *interval()* par exemple, pourrait prendre en entrée deux sons et retourner un entier de `dint`.

Opérateurs

Outre les fonctions, cette algèbre contient cinq opérateurs : de synchronisation, projection, selection, fusion et un opérateur de *map* pour appliquer les fonctions externes. Ces opérateurs font référence à ceux de l'algèbre relationnelle. Chacun d'entre eux prend une ou deux partitions et en produit une en sortie.

Définition 10 (Synchronisation, $\oplus$)**.** *Si S_1 et S_2 sont deux partitions, alors $S_1 \oplus S_2$ est une partition définie par :* $[S_1 \oplus S_2](t) = (S_1(t), S_2(t)), \forall t \in \mathcal{T}$.

L'opérateur de projection μ est le même que celui de l'algèbre relationnelle, extrayant une ou plusieurs voix d'une partition.

Définition 11 (Projection, μ)**.** *Si S est une partition de type $[v_1 : \mathsf{dom}_1, \cdots, v_n : \mathsf{dom}_\text{n}]$, alors $\mu_{v_{i_1},\cdots,v_{i_m}}(S), \forall i_j, j \in [1, n]$ est une partition définie comme :*

$$[\mu_{v_{i_1},\cdots,v_{i_m}}(S)](t) = (S.v_{i_1}(t), \cdots S.v_{i_m}(t))$$

L'opérateur de sélection σ_F ne change pas les événements d'une partition qui satisfont une condition, et remplace les autres par $\emptyset$.

Définition 12 (Sélection, σ)**.** *Si S est une partition de type $T = [v_1 : \mathsf{dom}_1, \cdots, v_n : \mathsf{dom}_\text{n}]$ et F une expression booléenne sur $\mathcal{T}, \mathsf{dom}_1, \cdots, \mathsf{dom}_\text{n}$, alors $\sigma_F(S)$ est une partition de type T telle que pour chaque voix $S.v_i$:*

$$[\sigma_F(S)].v_i(t) = \begin{cases} S.v_i(t), \textit{si} F(S.v_i(t)) = \text{vrai} \\ \emptyset, \qquad\qquad \textit{sinon} \end{cases}$$

L'opérateur de fusion, $\circ$, fusionne deux à deux les voix de deux partitions de même type, à condition, évidemment, qu'elles soient fusionnables. Cela permet en particulier d'aligner séquentiellement deux partitions.

Définition 13 (Fusion, $\circ$). *Si S_1 et S_2 sont deux partitions de type $T = [v_1 : \mathsf{dom_1}, \cdots, v_n : \mathsf{dom_n}]$ tel que le domaine temporel actif $\mathcal{P}(S_1.v_i)$ est fusionnable avec $\mathcal{P}(S_2.v_i), \forall i \in [1, n]$, alors $S_1 \circ S_2$ est une partition T définie comme :*

$$[S_1 \circ S_2].v_i(t) = \begin{cases} S_1.v_i(t), si\ S_1.v_i(t) \neq \emptyset, S_2.v_i(t) = \emptyset \\ S_2.v_i(t), si\ S_1.v_i(t) = \emptyset, S_2.v_i(t) \neq \emptyset \\ \emptyset, \qquad si\ S_1.v_i(t) = S_2.v_i(t) = \emptyset \\ S_1.v_i(t) \ \Xi\ S_2.v_i(t), \qquad sinon \end{cases}$$

Propriété 1. *Soit v_1 et v_2 deux voix telles que leurs domaines temporels actifs soient fusionnables. Alors, $v_1 \oplus v_2 \equiv v_1 \circ v_2$, où $\equiv$ exprime l'équivalence du contenu musical.*

Enfin, l'opérateur de map() $\textsc{map}_f$ applique une fonction f aux voix d'une partition, retournant une partition mono-voix avec le résultat de f.

Définition 14 (Transformation, MAP). *Si S est une partition de type $[v_1 : \mathsf{dom_1}, \cdots, v_n : \mathsf{dom_n}]$, et f une fonction $\Pi_i^n \mathsf{dom_i} \rightarrow \mathsf{dom_o}$ alors $\textsc{map}_{a:\ f}(S)$ est une fonction de type [a : $\mathsf{dom_o}$] définie par :*

$$[\textsc{map}_{a:\ f}(S)].a(t) = f(S.v_1(t), \cdots S.v_n(t)), \forall t \in \mathcal{T}$$

Expressions algébriques

Les expressions sont ensuite définies de façon *naturelle*, par rapport à ce qui fait en base de données :

- Si S est une partition, alors S est une expression.
- Si E_1 et E_2 sont deux expressions, alors $E_1 \oplus E_2$, $\textsc{map}_{a:\ f}(E_1)$, $E_1 \circ E_2$, $\sigma_F(E_1)$, et $\mu_{v_{i_1}, \cdots, v_{i_m}}(E)$ sont des expressions (sous réserve que soient validées les conditions sur les types des entrées et sorties).

Exemple. Si l'on souhaite extraire, à partir des quartets de Joseph Haydn, le titre et deux partitions, l'une contenant uniquement les voix du premier violon et du violoncelle, l'autre contenant le second violon et l'alto, on peut utiliser l'expression suivante :

$$\Pi_{\text{title,v1}\oplus\text{cello,v2}\oplus\text{alto}}(\sigma_{\text{composer='Haydn'}}(\text{Quartet}))$$

Comme les opérateurs retournent toujours des instances du modèle (des *vScore*), on peut donc les composer pour créer des expressions arbitrairement complexes. On montre que cette algèbre est complète [FRT18].

2.3 Conclusion

Dans ce chapitre, nous avons présenté plusieurs contributions en modélisation des données. La principale est une abstraction complète de la partition musicale, avec une représentation sous forme d'arbre, ce qui permet un accès fin aux éléments musicaux. Nous avons également introduit une algèbre, avec laquelle de nombreuses manipulations, similaires à celles des bases de données sont possibles pour les partitions (projection, sélection, fusion, etc.). Avec ces travaux de modélisation de données, nous avons ouvert la voie à des manipulations complexes de données, permettant d'extraire des informations diverses, de conserver des résultats de traitements analytiques, d'interroger les données, ce que nous verrons dans le chapitre suivant.

Les perspectives autour de ces travaux concernent surtout la validation et l'adoption de notre modélisation : si nous montrons dans les chapitres suivants qu'elle permet des traitements complexes et divers, il serait intéressant qu'elle soit reprise, améliorée peut-être, et intégrée à d'autres travaux et systèmes, tels que des bases de données dédiées à la musique (ou au moins à la musicologie).

Chapitre 3

Rechercher de l'information

I'm no expert, but I remember reading somewhere, every time you retrieve a memory, that act of retrieval, it corrupts the memory a little bit. Maybe changes it a little.

Emily St. John Mandel[1]

Sommaire

3.1 Extraire les caractéristiques musicales **32**
3.1.1 Mélodie 33
3.1.2 Rythme et parole 35
3.2 Indexer **36**
3.3 Rechercher **36**
3.4 Classer les résultats **37**
3.4.1 Similarité par blocs de rythme 37
3.4.2 Classement final 39
3.5 Implémentation et interface graphique **39**
3.5.1 Architecture 39
3.5.2 Interface 40
3.6 Conclusion **40**

Les moteurs de recherche sont devenus des éléments indispensables de notre quotidien. Ils nous permettent d'explorer de grandes collections de documents, éventuellement hétérogènes, en proposant des listes classées de résultats correspondant à un besoin d'information exprimé par un ensemble de mot-clefs. Aujourd'hui, pour le texte comme pour les images, on dispose de moteurs efficaces. Pour la musique, et en particulier la musique notée, la complexité de l'information rend

[1] Emily St John Mandel. *The Glass Hotel*. New York : HarperCollins, 2020.

la création d'un moteur de recherche plus délicate : la représentation classique est moins intuitive que le texte, et les aspects temporels (tempo, synchronisation) complexifient aussi notre travail. En outre, il existe une grande variété des documents musicaux, correspondants aux diverses formes de musique : musicien unique ou orchestre symphonique, musique savante ou populaire, improvisation ou forme contrainte, etc. Enfin, les styles ont grandement variés selon les lieux et les périodes de l'histoire.

Nous avons développé un *framework* général et *scalable* de recherche d'information musicale. Celui-ci a pour base un moteur de recherche de contenu musical, mais de nombreuses autres fonctionnalités sont envisageables, autour de ce socle. Ce framework est *général* dans le sens où il est indépendant d'un encodage initial de la musique, pour peu que l'on puisse traduire cet encodage vers la représentation interne de notre framework. Le moteur offre des temps de réponse très rapides, même lorsqu'il s'agit de grandes collections de documents.

L'architecture du framework est illustrée sur la Figure 3.1, qui peut être vue comme une variante de la figure 2.1. On extrait des descripteurs du contenu musical à partir de documents musicaux. Ensuite, des caractéristiques sont extraites de ces descripteurs, comme les étapes préalables à l'indexation quand il s'agit de texte (tokenisation, normalisation, etc.). Celles-ci sont encodées dans un format compatible avec l'indexation, la recherche et le classement dans un moteur de texte. En réponse à une requête contenant un motif musical, on obtiendra une liste ordonnée de documents (avec les détails des positions).

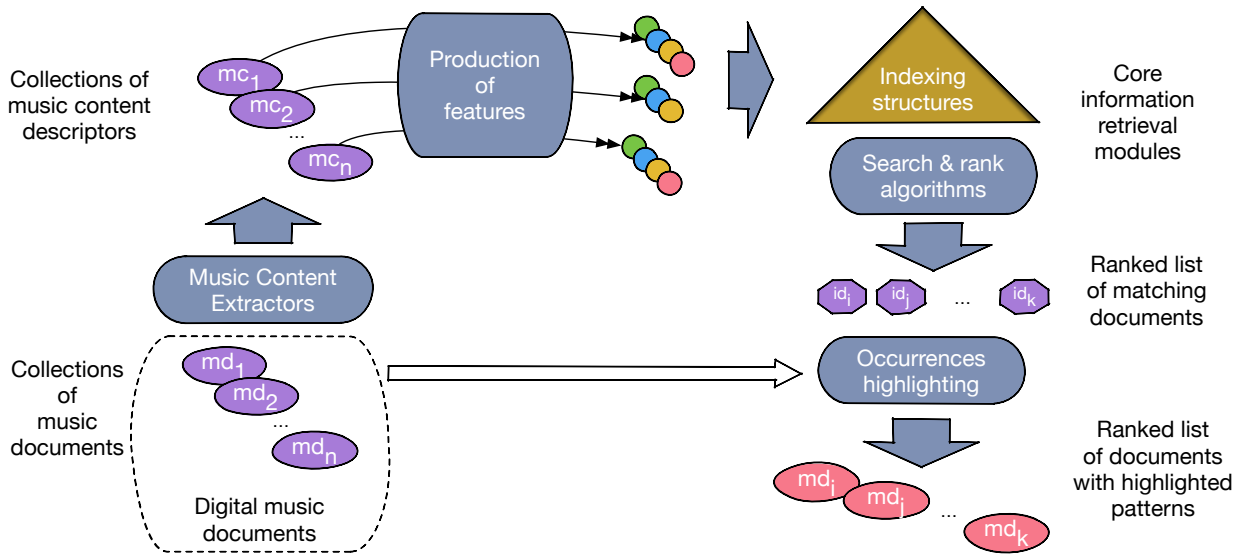


Figure 3.1 : *Illustration de notre architecture pour la recherche d'information musicale.*

3.1 Extraire les caractéristiques musicales

La première étape de la réalisation de notre moteur de recherche consiste à extraire de l'information de documents musicaux, et à proposer une représentation intermédiaire qui soit compatible

avec les moteurs de recherche dédiés au texte, pour éviter de réécrire le cœur du moteur (l'indexation) et bénéficier des avancées des vingts dernières années en recherche d'information *open source* (notamment les logiciels Solr [Apa] et ElasticSearch [Ela]).

Nous présentons l'extraction de quelques caractéristiques[2] . Bien entendu, d'autres caractéristiques pourraient être ajoutées, si les conditions suivantes sont satisfaites :

- on dispose d'un *analyseur* qui prend une partition en entrée et peut représenter cette caractéristique en sortie ;
- on peut *sérialiser* cette caractéristique sous la forme d'une chaîne de caractères, de façon à rendre possible le passage d'une requête musicale à une requête textuelle, comprise par le moteur de recherche ;
- on peut munir cette caractéristique d'une *fonction de score* qui peut s'intégrer au moteur de recherche pour le classement des résultats.

Les illustrations de ce chapitre sont extraites de *My way* [Ank69]. C'est la version anglophone de Paul Anka de *Comme d'habitude* [RF67], écrite par Claude François et Jacques Revaux (1967). Son descripteur de contenu est sur la Figure 3.2, d'après la modélisation vue au chapitre 2.

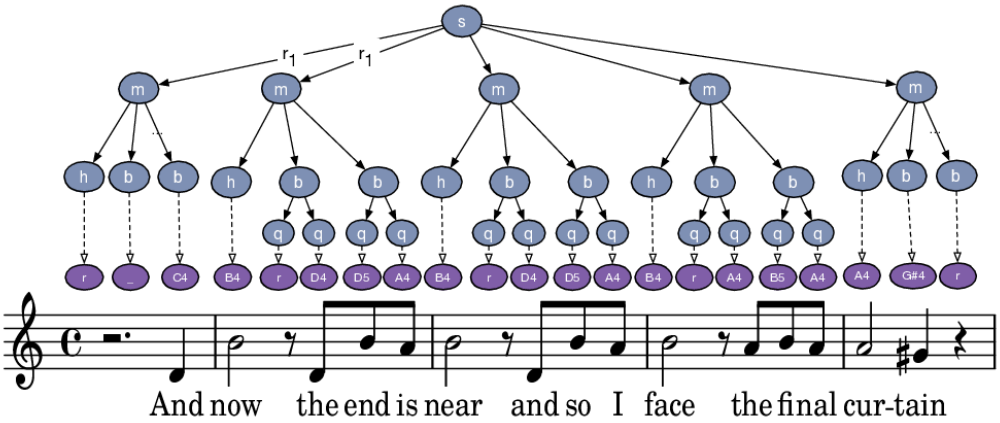


Figure 3.2 : *Descripteur pour* My way.

3.1.1 Mélodie

Pour reconnaître une mélodie, on s'intéresse aux intervalles entre les hauteurs des notes qui la constituent. Deux types d'intervalles sont musicologiquement intéressants, les intervalles chromatiques et les intervalles diatoniques. Pour faciliter la recherche, on simplifie la mélodie en fusionnant toutes les valeurs répétées de $PSeq(D)$, et en enlevant les pauses. On conserve ainsi les intervalles non nuls, comme le montre la Figure 3.3 (deux A4 vers la fin ont été fusionnés, les silences retirés).

Intervalles chromatiques

Un analyseur, noté A_{CIF}, prend en entrée un descripteur de contenu D, et produit en sortie la liste des valeurs des intervalles entre deux notes consécutives dans la version simplifiée de D, le

[2]Le terme *caractéristique* traduit l'anglais *feature*.

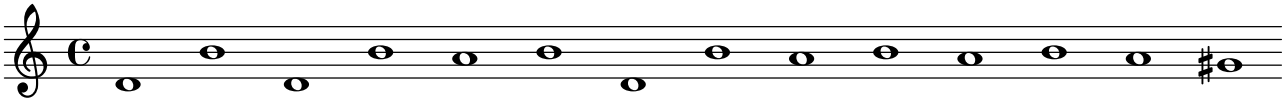


Figure 3.3 : My way*, après extraction de caractéristiques par l'analyseur adapté.*

CIF (pour *Chromatic Interval Feature*). Ainsi, pour la séquence de la figure 3.3, on obtient : CIF $=< 9, -9, 9, -2, 2, -9, 9, -2, 2, -2, 2, -2, -1 >$. Un même résultat identique peut être obtenu à partir d'autres séquences d'entrée : la Figure 3.4 présente une version de My Way plus adaptée à une voix féminine, son CIF est le même.

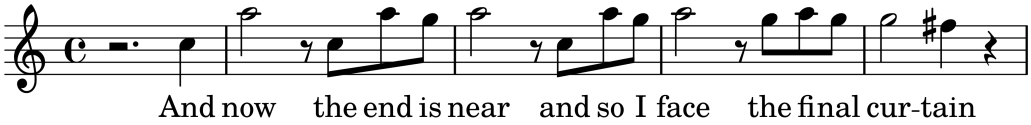


Figure 3.4 : My way*, avec transposition.*

On ne s'intéresse ici qu'aux aspects mélodiques de la partition : des variantes rythmiques peuvent avoir des CIF identiques. Ainsi, dans la version de Claude François, les contraintes linguistiques du français amènent quelques subtiles modifications (voir la Figure 3.5). Mais comme la séquence d'intervalles reste la même, le CIF aussi.

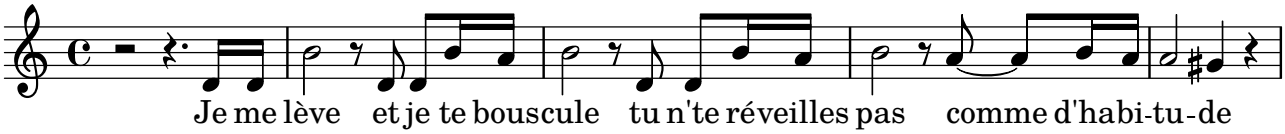


Figure 3.5 : Comme d'habitude*, première phrase (version française de* My way*).*

Les trois morceaux des figures 3.2, 3.4, et 3.5 se correspondent donc : dans un moteur de recherche, prendre l'un comme requête aboutirait aux deux autres parmi les résultats. Cela est cependant très dépendant des choix que nous avons fait pour notre analyseur : si on avait choisi pour CIF les noms des notes, la transposition aurait été exclue des résultats, ce qui aurait conduit à une *précision* meilleure, mais à un *rappel* plus faible, et un résultat sans doute moins intuitif pour l'utilisateur.

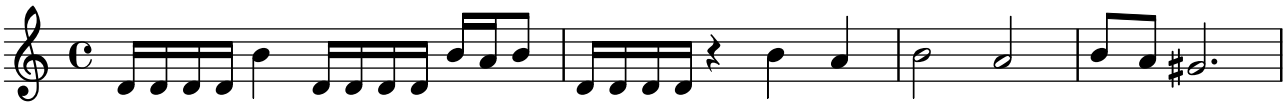


Figure 3.6 : My way*, avec un rythme modifié.*

De même, on peut imaginer des morceaux avec le même CIF mais un rythme *très différent*, comme celui de la Figure 3.6. Si on peut envisager d'incorporer celui-ci dans la même liste de résultats que les autres, il est assez évident qu'on doit avoir une fonction de score qui le classe significativement en dessous des autres, tant il est dissemblable.

Intervalles diatoniques

La Figure 3.7 présente la deuxième phrase de My Way. Son CIF diffère du précédent, il vaut :

$$< 8, -8, 8, -1, 1, -8, 8, -1, 1, 4, -2, -5, 3, -1 > .$$

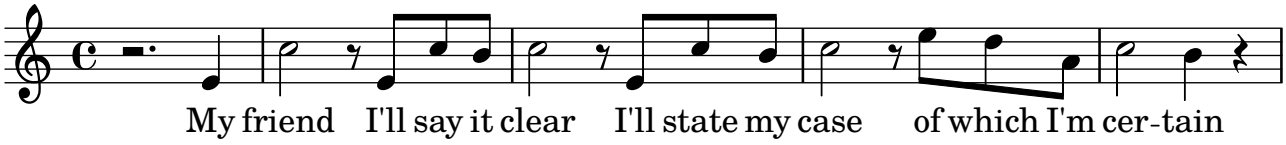


Figure 3.7 : My way, *deuxième phrase.*

Cette phrase débute par un intervalle de 5 degrés diatoniques, de E4 à C5, puis descend d'un degré, de C5 à B4. Dans une perspective diatonique, le premier intervalle de la première phrase et celui de la seconde sont similaires, ce sont tous les deux des sixtes, majeure dans le premier cas, mineure dans le second. C'est aussi le cas des intervalles suivants, des secondes, mineure et majeure respectivement. Pour capturer cet aspect musical, on introduit le DIF (*Diatonic Interval Feature*). Pour l'encoder, on part de l'ensemble des intervalles diatoniques {`U(nison)`, `S(e)c(ond)`, `T(hird)`, `Fo(urth)`, `Fi(fth)`, `Si(xth)`, `Se(venth)`, `O(ctave)`} [3]. On note d'un $+$ quand c'est ascendant et d'un $-$ si c'est descendant. On obtient ainsi le résultat suivant pour la Figure 3.7 :

$$< Si+, Si-, Si+, Sc-, Sc+, Si-, Si+, Sc-, Sc+, Si-, Si+, Si-, Si- > .$$

La première et la seconde phrase de My Way ont le même DIF, et c'est aussi le cas de leurs variantes transposées ou rythmiquement différentes (voir les Figures 3.4 et 3.5).

3.1.2 Rythme et parole

Les caractéristiques mélodiques utilisent les valeurs associées aux feuilles de l'arbre descripteur de contenu. Pour le rythme, c'est la structure de l'arbre qui importe. Une stratégie simple pourrait consister à sérialiser cette représentation, pour obtenir la caractérisation du rythme. Cela se heurte à deux écueils principaux :

1. la perception du rythme est invariante selon certaines transformations : doubler la durée de toutes les notes et le tempo ne change pas la musique jouée
2. l'arbre de rythme est une représentation complexe du rythme du morceau et mettre directement toutes ces informations dans la caractéristique amènerait une grande précision mais diminuerait grandement le rappel

On simplifie donc, comme pour les caractéristiques mélodiques, et ce sera aux fonctions de classement de distinguer les motifs les plus similaires à un motif-requête.

Pour un descripteur de contenu R, sa partition temporelle (*cf.* chapitre 2) indique les durées des événements. La première phrase de *My way* (Figure 3.2) commence par une noire, suivie d'une blanche. Le rapport de leurs durées est 2. Puis, vient une croche, donc un rapport de 4 (avec la blanche), suivie de 2 autres croches, donc des rapports de 1. Ce sont ces rapports qui nous servent pour définir la caractérisation rythmique de notre morceau : RIF $=< 2, \frac{1}{8}, 1, 1, 8, \frac{1}{8}, 1, 1, 8, \frac{1}{8}, 1, 1, 8, \frac{1}{2} >$.

[3] 0 (prime), 1 degré (seconde, C-D[4]), 2 degrés (tierce), quarte, quinte, sixte, septième, octave. Voir Chapitre 2.

3.2 Indexer

Une fois que l'on a extrait les caractéristiques, dans un format textuel, nous les adaptons au format d'un moteur de recherche, en les décomposant en n-grams. Chaque suite de valeurs $[v_1, v_2, \cdots, v_k]$ devient ainsi $\{< v_i, \cdots, v_{i+n-1} >, i \in [1, k-n+1]\}$ où n est la taille des n-gram (on utilise $n = 3$). Pour la séquence <6,-3,-3,1,2,-2>, par exemple, la liste des 3-grams est [<6,-3,-3>, <-3,-3,1>, <-3,1,2>, <1,2,-2>]. Chaque n-gram est ensuite encodé comme une chaîne de caractères, pour constituer un *token*. Ces tokens sont ensuite concaténés en texte, séparés par des espaces. Nous utilisons aussi « m » pour encoder le signe « - » et X comme séparateur, et nous envoyons ainsi au module d'indexation du moteur de recherche la séquence suivante : `6Xm3Xm3 m3Xm3X1 m3X1X2 1X2Xm2`.

Nous avons fait le choix de simplifier les caractéristiques, ce qui favorise le rappel (on récupère davantage de documents, qui correspondent « un peu moins précisément » au motif recherché) par rapport à la précision. Chacune de nos caractéristiques capture une dimension de la musique (mélodie ou rythme), et l'on peut donc avoir des résultats proches sur une dimension mais complètement différents sur l'autre : il faut ensuite organiser les résultats avec une fonction de classement (*scoring*), ce que nous allons voir dans la section suivante.

3.3 Rechercher

La recherche de documents musicaux s'opère de façon similaire au texte : on applique au motif recherché les mêmes opérations d'analyse (les mêmes transformations) que celles qu'on avait appliquées aux documents lors de l'indexation. Le calcul des documents correspondant à la requête est le même que pour le texte, ce qui autorise une parallélisation et donc un passage à l'échelle à de grandes collections. Cependant, les fonctions de classement usuelles pour le texte ne donneraient pas de bons résultats si on les appliquaient telles quelles à nos caractéristiques musicales : nous définissons des fonctions dédiées.

Un *motif de recherche* q (aussi appelé « requête », en anglais *query*) est une paire (P, FT) où P est un descripteur de contenu et FT une caractéristique FT $\in \{(CIF, DIF, RF)\}$.

Définition 15. *Soit $q = (P, FT)$ un motif de recherche, où $FT \in \{CIF, DIF, RF\}$, A_{FT} est l'analyseur associé à FT, D est un descripteur musical. On dit que q correspond (*match*) = D ssi il existe au moins une sous-chaîne F de $A_{FT}(D)$ (un* fragment*) telle que $A_{FT}(P) = F$.*

Par exemple, un utilisateur recherche le motif P de la Figure 3.8 avec la caractéristique CIF, la séquence $A_{CIF}(P)$ est $< 9, -2, 2, -2 >$, qui est une sous-chaîne du CIF des descripteurs des Figures 3.2, 3.5 et 3.6.

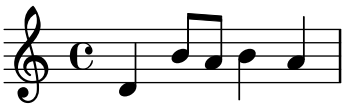


Figure 3.8 : *Un motif, correspondant à un fragment de* My way.

Cette définition s'étend naturellement au cas des polyphonies : un descripteur polyphonique M correspond à une requête $q = (P, FT)$ si et seulement si, pour au moins un descripteur $D \in M$, et au moins une sous-chaîne F de $A_{FT}(D)$, on a $A_{FT}(P) = F$. Les fragments qui correspondent à la requête sont appelés les *occurrences* (du motif).

Au moment de la recherche, le motif est encodé en n-grams et envoyé au moteur, qui trouve les documents contenant ces n-grams. Ce moteur scanne la liste pour chaque n-gram de la requête et trouve les descripteurs qui lui correspondent. Il vérifie ensuite que les positions sont les mêmes que dans le motif, avant d'appliquer la fonction de classement et de finalement grouper les descripteurs par document, pour trouver les meilleures partitions. Ces étapes se font soit au niveau du document (les premières), soit au niveau du serveur : on peut donc paralléliser la recherche, sur des serveurs d'indexation distincts (des *nœuds* pour Elasticsearch) [Zhu+22]. Les performances, en terme de vitesse, sont donc très bonnes.

3.4 Classer les résultats

Il est classique, en recherche d'information, d'ordonner la liste des résultats correspondant à un besoin d'information, car cette liste peut contenir un grand nombre d'éléments (son cardinal est grand). Pour cela, il est commun de qualifier la *pertinence* des résultats par une mesure chiffrée. Dans notre contexte, nous ne disposons pas de liens entre morceaux (comme c'est le cas pour les pages Web classées partiellement avec le PageRank [Pag+98a]), mais nous nous intéressons au contenu musical des documents, en particulier leur rythme.

3.4.1 Similarité par blocs de rythme

Comme on l'a vu précédemment avec les figures 3.2, 3.5, et 3.6, même si plusieurs documents correspondent à une requête donnée sur l'aspect mélodique, certains sont rythmiquement plus proches que d'autres. Le score que nous proposons dépend donc de la similarité rythmique entre un motif P (*pattern*) et les sous-arbres dans chaque descripteur de contenu parmi les résultats.

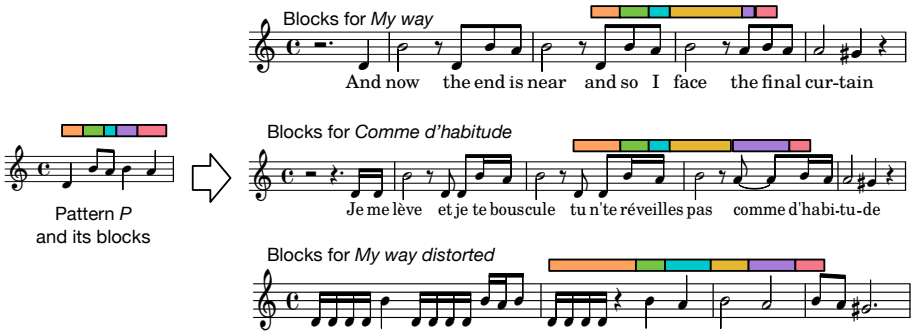


Figure 3.9 : *Les blocs, dans plusieurs fragments correspondant au motif de la Figure 3.8.*

On sait qu'il y a une grande similitude mélodique entre eux : comme $A_{FT}(P) = F$, F étant un fragment de $A_{FT}(D)$, il existe une suite non nulle d'intervalles identiques dans P et D. On représente chaque intervalle par une liste d'événements, que nous appelons un *bloc*. La Figure 3.9 illustre ce concept de *bloc*, pour les descripteurs des Figure 3.2, 3.5 et 3.6, avec le motif de la Figure 3.8.

Définition 16. *Soit $F =< I_1, \cdots, I_n >$ un fragment de $A_{FT}(D)$, d'un descripteur D. D'après la définition de A_{FT}, chaque intervalle $I_i, i \in [1, n]$ de F correspond à une sous-séquence $< p_1^i, e_2^i, \cdots, e_{k-1}^i, p_k^i >$ de $ESeq(D)$ telle que :*

- *p_1^i et p_k^i sont deux valeurs distinctes (et ne sont pas des silences), et $interval(p_1^i, p_k^i) = I_i$*
- *chaque $e_l^i, l \in [2, k-1]$ est soit un silence, soit une note telle que $e_l^i = p_1^i$*

$B_i =< p_1^i, e_2^i, \cdots, e_{k-1}^i >$ est appelé le bloc *de I_1 dans D,.*

Pour calculer un score de similarité (ou plutôt, son opposé, une distance), nous utilisons une distance d'édition entre arbres (*tree-edit distance*) [PT11], puisque notre représentation du rythme repose sur une structure d'arbre. Pour passer d'un arbre T_1 à un arbre T_2, on dispose d'opérations (des « éditions »), chacune étant associée à un coût. Les opérations standards sont : ajout d'un nœud, suppression d'un nœud, remplacement d'un nœud. La distance est définie comme la suite d'opérations de coût minimal (il existe évidemment de nombreuses suites d'opérations qui peuvent permettre de passer d'un arbre à un autre). Il y a des contraintes sur les opérations en fonction de la position dans l'arbre, pour assurer la préservation de la structure (non détaillées ici). On fixe le coût de chaque opération comme correspondant à la durée du nœud modifié : insérer, remplacer, supprimer une mesure coûte 1, insérer, supprimer un sous-arbre qui a pour racine une blanche (qui dure une demi-mesure) compte pour $\frac{1}{2}$, etc. : plus la modification est de courte durée, plus le coût est réduit.

Définition 17 (Similarité et distances rythmiques)**.** *Soient D_1 et D_2 deux descripteurs, leur similarité rythmique $Rsim(D_1, D_2)$ est la séquence d'opérations de coût minimal permettant de transformer l'arbre de rythme de D_1 en celui de D_2. La distance rythmique est $Rdist(D_1, D_2) = 1 - Rsim(D_1, D_2)$.*

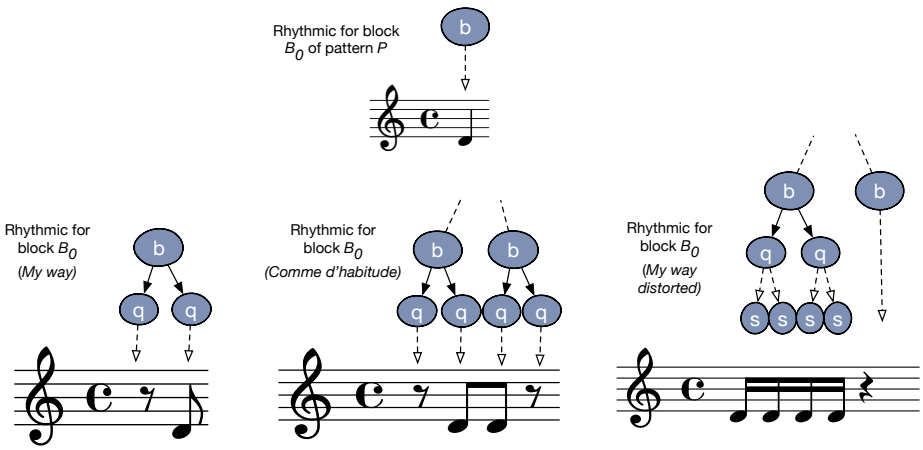


Figure 3.10 : *Arbres de rythme pour le premier bloc de chaque descripteur.*

Le calcul effectif de cette distance d'édition se fait avec des équations de programmation dynamique. Les algorithmes standards [Dem+09 ; ZS89] s'exécutent quadratiquement en temps par

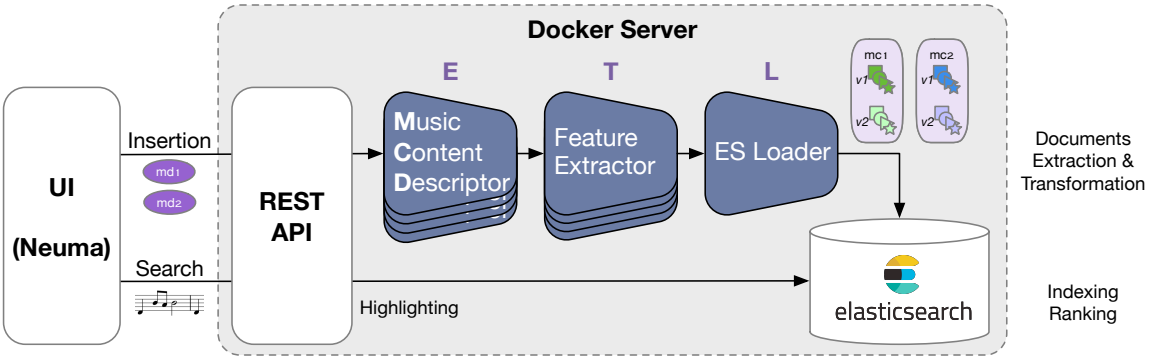


Figure 3.11 : *Notre architecture pour la recherche d'information*

rapport à la taille de l'entrée. La Figure 3.10 présente les arbres rythmiques pour le motif P et les premiers blocs des fragments qui lui correspondent. Les opérations d'édition requises sont respectivement une insertion de nœud (*My Way*), deux insertions (un temps et deux croches, *Comme d'habitude*), et 5 insertions pour *My way distorted*.

3.4.2 Classement final

La fonction de classement prend en entrée une paire de descripteurs et produit un score. Elle calcule ensuite l'alignement des blocs et somme les distances entre leurs arbres de rythme, pour aboutir à la distance $Rdist$ (Définition 17). Une fois que les descripteurs correspondant au motif ont été trouvés dans la collection, il faut localiser les séquences d'événements dans les documents originaux qui ont conduit à cette correspondance. Les moteurs de recherche standard disposent d'une fonction de « surbrillance » (*highlighting*), capable d'identifier les n-grams. Cependant, il n'est en général pas possible d'*inverser* un analyseur : il nous faut conserver une table de correspondance qui à chaque n-gram associe la séquence d'événements qui l'a produit.

3.5 Implémentation et interface graphique

3.5.1 Architecture

La Figure 3.11 illustre l'architecture que nous avons utilisée pour implémenter le framework présenté sur la figure 3.1. Un ETL (Extract/Transform/Load) se charge de produire les caractéristiques musicales des documents musicaux, Elasticsearch indexe ces caractéristiques, traite les requêtes et classe les résultats. Une API REST complète permet de charger comme d'interroger le moteur.

Un document indexé est de la forme suivante :

```
{
    "_id": "doc_id:descr_id",
    "chromatic": "7+;3-;2+; 3-;2+;1+; 2+;1+;1-; 1+;1-;5+;...",
    "RAT_chromatic": {[...]},
    "diatonic": "Fi+;T-;Se+ T-;Se+;Se+ Se+;Se+;Se- Se+;Se-;Fo+...",
    "RAT_diatonic": {
        "Fi+;T-;Se+": [1, ...],
        "T-;Se+;Se+": [2, ...],
        ... },
    "rhythmic": "(1)(1/2)(1) (1/2)(1)(2) (1)(2)(1/2) (2)(1/2)(1)...",
    "RAT_rhythmic": {[...]},
    "lyrics": "And now, ....",
    "RAT_lyrics": {[...]}
}
```

De nouveaux extracteurs de caractéristiques pourraient facilement être intégrés au système, cela ajouterait de nouveaux champs à chaque document indexé.

3.5.2 Interface

Ces travaux ont d'abord été implémentés dans la bibliothèque de partitions numériques Neuma (déjà mentionnée précédemment, voir [Rig+12 ; Neuma]). Depuis 2021, ils sont également la base du démonstrateur FACETS du projet européen Polifonia[5], pour lequel nous développons un outil de gestion de corpus reposant sur les facettes de la musique. La recherche par le contenu est un des aspects de cet outil, illustré sur la figure 3.12. Un clavier de piano permet de saisir le motif musical à rechercher, et l'on peut interroger l'un des index disponibles (une « collection » de documents) selon l'un des modes (mélodique ou rythmique).

3.6 Conclusion

Les travaux présentés dans ce chapitre constitue un prolongement des travaux du chapitre 2. Ils partent de la représentation abstraite de documents musicaux pour aboutir à un système novateur complet de recherche de partitions, capable de trouver des motifs mélodiques et des motifs rythmiques dans de grandes collections, et de classer les résultats en fonction de mesures de similarité adéquates. La diffusion de ces travaux se fait par la publication [Zhu+22], mais aussi par la mise en ligne du code source [6], d'un site web de démonstration [7] et d'une image Docker librement réutilisable.

En terme de perspectives, il serait logique et intéressant d'élargir les représentations musicales disponibles au contenu audio et sous forme d'images. Cela suppose de créer des analyseurs dédiés, et

[5] Accessible à l'adresse http://neuma-dev.huma-num.fr/.
[6] Voir https://github.com/polifonia-project/facets-search-engine/
[7] http://neuma-dev.huma-num.fr/

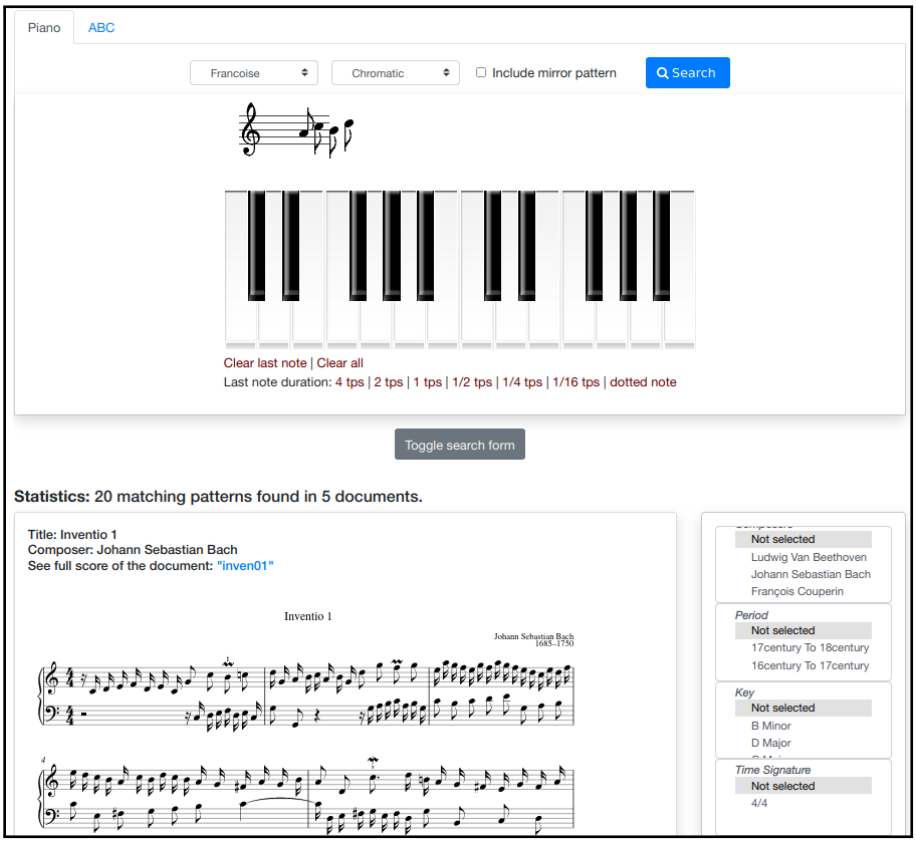


Figure 3.12 : *Interface de recherche par le contenu dans FACETS.*

cela fait déjà, de façon plus ou moins directe l'objet de recherches. Pour l'audio, le processus de transcription automatique fonctionne raisonnablement bien sur des données contrôlées et monodiques. Pour les polyphonies, c'est encore délicat. De façon similaire, pour les images, les performances d'OMR-isation, requièrent encore trop souvent des corrections manuelles *a posteriori*. La qualité des données est un enjeu-clé de la recherche d'information : si les utilisateurs cherchent quelque chose qui existe mais a été mal encodé, ils ne pourront le trouver.

D'autre part, le moteur de recherche que nous avons implémenté pourrait bénéficier d'autres améliorations. Par exemple, arriver à un *clustering* dynamique des collections ou des résultats d'une requête pour donner une *recherche à facettes*, qui faciliterait la navigation dans de grandes collections de documents musicaux. L'extraction de nouvelles caractéristiques musicales serait ici précieuse : harmonie, tonalité, etc. L'alignement des méta-données sur des standards permettrait d'accroître la taille des corpus présentés, et la visibilité de l'outil. Il serait également intéressant de poursuivre notre travail en proposant une recherche d'information qui ne soit pas uniquement « par le contenu » : l'enrichissement de ce contenu en récupérant des informations supplémentaires – via des graphes de connaissance, ou à l'aide de *Large Language Models* – est une perspective de recherche, qui suppose de mener une étude sur les ontologies musicales existantes et leurs possibilités de rapprochement de nos travaux.

Chapitre 4

Produire des partitions

[...]
La Vie devint ritournelle
Fugue Impromptu
Refrain

Se fit dissonance
Mélodie Brisure
Se fit battement
Cadence Mesure

Et se mira
Dans le destin
[...]

Andrée Chedid[1]

Sommaire

4.1 Interroger et manipuler des (ensembles de) partitions **44**
4.1.1 Schémas 45
4.1.2 Définition des requêtes 45
4.1.3 Évaluation des requêtes 46
4.2 Comparer des partitions **47**
4.2.1 Arbres de ligatures, arbres de n-olet 48
4.2.2 Comparaisons inter-mesures et intra-mesure 49
4.2.3 Visualisation de différences 50
4.3 Conclusion **51**

[1] Andrée Chedid. *Rythmes*. Paris : Gallimard, 2018.

Dans ce chapitre, nous allons voir comment, avec les partitions abstraites et l'algèbre définies précédemment, nous pouvons *manipuler* les partitions à l'envi, et combiner cela avec un système de base de données classique. Dans un second temps, nous verrons comment l'utilisation de distance d'édition sur certaines structures arborescentes permettent de mettre en évidence les différences entre partitions.

La manipulation de partitions permet d'envisager des études musicologiques approfondies, avec une granularité fine. Plus largement, c'est un maillon essentiel de la mise en œuvre d'une chaîne de traitement complète de partitions, en vue de disposer de bibliothèques numériques avec des fonctionnalités complexes. Les travaux sur la différence entre partitions constituent une autre étape vers cet objectif : pour disposer de bibliothèques *utiles*, il faut, autant que possible, assurer la qualité des données qui y sont entreposées, par exemple en ayant un outil pour identifier automatiquement des erreurs de transcription.

4.1 Interroger et manipuler des (ensembles de) partitions

Pour profiter au maximum de l'existant, tant en terme de systèmes que de corpus de partitions, nous avons choisi d'implémenter l'algèbre vue précédemment dans un système de bases de données orienté XML, qui dispose du langage XQuery [XQuery].

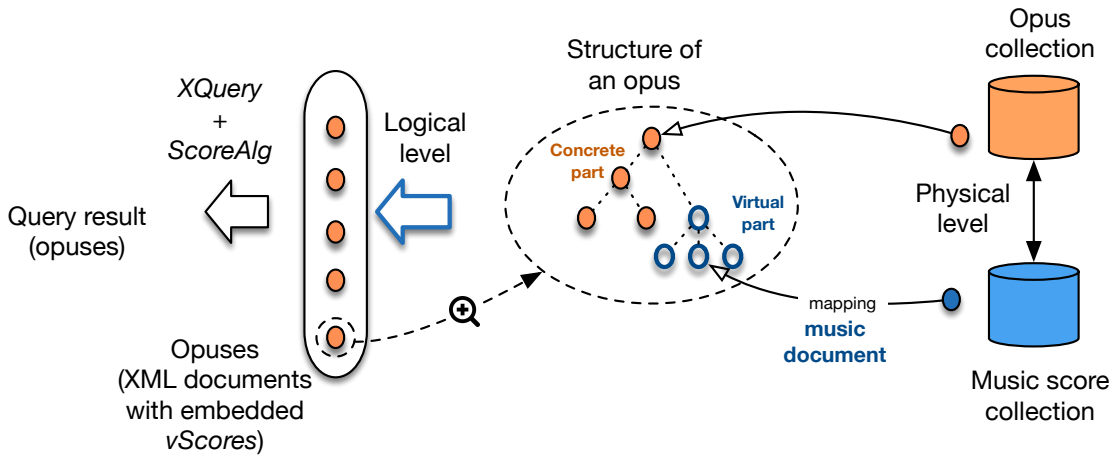


Figure 4.1 : *Architecture d'un système interrogeant des partitions.*

L'architecture est illustrée sur la figure 4.1. Chaque document XML contient une ou plusieurs partitions (*vScore*). Ces vScores sont manipulables directement avec des opérateurs XQuery, ou par nos opérateurs algébriques (transformés en fonctions XQuery). Une vScore reste le plus possible virtuelle, et n'est instanciée qu'à l'exécution. Au niveau physique, les opus sont représentés comme des documents XML standards, en MusicXML ou MEI (*cf.* Chapitre 1). Quand c'est nécessaire, le système récupère les documents, retire les informations inutiles et produit une instance de vScore à partir du contenu musical.

4.1.1 Schémas

L'implémentation repose sur des schémas, qui peuvent être assez flexibles, ou au contraire très contraints. Plus le schéma est strict, plus on s'assure que les éléments d'une collection ont une structure homogène, ce qui simplifie le travail des programmeurs d'applications aux niveaux supérieurs, puisque l'organisation des éléments est connue et attendue. Cependant, selon les corpus de données disponibles, voire d'un document à l'autre, le nombre de voix, leurs identifiants, etc. peuvent varier grandement.

Pour un morceau, on peut disposer d'un schéma comme celui de la figure 4.2a. Il contient des méta-données textuelles (`title, composer, published`), puis une partition, un élément de type `scoreType`. Ce dernier peut avoir un schéma comme celui de la figure 4.2b, contenant soit une ou plusieurs voix, soit une ou plusieurs scoreType (une vScore est un arbre de voix ou de vScores). La figure 4.3 présente un schéma pour un *choral* contraint à disposer d'un certain nombre de voix.

```
<xs:complexType name="opusType">
   <xs:sequence>
      <xs:element name="title" type="xs:string"/>
      <xs:element name="composer" type="xs:string"/>
      <xs:element name="published" type="xs:string"/>
      <xs:element type="scoreType"/>
   </xs:sequence>
   <xs:attribute type="xs:ID" name="id"/>
</xs:complexType>
```

(a) *Schéma pour un opus.*

```
<xs:complexType name="scoreType" abstract="true">
  <xs:sequence minOccurs="1" maxOccurs="unbounded">
      <xs:choice>
        <xs:element type="scoreType"/>
        <xs:element type="voiceType"/>
      </xs:choice>
  </xs:sequence>
</xs:complexType>
```

(b) *Schéma d'un type pour une partition.*

Figure 4.2 : *Exemples de schémas pour un opus et une partition (*`scoretype`*).*

```
<xs:complexType name="quartetType">
   <xs:complexContent>
      <xs:restriction base="scoreType">
         <xs:sequence>
            <xs:element name="soprano" type="soundVoiceType"/>
            <xs:element name="alto" type="soundVoiceType"/>
            <xs:element name="tenor" type="soundVoiceType"/>
            <xs:element name="voices" type="soundVoiceType"/>
         </xs:sequence>
      </xs:restriction>
      <xs:attribute type="xs:ID" name="id"/>
   </xs:complexContent>
</xs:complexType>
```

Figure 4.3 : *Exemple de schéma pour un choral.*

Ce modèle de données pourrait disposer d'une représentation pour les événements, mais nous limitons la granularité au niveau des voix pour éviter de manipuler des séquences d'événements, ce qui serait inutilement complexe.

4.1.2 Définition des requêtes

Avec le langage XQuery, on dispose de fonctions qui peuvent accéder finement à la hiérarchie du document musical. En représentant les expressions de notre algèbre avec de telles fonctions, on peut interroger et manipuler le contenu musical, sans devoir développer une extension importante du

système de base de données sous-jacent. Voici quelques exemples de requêtes, avec une collection de chorals[2] de Bach. Chaque opus a 4 voix (respectivement nommées *soprano*, *alto*, *tenor* et *bass*), comme dans le schéma XML *supra*[3]. La requête 4.1 permet d'extraire les 5 premières mesures de la voix de soprano pour tous les chorals.

```
for $o in collection("Chorals")/opus
let $incipit := scoreql:select ($o/score/soprano, "measure() in [1,5]")
return <result>
          <title>{$o/title}</title>
          <incipit>{scoreql:eval($incipit)}</incipit>
       </result>
```

Requête 4.1 : *Filtrage par les méta-données, sélection et projection.*

On voit comment s'intègrent XQuery et notre algèbre. Outre les fonctions XQuery pour implanter les opérateurs, on recourt à XPath, qui exprime l'opérateur de projection π. Ainsi, `$o/score/soprano` correspond à $\pi_{\text{soprano}}(\$o.score)$. Le `select ($o/score/soprano, «measure() in [1,5] »)`, qui correspond à l'opérateur de sélection, se fait à l'aide d'un appel de fonction algébrique. Les expressions de l'algèbre sont évaluées *paresseusement* (*lazy evaluation*). Cela permet de décomposer une expression complexe en plusieurs étapes, affectées à des variables XQuery, et de faire référence à la même variable plusieurs fois sans avoir besoin d'effectuer plusieurs fois l'évaluation. C'est la fonction *eval()* déclenche l'évaluation.

À titre d'exemple, la requête 4.2 réalise une opération courante de transposition, ici en abaissant de 2 demi-tons les voix de basse et soprano, si la voix de soprano va au-delà d'un F5.

```
for $o in collection("Chorals")/opus
where scoreql:highest($o/score/soprano) > scoreql:frequency("F5")
let $transpS := scoreql:map ($o/score/soprano, "transpose (-2)")
let $transpB := scoreql:map ($o/score/bass, "transpose (-2)")
return   scoreql:eval (scoreql:sync ($transpS, $transpB))
```

Requête 4.2 : *Synchronisation,* `map` *et fonction utilisateur (UDF).*

La fonction *highest()* prend une vScore et retourne une valeur, la plus haute note dans la voix de soprano. L'opérateur `map` effectue la transposition et `sync` se charge finalement de créer la partition finale avec les voix transposées.

4.1.3 Évaluation des requêtes

La figure 4.4 montre le système d'évaluation des requêtes. À la première étape, la requête est envoyée au processeur XQuery, et récupère l'ensemble des opus correspondant à la clause `where`. Chaque opus (les disques verts) sont liés à des partitions sérialisées. Durant l'évaluation de la clause `return`, la fonction *eval(exp)* déclenche l'évaluation de l'expression. Il y a ensuite 3 étapes, pour obtenir le résultat (une ou plusieurs vScore(s)) :

[2]Un choral (des chorals au pluriel) est un genre musical liturgique, voir : https://fr.wikipedia.org/wiki/Choral.

[3]L'ensemble des chorals est disponible à l'adresse : http://neuma.huma-num.fr/home/corpus/composers:bach:chorals/

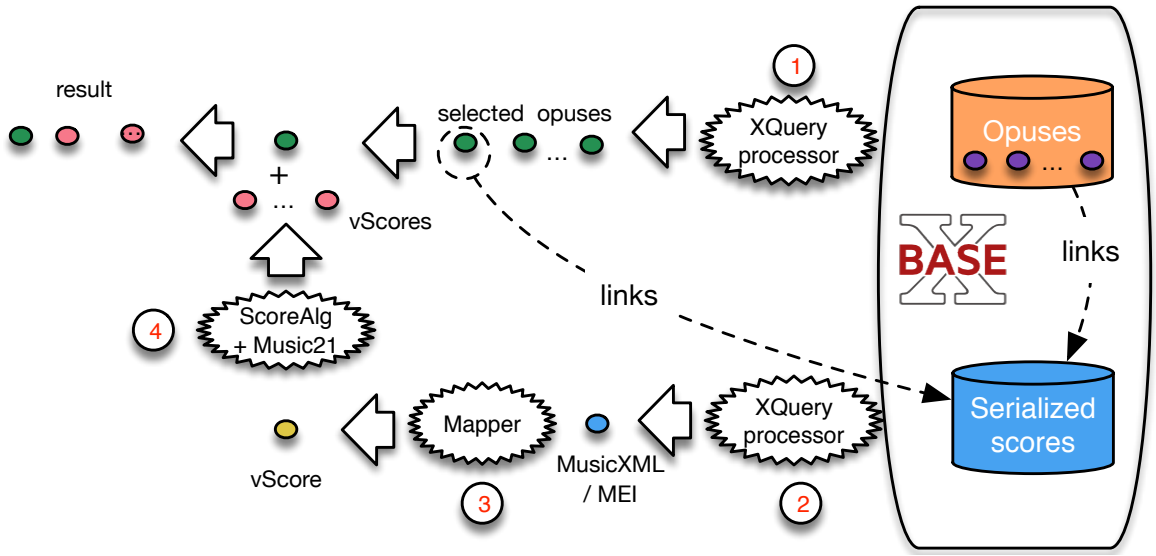


Figure 4.4 : *Évaluation des requêtes.*

1. on récupère le document XML S de l'opus courant (via la référence)
2. on applique le mapping M pour extraire le contenu musical de la vScore
3. on évalue exp, à l'aide des opérateurs algébriques et de Music21 pour les *User Defined Functions*.

Cette algèbre a été intégrée dans la bibliothèque de partitions numériques NEUMA et une version indépendante a été rendue publique sous la forme d'une image Docker[4]. L'implémentation s'appuie sur des outils standards, à savoir une base de données XML native, BASEX[5] et une bibliothèque Python de notation musicale, MUSIC21[6][CA10]. En terme de performances, hormis la fonction map qui peut être très complexe et s'applique au niveau des événements (le plus bas), le coût des autres opérateurs est négligeable devant ceux de chargement d'un système de base de données standard [FRT18].

4.2 Comparer des partitions

Dans cette section, nous étudions comment comparer des partitions, en vue d'assister le travail de production de partitions (et leur analyse). On développe un outil, composé de plusieurs parties, pour relever les différences à deux niveaux : concernant le *contenu musical* et concernant la *notation* (dont on a surtout cherché, dans les autres travaux, à s'abstraire). En effet, en XML/MEI, il existe plusieurs façons de représenter les mêmes contenus graphiques, par exemple en utilisant des structures imbriquées pour les ligatures, ou en échangeant simplement l'ordre des notes dans un accord. La figure 4.5 présente trois portées, illustrant ces niveaux auxquels on peut comparer des partitions.

[4]Voir https://hub.docker.com/r/traversn/scoreql/.
[5]http ://basex.org
[6]http ://web.mit.edu/music21

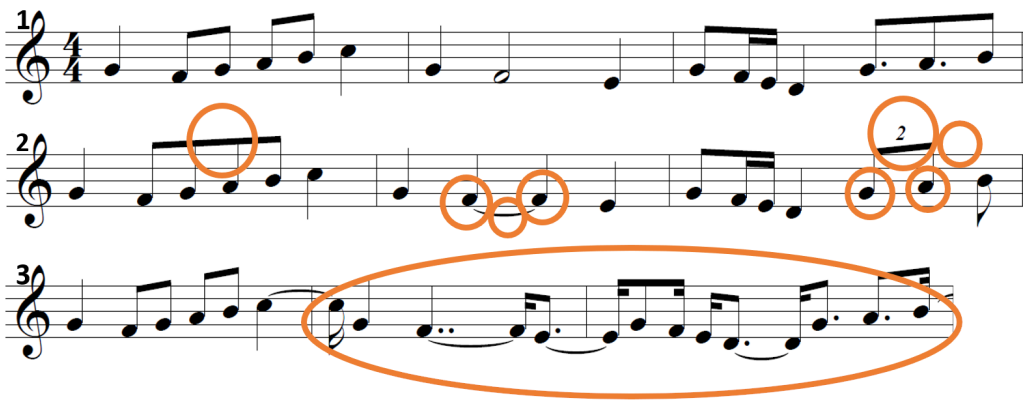


Figure 4.5 : *Trois portées pour illustrer la comparaison de partitions au niveau musical ou au niveau du rendu graphique. Les portées 1 et 2 ont le même contenu musical mais leurs notations diffèrent. Les portées 1 et 3 diffèrent d'une note seulement (un C plus long à la fin de la première mesure) mais elles sont très différentes d'un point de vue musical.*

Une partition XML est une structure récursive : au niveau supérieur, nous avons une liste de ses parties (généralement une par instrument), chaque partie est constituée d'une liste de mesures, chaque mesure est une liste de voix et chaque voix est une liste de *symboles* (notes, silences ou accords) qui se produisent de manière séquentielle sans se chevaucher. Notre approche est similaire à celle utilisée pour l'outil `diff` textuel [HM76 ; Hec78], qui cherche d'abord les lignes qui ont des différences, avant de rechercher les différences au niveau des mots. Nous procédons donc en deux phases : une étape « inter-mesures », où l'on trouve les séquences de mesures identiques les plus longues, suivie d'une étape de recherche des différences à l'intérieur des mesures. La sortie de notre outil est une distance (le nombre minimal d'opérations pour passer d'une partition à l'autre), et une liste de différences. Techniquement, on commence par représenter chaque mesure d'une partition par deux arbres, un arbre de ligatures (en anglais, *beaming tree*) et un arbre de n-olet (en anglais, *tuplet tree*). La comparaison au niveau des mesures se fait à l'aide d'un algorithme de Longest-Common-Subsequence (LCS) sur ces (paires d') arbres. Dans un second temps, on rentre dans les mesures non alignées par cet algorithme en utilisant des distances d'édition sur arbres, pour identifier les différences.

4.2.1 Arbres de ligatures, arbres de n-olet

Le rendu graphique d'une partition. Celui-ci se compose de deux parties, la mélodie et le rythme. Pour la première, la représentation repose sur les clefs, les positions sur la portée et les altérations. Pour le rythme, on utilise les têtes de notes, les ligatures, les triolets. La structure des ligatures et des triolets est hiérarchique : à partir de la modélisation arborescente présentée dans le chapitre 2, on peut dériver une modélisation du rythme sous forme d'arbres. Nous définissons des arbres de ligature et des arbres de triolet (plus précisément, de n-olet, mais n vaut souvent 3).

Ces arbres partagent les mêmes symboles et ont, pour une mesure donnée, les mêmes ensembles de feuilles. Chaque feuille contient les informations suivantes :

- une note/un accord, une altération, un marqueur de liaison (indiquant si la note est liée à la feuille la plus proche sur sa gauche)

- une tête de note
- une valeur spécifiant si la note est pointée
- un booléen précisant s'il s'agit d'une appogiature (*grace note*).

L'arbre de ligature encode le nombre de ligatures au-dessus d'une note et le nombre de ligatures entre deux notes. L'arbre de triolet représente le nombre de crochets numérotés qui définissent les n-olets. La figure 4.6 présente ces arbres pour deux exemples.

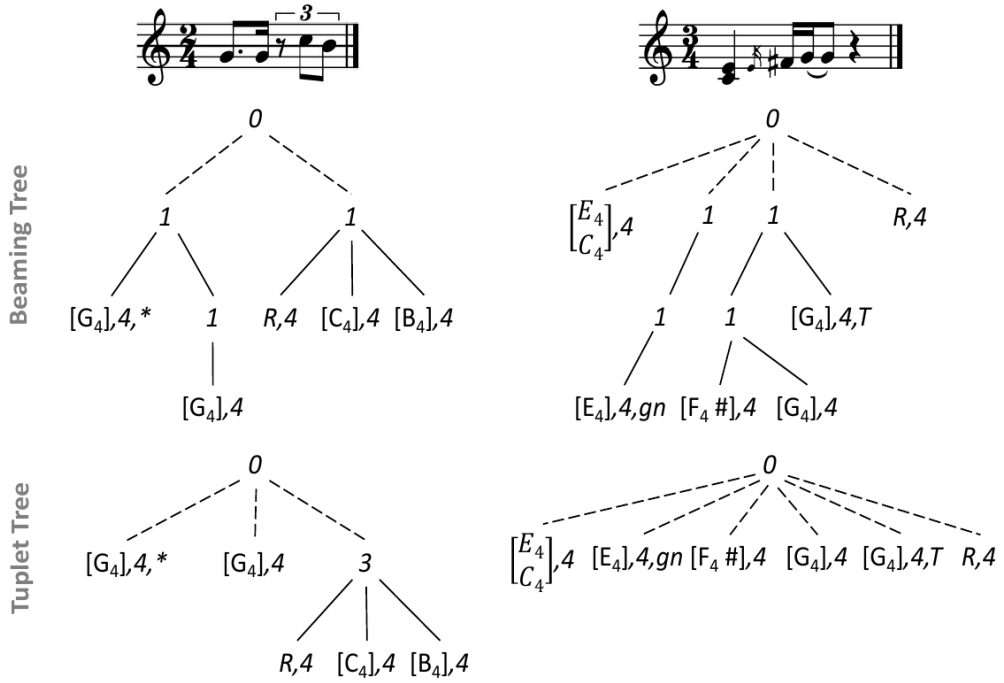


Figure 4.6 : *Arbre de ligatures et arbre de triolet.*

4.2.2 Comparaisons inter-mesures et intra-mesure

Pour comparer deux parties, venant de deux partitions différentes, on représente chaque partie par une séquence de paires $p =< bt, tt >$, une paire par mesure, où bt est un arbre de ligature et tt est un arbre de triolet. On aligne les mesures identiques des deux parties en trouvant la plus longue séquence commune de paires identiques (*longest common subsequence*, LCS). On peut trouver un alignement maximal de ces paires identiques, et définir des *blocs de différences* (*cf.* page 37, en particulier la Figure 3.9).

Les comparaisons à l'intérieur de deux mesures visent à identifier les (positions des) différences, et à calculer une distance. On compare séparément les arbres de ligatures et les arbres de triolet des mesures. Les arbres de triolets ont des structures assez simples, on peut les comparer avec les mesures de distance d'arbres (*tree-edit distance*) de Zhang-Sasha [ZS89], qui calcule le plus petit nombre d'éditions de nœuds qui sont nécessaires pour passer d'un arbre à l'autre. La comparaison d'arbres de ligatures est plus délicate, nécessitant de distinguer finement des éditions d'insertion ou de suppression de sous-arbres dans la structure. La comparaison de blocs de différences fait

appel aux comparaisons d'arbres de ligatures et d'arbres de triolet, en se munissant en outre d'une fonction capable d'éviter de compter deux fois certaines erreurs [FFJ19a]. Enfin, la distance entre deux partitions est la somme des distances entre tous les blocs de différences.

4.2.3 Visualisation de différences

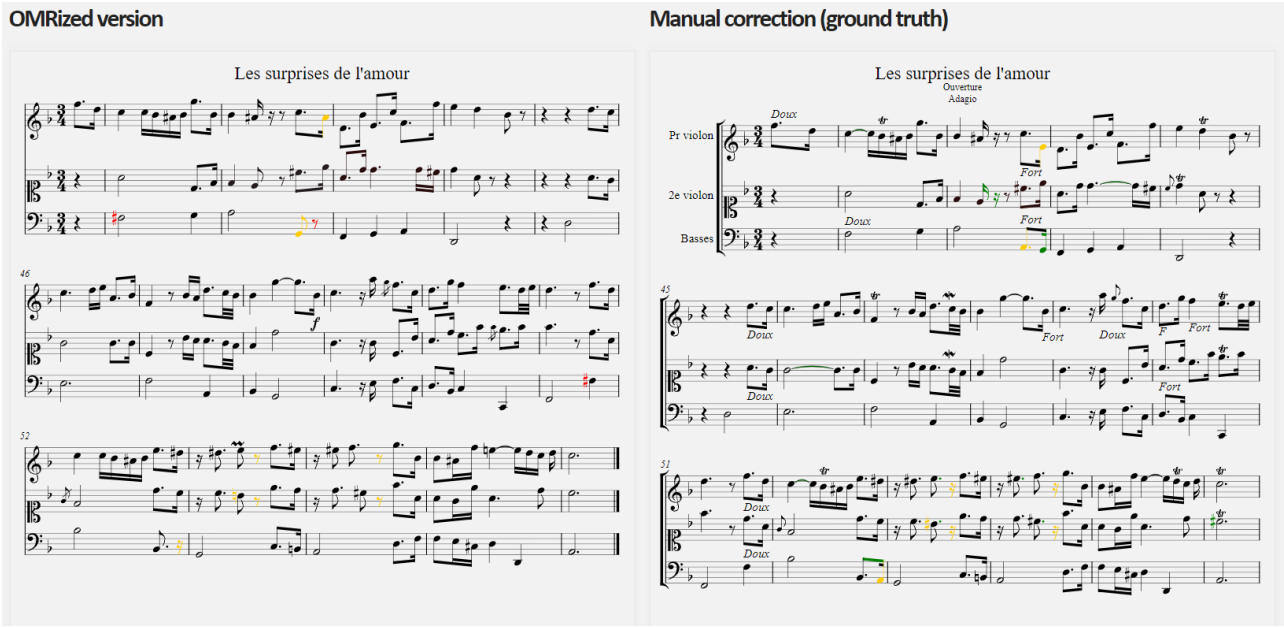


Figure 4.7 : *Deux partitions côte à côte dans l'outil graphique.*

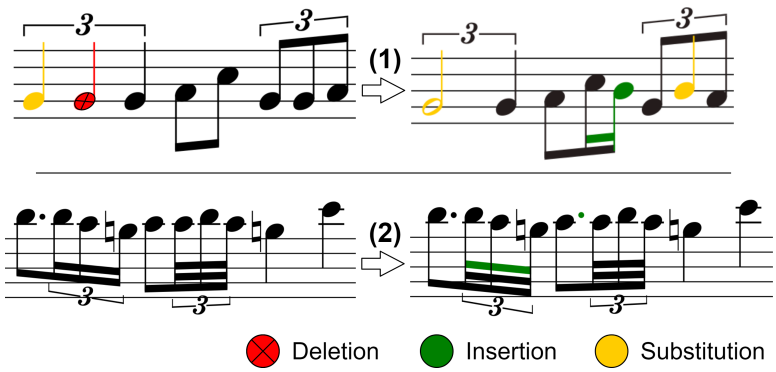


Figure 4.8 : *Visualisation de différences entre mesures. En haut à gauche, une mesure de référence, modifiée manuellement pour ajouter les différences visualisées à droite. En bas, un exemple issu d'une procédure OMR (à gauche), comparée à la référence, à droite.*

Nous avons développé un outil de visualisation qui met en évidence graphiquement les différences entre partitions, grâce à la bibliothèque Verovio [PZR14], comme illustré sur les Figures 4.7 et 4.8.

4.3 Conclusion

Dans ce chapitre, nous avons présenté deux contributions, la manipulation de partitions à l'aide d'une algèbre munie de fonctions classiques et arbitraires, et la comparaison de partitions à l'aide de distance d'édition sur des représentations à base d'arbres. Ces travaux sont importants pour la production de partitions, c'est-à-dire pour la transcription de la musique en musique notée. On peut ainsi intégrer l'outil de comparaison dans divers processus, comme ceux d'OMR ou l'édition collaborative de partitions (en vue d'améliorer la qualité des données produites), ou pour l'analyse, pour confronter la performance d'un interprète avec une partition de référence. De même, les « manipulations algébriques de partitions » peuvent être utilisés en analyse musicologique (pour sélectionner des éléments de partitions en vue de les regrouper) ou pour de la création musicale (transposer, mélanger des voix et produire de nouvelles partitions).

En terme de perspectives, elles sont essentiellement applicatives : il serait intéressant de rendre plus robustes nos implémentations et de les intégrer à des outils complets afin que des musicologues puissent s'en saisir et en tirer pleinement partie.

Chapitre 5

Conclusion et perspectives

Laquelle des deux puissances peut élever l'homme aux plus sublimes hauteurs, l'amour ou la musique ? C'est un grand problème. Il me semble qu'on devrait dire ceci : L'amour ne peut pas donner une idée de la musique, la musique peut en donner une de l'amour.

Hector Berlioz[1]

5.1 Contributions

Le domaine de la recherche d'information musicale symbolique avait été marqué par la publication en 2013 d'une feuille de route[2] [Ser+13], fruit d'une collaboration large entre chercheurs majeurs du domaine (projet MIReS). Celle-ci ne se limitait pas à la notation symbolique et identifiait 4 directions principales : « technologie », « utilisateurs », « phénomène socio-culturel », « exploitation », soulignant que la première était la plus classique dans le domaine. En 2015, Burgoyne *et al.* affirmaient que « les humanités numériques allaient contribuer substantiellement à mettre les données patrimoniales sur Internet, non seulement pour les universitaires mais aussi pour l'apprentissage tout au long de la vie et pour le grand public. Les outils et techniques développés dans ce domaine vont faciliter l'étude de ces travaux. » [BFD15]. Plusieurs années après la publication de ces articles, nous pouvons constater que mes travaux ont œuvré dans les directions identifiées.

Comme nous l'avions vu en introduction, mes travaux ont repris l'articulation de cette « feuille de route scientifico-technologique » MIReS, autour d'une chaîne de traitement assez classique en recherche d'information : on conçoit des représentations numériques, correspondant aux données

[1] Hector Berlioz. *Mémoires*. Paris : Calmann-Lévy, 1878.

[2] Voir le site du projet MIReS à l'adresse https://mires.eecs.qmul.ac.uk/.

et adaptées aux traitements ultérieurs que l'on souhaite rendre possible. Ces représentations sont ensuite analysées, et le résultat de ces analyses peut être, ou non, matérialisé (avec éventuellement une représentation dédiée au stockage). Enfin, ces résultats peuvent être interrogés et *manipulés*. La figure 5.1 illustre ces étapes.

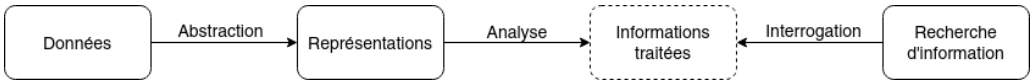


Figure 5.1 : *Les données sont représentées numériquement, avant d'être analysées et interrogées*[3].

L'originalité de mes travaux réside, nous l'avons dit, dans le travail exclusif autour de la notation musicale. Dans le chapitre 2, nous avons tout d'abord détaillé la représentation du contenu musical dans les partitions avec une représentation reposant sur des arbres. Cela a permis d'aborder différentes tâches de l'informatique musicale, comme la gestion et les manipulations complexes de partitions, *via* un langage dérivé de Xquery. Puis, dans le chapitre 3, nous avons abordé une autre forme d'interrogation des partitions, à partir d'un moteur de recherche. Pour cela, nous avons détaillé la mise en œuvre de représentations à base de *n-grams*, un format compatible avec des moteurs de recherche pour le texte. Nous pouvons extraire divers aspects de la musique et les indexer, de façon à proposer un moteur de recherche centré sur le contenu. Celui-ci propose déjà une recherche flexible avec plusieurs modes (mélodique, rythmique), complétée par des fonctions de classement adaptées à la musique. Enfin, il a été conçu pour passer à l'échelle, avec une gestion de la distribution des données.

Enfin, dans le chapitre 4, nous avons présenté deux lignes de travaux ayant pour but de transcrire des partitions musicales et d'en assurer la qualité. Nos travaux sur l'algèbre de partitions permettent de manipuler des partitions existantes et de les transformer complètement, notamment grâce à la possibilité de définir des fonctions spécifiques. Outre les usages créatifs que l'on peut faire, une application pratique est de faciliter le travail du musicologue qui peut réorganiser finement des éléments d'un corpus. Dans un deuxième temps, nous avons vu des travaux visant à améliorer la qualité des données représentées dans des partitions musicales numérisées, avec la comparaison de partitions. L'outil et les algorithmes développés peuvent s'intégrer à des processus de production de partitions, collaboratifs ou individuels, professionnels ou amateurs.

5.2 Visibilité et diffusion

La publication de ces travaux s'est faite essentiellement dans des journaux et conférences du domaine *Music Information Retrieval*, à une ou deux exceptions près. Leur visibilité est sans doute un peu moindre que dans des conférences d'informatique généralistes. Nos travaux sur la comparaison de partitions ont été repris, par la communauté de la transcription, en OMR [RCR20 ; San20] ou en notation symbolique [DJR22 ; Fei21]. Nos travaux sur l'interrogation, plus récents, s'insèrent dans la communauté de la recherche d'information musicale [Rom+23 ; BSH23]. Notre équipe a

[3]Schéma personnel, qui peut être vu comme une simplification d'un schéma de [Ser+13].

donc pris toute sa place dans l'écosystème français et mondial de la recherche en informatique musicale, (co-)proposant et menant à bien plusieurs projets d'envergure (projets ANR MuNIR et H2020 Polifonia).

Mes travaux constituent un ensemble cohérents de résultats autour de la musique symbolique, avec des problématiques spécifiques, allant au-delà du seul « cas d'application » de techniques standards en informatique. Peu à peu, on a progressé vers *un moteur de recherche de partitions numériques* multi-fonctions, une étape importante à laquelle nous sommes arrivés, d'abord théoriquement, avec le travail de [Zhu+22], et qui culmine avec le développement récent d'une plateforme de démonstration et la mise à disposition d'une image Docker open source[4]

5.3 Perspectives

Aujourd'hui, bien que des progrès sensibles aient été effectués sur chaque problème, les tâches principales sur lesquelles les chercheurs en MIR souhaitent avancer sont sensiblement les mêmes qu'il y a dix ans :

- améliorer la qualité des données symboliques, notamment par de meilleurs procédés de transcription (depuis le MIDI ou des images)
- permettre la constitution de grandes bases de données (pour le public comme pour la reproductibilité de résultats de recherche)
- développer des représentations permettant des analyses complexes mais également interopérables et passant à l'échelle
- faire progresser la classification et l'exploration de données
- améliorer les méthodes du domaines, pour une bonne reproductibilité des résultats.

Comme le disent Burgoyne *et al.*, « peu de tâches MIR classiques sont pleinement résolues, et l'on s'attend à ce qu'une énergie considérable soit déployée pour améliorer l'état de l'art sur ces tâches fondamentales pendant encore un certain temps. » [BFD15].

La question des données est fondamentale : il n'y a pour le moment pas encore assez de données bien étiquetées et de bonne qualité à disposition des chercheurs et du grand public. Des efforts importants ont été réalisés [Fos+20 ; Raf16], mais constituer de grands corpus avec des formats inter-opérables et facilitant l'analyse reste encore un enjeu, pour décloisonner les pratiques et les entrepôts de données.

Quand les membres du projet MIReS ont publié leur feuille de route pour le domaine en 2013, nous étions quelques mois après la révélation des performances d'AlexNet [KSH17], l'architecture de réseau convolutionnel (CNN) qui allait remettre l'apprentissage profond au centre des préoccupations (d'abord, en vision par ordinateur). Au cours de ces dix dernières années, la porosité entre les communautés « vision », « traitement automatique du langage » et MIR s'est progressivement accrue, à mesure que l'on sortait d'un hiver de l'IA [CCM18], c'est-à-dire que les performances des

[4]Voir le prototype en fonctionnement à l'adresse http://neuma-dev.huma-num.fr/ et le code source à l'adresse https://github.com/polifonia-project/facets-search-engine.

modèles d'apprentissage s'amélioraient. Et les différents problèmes ci-dessus ont été abordés avec des approches d'apprentissage profond (voir [Wu+23 ; KW23 ; Zha+23] pour des exemples récents à ISMIR 2023). J'ai moi-même travaillé sur la possibilité d'utiliser des réseaux de neurones récurrents pour estimer les hauteurs et les clés d'un fichier MIDI, un problème de MIR assez ancien [FAF21]. Nous avions obtenu des performances comparables à l'état de l'art, avec une approche complètement différente. Les récents travaux révolutionnaires des architectures *attentionnelles* [Vas+17] proposent une approche encore nouvelle et l'on assiste à une convergence des pratiques en informatique musicale, en traitement de données symboliques [Zen+21] et de données *audio* [Haq+19 ; SW21]. La question de leurs performances (et donc de leur entraînement, gourmand en données), est encore l'objet d'études diverses. Leurs « hallucinations » demandent encore à être contrôlées, et une piste intéressante est de les coupler à des graphes de connaissance[5], pour ce qu'on appelle de la RAG (*retrieval augmented generation*). C'est une voie que je souhaite explorer dans les mois à venir. Ces architectures, mises à disposition du grand public avec des *chatbots* comme chatGPT, amènent à une question fondamentale en recherche d'information : la question des utilisateurs et de leur manière d'aborder les moteurs de recherche. Les attentes des utilisateurs ont été transformées[6] , et il n'est plus sûr qu'ils soient (ou restent longtemps) à l'aise avec un paradigme de recherche demandant de saisir des motifs mélodiques. Des nouveaux modèles d'interaction sont sans doute à (ré-)inventer, sur ces bases.

Au-delà d'un moteur de recherche, les travaux que nous avons réalisés sont des briques essentielles qui pourraient conduire, dans les années à venir, à la création d'une bibliothèque numérique collaborative de partitions de taille mondiale, similaire à ce qu'est Wikipedia pour la connaissance sous forme de texte. L'initiative Musipedia pourrait constituer la base de ce projet [Typ97], mais elle devrait cependant dépasser le cadre personnel de son développeur initial pour devenir collaborative à large échelle, et quelques choix techniques pourraient être revus, pour intéresser davantage de personnes. En particulier, la question de la multi-modalité est cruciale : la musique doit s'envisager davantage comme un tout, et les problématiques abordées doivent prendre en considération les formats symboliques, « images » et « sons ». La notation symbolique devrait pouvoir être lue seule ou avec un son synchronisé, être transcrite depuis une image, voire d'un enregistrement de concert.

[5] *knowledge-graphs*

[6] Les Transformers *transforment* nos domaines de recherche. *Pun intended.*

Deuxième partie

Systèmes de recommandation et graphes

Chapitre 1

Introduction

« Knowledge is knowing that a tomato is a fruit.
Wisdom is knowing not to put it in a fruit salad. »

Brian O'Driscoll[1]

Sommaire

1.1 Les systèmes de recommandation et les réseaux complexes **59**
1.1.1 La recommandation 59
1.1.2 La science des réseaux 61
1.2 Positionnement et contributions **62**
1.3 État de l'art **65**
1.3.1 Recommandation 65
1.3.2 Modèles de graphes 67
1.3.3 Recommander avec des graphes 68

1.1 Les systèmes de recommandation et les réseaux complexes

1.1.1 La recommandation

La *recommandation* désigne, en informatique, les techniques algorithmiques qui sont déployées pour proposer aux *utilisateurs et utilisatrices* d'un système un sous-ensemble des *éléments* disponibles, c'est-à-dire en recourant à un filtrage. En particulier, on s'intéresse aux systèmes dans lesquels cette sélection est entièrement automatisée. Le développement d'un système de recommandation repose en général sur le besoin, réel ou supposé, de lutter contre une surcharge d'information dans un

[0]Extrait d'une interview du capitaine de l'équipe d'Irlande de rugby, cité dans [Agg16] (source précise inconnue).

système informatisé : on souhaite à faciliter la navigation dans une grande collection de documents, pour que le système sous-jacent reste *intéressant* pour ses utilisateurs. Autrement dit, on organise, avec des machines, la surcharge de données numériques produites par les machines.

La recommandation consiste à prédire quelles sont les interactions qui sont les plus susceptibles d'avoir lieu dans le système, afin que l'utilisatrice n'ait pas à chercher elle-même. À l'échelle d'un individu, on cherche quels seront les éléments qui « devraient lui plaire ». On vise, plus ou moins explicitement, une transposition de la recommandation « non informatisée », le bouche-à-oreille qui existe depuis longtemps, quand une personne conseille *quelque chose* à quelqu'un, parce qu'elle connaît ses goûts, son besoin. Pour fonctionner, ces systèmes reposent sur la collecte d'informations, à propos des utilisateurs, des éléments du systèmes, ou des interactions (« utilisatrice A a vu et aimé ce film ce jour-là », « utilisateur B a écouté cette chanson », etc.). Selon les besoins et les données disponibles, le problème de la recommandation se modélise classiquement par une tâche dite de régression, où l'on souhaite prédire les « notes » sur tous les éléments du système (pour un utilisateur donné), et une tâche de classement, où les notes individuelles importent peu mais où l'on souhaite ordonner les éléments du système pour correspondre aux goûts de l'utilisateur (on parle de recommandation top-k car on choisit les k meilleurs éléments).

Les premiers modèles de systèmes informatiques de recommandation sont nés avec les travaux académiques du laboratoire GroupLens (University of Minnesota) autour des messages du réseau Usenet. Paul Resnick et John Riedl ont proposé, et amélioré avec leur collègues, un système collectant les notes d'articles et les utilisant pour prédire si les lecteurs allaient aimer un article donné [Res+94]. Ensuite, dès 1998, le libraire en ligne (et aujourd'hui géant mondial de l'e-commerce) Amazon a utilisé le filtrage collaboratif pour « réarranger ce que voient ses visiteurs, en fonction du contexte actuel de visite et du comportement passé » [LSY03]. En 2007, Netflix a proposé une compétition dotée d'un million de dollars pour améliorer son algorithme de filtrage collaboratif *Cinematch*. De nombreuses équipes proposèrent leur solution, avant que la victoire revienne au trio KorBell de chercheurs des AT&T Labs, Yehuda Koren, Robert Bell, et Chris Volinsky [TJB09 ; BKV07]. Durant les années 2010, les systèmes de recommandation ont été transformés par l'introduction des méthodes d'apprentissage profond (*deep learning*), dans le monde académique comme dans le secteur privé [Zha+19 ; CAS16]. Ces toutes dernières années, les *Graph Neural Networks* sont une architecture très étudiée.

L'étude des systèmes de recommandation est donc, comme la recherche d'information musicale abordée dans la première partie de ce manuscrit, une discipline issue de travaux en recherche d'information. Pensés d'abord pour compléter les moteurs de recherche, ces systèmes en ont d'abord utilisé des techniques (par exemple, des similarités reposant sur un score *tf.idf*). Progressivement, les échanges avec les autres champs de recherche, mais aussi avec la société ont mis la recommandation au centre des préoccupations de plusieurs disciplines : sociologie, psychologie, interaction homme-machine, etc. [BS12 ; Lex+21] Ainsi, depuis une quinzaine d'années, l'essor académique de la recommandation ne cesse de s'accroître : la conférence ACM RecSys est désormais extrêmement sélective, les prestigieuses conférences de fouille de données WWW-TheWebConf (centrée initialement sur le Web) et KDD (*Knowledge discovery in databases*) ont des journées entières dédiées aux publications sur ce sujet.

1.1.2 La science des réseaux

La science des réseaux, ou l'étude des systèmes complexes, est une discipline qui s'intéresse aux systèmes reposant sur les interactions entre les composants du système. « Un réseau est défini par deux éléments : l'ensemble des entités du réseau (que l'on nomme les *nœuds*) et l'ensemble des relations entre ces entités (que l'on nomme les *liens*) » [Tar19].

La théorie mathématique des graphes est née au XVIIIe siècle, avec le problème « des 7 ponts de Königsberg », résolu par Euler en 1735 [Ale06]. Sa solution modélisait la question de la promenade dans la ville comme un graphe (sans le nommer encore), avec des nœuds (les parties de la ville) et des liens (les ponts entre celles-ci). Dans les décennies et siècles suivants, la discipline a étendu et généralisé son étude (grâce à Cauchy, L'Huilier, Vandermonde et bien d'autres [Wik]). Le terme de « graphe » lui-même n'est apparu qu'en 1878, par proximité avec la chimie. Au XXe siècle, les travaux et ouvrages de Kőnig, Berge et Harary [Kön90 ; Har69 ; Ber58] d'une part, et d'Erdős et Rényi d'autre part [ER60], en ont fait un champ complet, aux ramifications dans de nombreuses autres sciences. La théorie des graphes offre donc un cadre formel pour étudier les réseaux, avec un vocabulaire et des concepts précis (« degré d'un nœud », « graphe orienté », « poids d'un lien »). Cependant, son approche se focalise sur l'étude des propriétés de l'objet « graphe », ou plutôt d'objets « graphes » particuliers, aux tailles réduites et bien définies (voir par exemple, « la conjecture des quatre couleurs »).

De façon indépendante, dès les années 1960, des chercheurs en psychologie sociale comme Milgram puis Granovetter ont étudié les réseaux sociaux par des études de terrain [Gra73 ; Mil67], mettant en évidence des phénomènes – aujourd'hui devenus des expression du langage courant – comme « les réseaux petit-monde », « les six degrés de séparation » ou « la force des liens faibles ». Cependant, c'est seulement à la fin du XXe siècle que les travaux de Kleinberg, Watts et Strogatz ont fait converger ces études avec le formalisme de la théorie des graphes, faisant naître une nouvelle sous-discipline [WS98 ; Kle00]. Les réseaux *complexes* sont progressivement devenus un sujet d'étude à part entière. Le réseau Internet, décentralisé, a amené un besoin pour davantage de « science des réseaux », en vue de le comprendre, d'appréhender les phénomènes qui y prenaient place, de l'améliorer (nouveaux protocoles, résolutions de problèmes de routage, etc.). Mais l'Internet a aussi apporté de nombreuses données, issues de réseaux collectés à divers niveaux du modèle OSI, des routeurs aux « réseaux sociaux en ligne » (comme Facebook et Twitter). On a rapidement constaté que nombre de réseaux ainsi constitués avaient des propriétés bien différentes des graphes étudiés par les mathématiciens. L'analyse de méthodique de ces réseaux sociaux[1] est devenu un outil important pour les sciences sociales modernes et a également des applications en sciences de la communication, en économie, géographie [Col+06 ; Lam+08 ; BMZ11]. De nouveaux concepts et de nouvelles techniques sont ainsi apparus, comme la détection de communautés [Blo+08] ou les réseaux « sans échelle » de Barabási et Réka [BA99] : utiles dans des contextes très divers, des réseaux de transport aériens aux interactions entre protéines en passant par des réseaux financiers, ils ont renforcé la légitimité de cette discipline en lui permettant de tisser des liens avec d'autres disciplines. Plus récemment, l'étude de l'évolution des graphes au cours du temps, qui a mené à la mise au point de modèles de *graphes dynamiques* [ML16 ; HS19], apparaît progressivement comme un autre apport majeur de la science des réseaux.

[1] En anglais : *social network analysis* (SNA).

1.2 Positionnement et contributions

Depuis le milieu du XXe siècle, nous assistons à une numérisation progressivement généralisée des activités humaines, en particulier médiatiques et culturelles (écrits, musiques, images fixes et animées). Au-delà de la reproduction de contenus auparavant sous forme d'artefacts physiques, on constate que le phénomène s'entretient : une part de plus en plus importante du quotidien d'un humain passe par un ordinateur, mobile ou non, démultipliant la quantité d'informations disponibles. Ce processus, parfois qualifié d'« ère du Big Data » est favorisé par une interconnexion permanente des réseaux de communication, et l'accroissement du nombre de capteurs (capables d'enregistrer une géolocalisation ou diverses informations). Bien sûr, ces données ne sont pas uniquement *produites* et *collectées*, elles sont aussi *analysées*. Dans des systèmes destinés à la consultation par des utilisateurs, un *tri automatisé* de l'information est ainsi d'une grande aide pour améliorer la présentation aux utilisateurs : ce sont les systèmes de recommandation qui réalisent ce *filtrage*. Leurs succès, techniques et commerciaux, ont progressivement encouragé de nombreuses sociétés à introduire des recommandations dans leur système. On trouve aujourd'hui des recommandations algorithmiques dans de très nombreux domaines : médias et informations (Google Actualités[2]), plateformes culturelles (musique[3], films et séries[4]), vente en ligne, moteurs de recherche, etc.

Ainsi, en quelques années, ces systèmes algorithmiques en sont venus à occuper une place centrale dans la hiérarchisation et l'organisation des informations numériques. Auparavant, les moyens de communications qui transmettaient des messages (téléphone, forums, e-mails) étaient « neutres », proposant éventuellement une organisation chronologique. Les systèmes de recommandations introduisent un filtrage différent, personnalisé, selon des formules généralement inaccessibles à l'utilisateur, à la fois parce qu'elles sont complexes à détailler mais aussi parce que les plateformes n'offrent pas d'accès à ces règles. Leur place prépondérante amène donc une vaste question : quelles sont les conséquences, individuelles et collectives de ces choix algorithmiques ?

Les inquiétudes venues de la société civile comme du champ académiques sont nombreuses, et anciennes. Jacques Ellul avertissait, dès 1954, que « la technique est [...] une réduction au schéma logique des faits, des pulsions, des phénomènes, des moyens, des instruments » [Ell54]. Éric Sadin, plus récemment, regrettait que ce « biotope numérique [a] contribué à instaurer une technogouvernementalité de la vie, [...] perturb[ant] le champ d'action de la politique institutionnelle et infléchissant la nature de nos comportements » [Sad15]. Sans céder au catastrophisme ou à l'exagération, nous pouvons nous en tenir aux effets de la recommandation sur la transformation des modes de production et de circulation de l'information. Depuis le début des années 2010, le débat est vif sur la notion de « bulle de filtre » (et de « chambre d'écho ») [Par11 ; RMM20 ; And+20 ; Möl+20 ; BMA15] : est-ce que les algorithmes ont tendance à enfermer progressivement les utilisateurs dans un « profil », ne leur montrant que des contenus proches de ce qu'ils ont (supposément) aimé par le passé ? Si c'est confirmé, cela peut avoir de lourdes conséquences, à l'heure où de nombreuses personnes s'informent prioritairement via des fils d'actualités recommandées, parfois contenant des informations non validées. Avec David Chavalarias, nous pouvons ainsi prendre l'exemple marquant de la circulation de fausses informations, ou *fake news*, sur le réseau social Twitter, qui a culminé le

[4]Voir https://news.google.com.

[4]Voir Spotify http://www.spotify.com ou Deezer http://www.deezer.com par exemple.

6 janvier 2021 en une tentative de prise d'assaut du Capitole par de nombreux émeutiers menaçant de renverser une élection légitime dans l'une des plus grandes démocraties du monde [Cha23].

La question sous-jacente est celle de la *diversité*, ici centrée sur l'individu. Il y a également des enjeux collectifs à cette question : est-ce qu'il existe des objets qui ne sont jamais recommandés ? Est-ce que les systèmes de recommandations favorisent, sciemment ou non, l'émergence d'"objets stars", systématiquement mis en avant ? [Cha12] Malgré des études préliminaires, les effets à long terme de ces systèmes sur la diversité et la popularité des objets sont encore mal connus [Gil14 ; CP07 ; PM12 ; SDW06] et leur généralisation pourrait avoir des conséquences notables, par exemple sur l'étendue de la culture générale de la population (si quelques distributeurs en situation de monopole ne diffusent qu'un nombre restreints d'œuvres) voire sur les idées en général : un moteur de recherche comme Google représente selon les pays jusqu'à 90% des requêtes, ses biais de filtrage ont un impact certains sur ce qui est vu, lu et consommé.

La Technique est éminemment politique : si elle peut permettre à l'être humain de maîtriser son environnement, elle peut aussi permettre de dominer, de contrôler d'autres êtres humains (ou êtres vivants). Quels sont les desseins, les motivations de ceux qui conçoivent et contrôlent des systèmes de recommandation ? Les enjeux sociétaux sont très importants, pour garantir le respect de la sécurité et de la vie privée, mais aussi pour permettre aux systèmes législatifs et judiciaires, ainsi qu'aux citoyens, de s'adapter aux évolutions techniques.

Les systèmes de recommandation peuvent être considérés comme des réseaux, constitués d'interactions, entre utilisateurs et objets, voire entre utilisateurs et utilisateurs. Il est possible de modéliser ces systèmes par un ou plusieurs graphes. J'ai commencé mon parcours de recherche dans une équipe de spécialistes de la science des réseaux, et j'ai découvert l'étude des systèmes de recommandation lors de mon post-doctorat à l'Université Paris 13, auprès d'Emmanuel Viennet, en collaborant avec Françoise Soulié-Fogelman [Ber+14]. Mes travaux poursuivent, depuis lors, deux ambitions :

- contribuer significativement à l'effort de développement des systèmes de recommandation incorporant des informations « sociales », c'est-à-dire reposant sur des modélisations avec des graphes ;
- mettre à profit les structures de graphes mobilisées pour comprendre et critiquer les effets des systèmes de recommandation, expliquer leurs suggestions, évaluer leur impact.

Il me semble qu'on peut mettre cette dualité en regard avec ces paroles de Donna Haraway : « Il faut faire de la critique [de la technique], mais la critique n'est pas le travail fondamental. La critique clarifie et met en lumière les processus de domination […]. Mais le travail à faire est [aussi] un travail de *co-engendrement.* » [Por23]

Mon approche de la recommandation est donc originale dans la mesure où je n'ai travaillé qu'avec des modélisations « graphes », « graphiques ». D'autre part, si je me suis intéressé de près à l'évaluation des systèmes de recommandation, en particulier sous l'angle de la diversité comme nous le verrons dans le chapitre 2, j'ai toujours utilisé des données publiques, des *benchmarks* standards de la communauté (MovieLens, par exemple). D'autres chercheurs, notamment du secteur privé, emploient parfois des données collectées, ou créent leurs jeux de données. Enfin, j'ai abordé la recommandation comme une tâche générique, et les études que j'ai menées concernent des cas

classiques de recommandation, à la fois en terme de domaine (recommandation de produits culturels, notamment les films Movielens, donc) et de tâche abordée (essentiellement de la recommandation top-k). Les études menées ont été, comme tout travail de recherche, très influencées par le contexte d'exercice (institution, financement, habitudes locales) : un autre contexte aurait permis des avancées différentes.

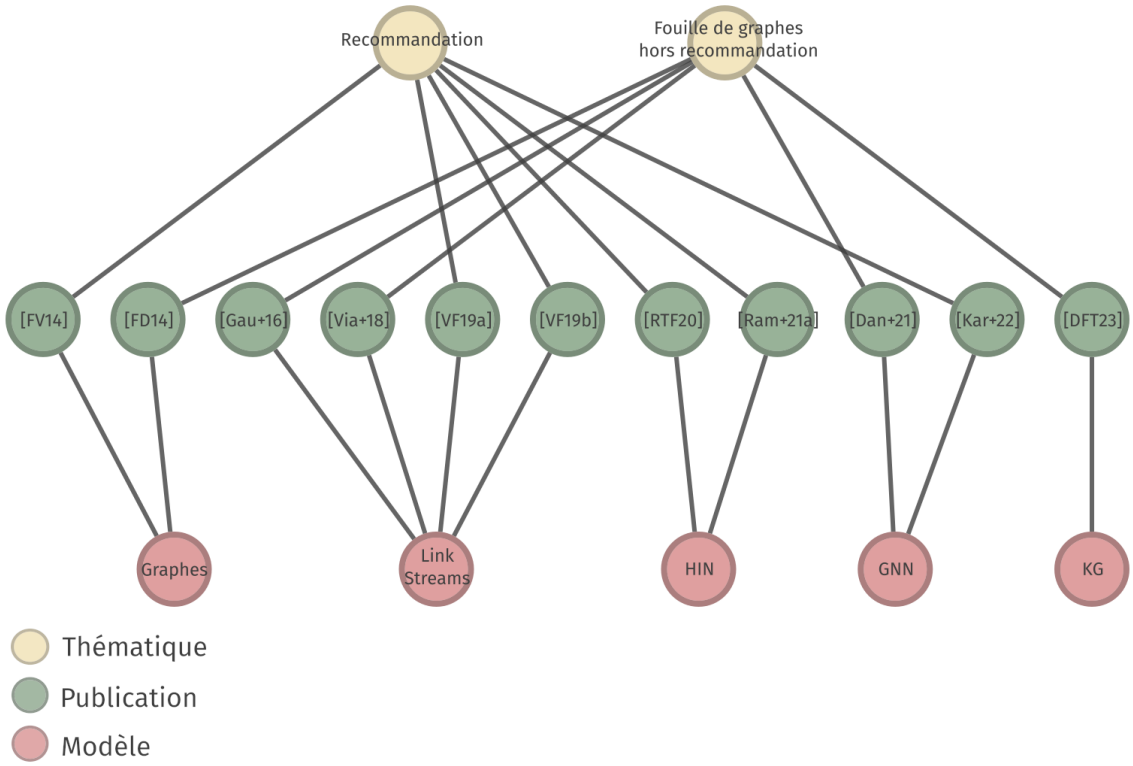


Figure 1.1 : *Graphes triparti de mes publications en fouille de données. Les articles (en vert) sont ordonnés chronologiquement de gauche à droite.*
GNN : Graph Neural Networks
HIN : Heterogeneous Information Networks
KG : Knowledge Graphs

La figure 1.1 illustre mon parcours de publication dans le domaine[5], en mettant en évidence les thématiques auxquelles mes articles sont liés (en particulier, la recommandation), et les modèles de graphes que j'ai explorés et étudiés. Diverses publications sont en dehors du champ de la recommandation, mais ces travaux sur des modèles de graphes ont régulièrement été suivis (ou précédés) d'une application au domaine de la recommandation. Nous n'en détaillons pas les contributions dans ce manuscrit, mais cela n'enlève évidemment rien à leur intérêt. On pourra noter que je n'ai pas (encore ?) employé de *knowledge graphs* pour la recommandation, ce qui a déjà été exploré dans [Hub+22 ; HBM22]. Les contributions dans ces domaines sont plus éparses, moins unifiées que celles de la première partie, et j'ai donc fait le choix d'en présenter deux plus en profondeur dans les

[5]Bien entendu, cette analyse rapide est avant tout un prétexte pour créer un graphe sur des articles à propos de graphes…

chapitres suivants, laissant de côté certains pans de mes travaux. En particulier, mes contributions avec le modèle flots de liens (*link stream*) ne sont pas abordées.

Dans les deux prochains chapitres, nous verrons les contributions en recommandation avec des graphes. Elles seront organisées selon les deux axes mentionnés *supra*. Dans le chapitre 2, nous détaillons comment, à l'aide du modèle des Heterogeneous Information Networks, nous avons établi une taxonomie approfondie de la notion de *mesure de diversité* [Ram+21a]. Ces travaux peuvent aider la *critique* des systèmes, en permettant d'avoir un regard différent sur leurs performances. Dans le chapitre 3, nous présentons des travaux centrés sur l'apprentissage de représentations avec le modèle Graph Neural Networks, en proposant des pistes d'améliorations [Kar+22]. On cherche alors à poursuivre l'effort de développement de ces systèmes, pour mieux les connaître et les comprendre.

1.3 État de l'art

1.3.1 Recommandation

Un processus de recommandation est généralement décomposé en plusieurs étapes génériques, partagées par tous les systèmes :

1. analyse de l'information sur l'utilisateur : son historique d'interactions dans le système, mais aussi ses caractéristiques externes éventuellement disponibles. On crée ainsi un *profil utilisateur* ;
2. calcul de correspondances, entre les éléments et ce profil utilisateur. Les éléments jugés les meilleurs sont sélectionnés et affichés à l'utilisateur ;
3. collecte d'un retour de l'utilisateur sur les éléments du système, recommandés ou non (pour enrichir son profil).

Les approches pour réaliser algorithmiquement de la recommandation (étapes 1 et 2 ci-dessus) sont nombreuses, mais se regroupent généralement en *filtrage par le contenu* et *filtrage collaboratif*. Il existe, depuis assez longtemps, des approches hybrides, mêlant les deux [Bur02]. L'approche par le contenu repose sur la création de caractéristiques (*features*) à partir des données sur les utilisateurs et les items[6], puis l'on cherche, avec des techniques d'apprentissage ou de recherche d'information, à rapprocher les caractéristiques représentant les utilisateurs des caractéristiques des items, à calculer des similarités. Le filtrage collaboratif repose sur les corrélations entre les notes des utilisateurs ou items du système : pour un produit non vu par un utilisateur, on peut estimer son avis à partir des avis des autres utilisateurs, en pondérant cela en fonction des profils de notes. Les techniques de factorisation de matrices sont extrêmement efficaces dans cette approche. Historiquement, des méthodes dites « à voisinage » (*neighborhood-based*) ont été mises au point

[6]Nous utilisons dans ce manuscrit indifféremment *élément*, *article* ou *item* pour désigner les objets d'un système de recommandation.

pour faire de la recommandation. Cependant, ce sont avant tout des méthodes travaillant sur les proximités entre lignes ou colonnes d'une matrice, voire des blocs. La liaison « graphe-matrice » est presque immédiate, puisque la matrice des interactions d'un système de recommandation peut être considérée comme étant la matrice d'adjacence d'un graphe. Cependant, les méthodes d'analyse de graphes, avec des indicateurs de centralité comme PageRank ou Katz, des marches aléatoires, des projections du graphe biparti, etc., sont historiquement moins mobilisées dans la littérature que leurs pendants matriciels.

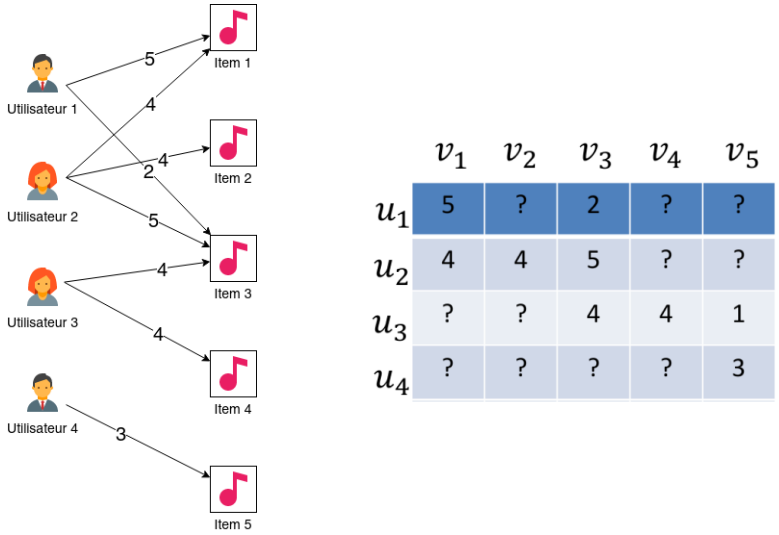


Figure 1.2 : *Modélisations d'un système de recommandation avec 4 utilisateurs et 5 éléments. À gauche, par un graphe biparti. À droite, par une matrice. Les cases remplies de la matrice correspondent à des avis normalisés des utilisateurs sur les éléments.*

La figure 1.2 illustre un système de recommandation. Les interactions des utilisateurs avec les produits forment un graphe, biparti. Une ligne de la matrice caractérise le profil d'un individu, par les appréciations qu'il a donné aux éléments du système (ou leur simple consultation, selon les cas). On utilise en pratique de nombreuses matrices pour un système de recommandation donné (par exemple : achat, temps passé sur la page, notes), mais on illustre généralement le principe sur une seule.

Au cours de sa riche histoire, la communauté « recommandation » a identifié et abordé de nombreuses tâches, avec des défis associés, et il serait vain de prétendre être ici exhaustif. Nous pouvons toutefois en lister quelques unes, et l'on se réfèrera aux manuels [RRS11 ; Agg16 ; McA22] pour des études complètes.

La recommandation est alimentée par des *données*, et il faut donc en collecter, ce qui pose la question des incitations (monétaire ou par l'IHM) pour que les utilisateurs fassent ce travail (de donner *explicitement* de l'information). La récupération d'informations *implicites* est aussi possible, mais elles sont davantage bruitées et requièrent des traitements plus complexes. Le démarrage à froid (*cold start*, quand un utilisateur arrive dans le système et qu'il y a peu d'informations sur lui) et la *data sparsity* (quand les données collectées sont trop peu nombreuses) sont des champs de recherche actifs, proposant diverses techniques pour pallier ces problèmes.

Les tâches de recommandation sont variées, dépendant du système dans lequel elles prennent place. Outre les tâches classiques de prédiction de notes et de recommandation top-k déjà mentionnées, il existe par exemple la recommandation séquentielle, qui produit des enchaînements de chansons dans une « radio » sur Spotify, et requiert des techniques spécifiques pour trouver le bon titre à diffuser à l'auditeur. La recommandation *session-based* ne considère pas les préférences à long terme d'un utilisateur mais seulement les informations récentes, ce qui marche bien dans un contexte de commerce en ligne. La recommandation contextuelle essaie de tirer partie d'informations spécifiques (comme l'heure dans la journée ou la géolocalisation de la personne) pour adapter les suggestions. La recommandation groupée, plus rare, sert à proposer des items culturels à des groupes d'utilisateurs (par exemple, une famille visitant un musée, ou des amies voulant choisir un film pour aller au cinéma). Pour les tâches classiques comme pour les moins communes, la construction des profils utilisateurs et des correspondances avec ceux des items sont centrales, et il existe de nombreuses sous-communautés, selon les modèles employés. Nous présenterons en détail les plus proches de mes travaux dans la section 1.3 de ce chapitre.

Enfin, l'un des enjeux importants de l'étude des systèmes de recommandation réside dans l'évaluation des performances. On travaille avec des indicateurs comme la (R)MSE (*(Root) Mean Squared Error*), qui mesure l'erreur de prédiction dans la note d'un utilisateur pour un élément, l'AUC (*Area under roc curve*), la MAP (*Mean average precision*) ou le NDCG, qui elles mesurent davantage la qualité d'un ensemble d'éléments ordonnés [Her+04]. Il existe bien d'autres indicateurs qui caractérisent un système de recommandation : la nouveauté (est-ce que les éléments présentés sont nouveaux pour l'utilisateur ?), la couverture (quelle proportion des utilisateurs peuvent recevoir des recommandations ?), et la diversité (est-ce que l'utilisateur voit toujours le même type de suggestions ?). Nous discuterons en détail cette mesure dans le chapitre 2. La standardisation des mesures permet également d'établir des comparaisons fiables entre les différents algorithmes proposés : avec des jeux de données appropriés, cela permet d'établir des protocoles d'étalonnage (*benchmarking*). On utilise fréquemment des données cinématographiques venant du projet MovieLens[7].

1.3.2 Modèles de graphes

Avant de présenter des éléments d'état de l'art de la recommandation avec des graphes, nous introduisons ici deux modèles de graphes utilisés dans les chapitres 3 et 2.

1.3.2.1 *Heterogeneous Information Networks*

On définit généralement un graphe $\mathcal{G} = (V, E)$ par un ensemble V de sommets (ou nœuds, « V » pour *vertices*) et d'arêtes (ou liens, « E » pour *edges*, $E \in x, y|(x, y) \in V^2$). Un *Heterogeneous Information Network* (HIN) $\mathcal{H} = (\mathcal{G}, \mathcal{A}, \mathcal{R}, \varphi, \psi)$ est un graphe orienté $\mathcal{G} = (V, E)$ avec des fonctions associant les nœuds et les arêtes à des types, respectivement $\varphi : V \to \mathcal{A}$ et $\psi : E \to \mathcal{R}$. Les HINs peuvent représenter différents sortes de nœuds : certains représentent des utilisateurs, d'autres des éléments, d'autres des types (du système considéré). Les nœuds de différentes sortes sont reliés par

[7]Voir http://www.grouplens.org/node/73.

des relations spécifiques, telles que « un utilisateur a vu un élément » ou « un élément appartient à un genre ». Le *schéma* d'un HIN $\mathcal{H} = (\mathcal{G}, \mathcal{A}, \mathcal{R}, \varphi, \psi)$ est un graphe orienté, dont les nœuds sont les types $\mathcal{A}$ et les arêtes sont les relations $\mathcal{R}$. Un exemple simple est illustré sur la Figure 1.3.

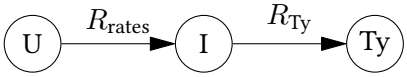


Figure 1.3 : *Un schéma de HIN où des utilisateurs (U) évaluent des items(I) qui sont typés (Ty), définissant les types d'arêtes R_{rates} et R_{Ty}. Cela correspond évidemment à la modélisation d'un système de recommandation.*

Un groupe de liens, représenté par une arête sur le schéma du HIN, $R \in \mathcal{R}$, est un sous-ensemble d'arêtes de E de type R, liant un un ensemble d'objets-sources $S \in \mathcal{A}$ à des objets-cibles $T \in \mathcal{A}$. On le note R, et son inverse (composé des arêtes de directions inversées) R^{-1}. Un méta-chemin dans un HIN est un chemin sur son schéma. On le note $\Pi = A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \dots \xrightarrow{R_l} A_{l+1}$, ou simplement $\Pi = R_1 R_2 \cdots R_l$. Sur la Figure 1.3, $\Pi = R_{rates} R_{Ty}$ est un méta-chemin connectant les utilisateurs aux types des éléments avec lesquels ils ont interagi.

1.3.2.2 *Graph Neural Networks*

Les *Graphs Neural Networks* (GNN) sont des architectures de réseaux de neurones qui ont été inventées pour permettre de transférer les (efficaces) méthodes d'apprentissage profond aux données graphes. On souhaite apprendre des représentations des nœuds qui dépendent de la structure du graphe (et des éventuelles caractéristiques de ces nœuds), afin d'éviter de faire du *feature engineering* pour chaque tâche d'analyse sur le graphe, un processus généralement coûteux (en temps) et délicat (pléthore de méthodes existent et il n'est pas évident de sélectionner la plus adaptée).

L'apprentissage de représentations dans un GNN repose sur le passage de messages entre nœuds. Dans chaque couche, une représentation cachée h_u^k du nœud $u \in V$ est mise à jour avec les informations agrégées à partir du voisinage de u. On peut formaliser ainsi cette mise à jour :

$$h_u^{(k+1)} = \text{UPDATE}^{(k)}(h_u^k, \text{AGGREGATE}^{(k)}(\{h_v^k, v \in N(u)\})).$$

k est l'indice d'une couche, $N(u)$ désigne le voisinage de u dans le graphe. Les fonctions UPDATE et AGGREGATE sont des fonctions différentiables, c'est-à-dire des réseaux de neurones, qui varient suivant les architectures. L'agrégation prend un ensemble de vecteurs (dans un ordre indéterminé) et retourne un vecteur unique : on peut utiliser diverses moyennes ou sommes, voire de l'*attention* (c'est-à-dire d'apprendre des coefficients différents selon les voisins du nœud). La fonction d'UPDATE est aussi importante, puisqu'elle préserve l'information venant du nœud lui-même, de couche en couche. La figure 1.4 présente un graphe d'entrée et les calculs qu'une architecture GNN effectue sur le graphe.

1.3.3 Recommander avec des graphes

La vision orientée « graphes » du filtrage collaboratif est apparue en 1999 [Agg+99]. La plupart des méthodes qui ont suivi se sont focalisées sur les cas des réseaux sociaux (réseaux de confiance,

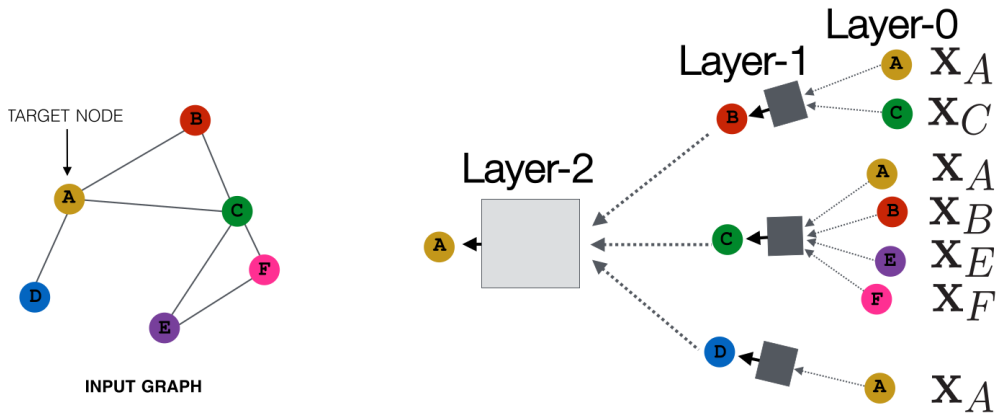


Figure 1.4 : *Un graphe et une représentation des calculs GNN sur ce graphe : les représentations initiales X_A, X_C des nœuds A et C sont agrégées et la mise à jour utilise aussi $h_B^{(0)} = X_0$ pour calculer l'*embedding *de B à la couche 1, $h_B^{(1)}$. De même, $h_A^{(2)}$ est calculé avec $h_A^{(1)}$ et le résultat de l'agrégation de $h_B^{(1)}$, $h_C^{(1)}$ et $h_D^{(1)}$. Source : [Les23].*

par exemple [Gol05 ; THL13]). Il a été proposé de compléter le filtrage collaboratif classique avec des techniques centrées sur les (plus courts) chemins ou des marches aléatoires dans le graphe pour évaluer les similarités [DK11]. La détection de communautés est un problème classique d'étude des graphes [Blo+08] mais n'a pas beaucoup d'écho dans le contexte de la recommandation (contrairement à ce qu'on pourrait attendre). L'absence de consensus sur la détection de communautés dans un graphe biparti [Bar07 ; Mur09 ; Bec16] est peut-être à l'origine de cela.

Des modélisations reposant sur des « graphes étendus », comme les Heterogeneous Information Networks (HIN) ont été proposés dans la dernière décennie [BVM14 ; Shi+16]. L'avantage est de permettre de préserver de l'information sémantique sur le système, dans la structure même du modèle. On peut en tirer partie pour la conception d'algorithmes de recommandation. Ces modèles ont montré qu'ils pouvaient avoir de bonnes performances, et faciliter l'explicabilité des résultats [Shi+15 ; Shi+18]. En outre, les HIN s'adaptent bien dans des contextes de démarrage à froid (quand il y a peu d'informations sur les utilisateurs) [Tan+12], des contextes sociaux [YSL12] ou de l'information implicite seulement [Yu+14].

Si la diversité peut avoir de nombreux noms et de nombreuses formes, il y avait eu, avant nos travaux, assez peu d'approches tentant d'approcher la diversité des recommandations avec des modèles graphes [KJR14]. Les travaux de Kunaver et al. sont une référence majeure sur la diversité dans les systèmes de recommandation [KP17]. Certains travaux s'intéressaient à la traversée de graphes et à la diversité des types de nœuds rencontrés, mais peu exploitent la variété des relations entre nœuds de types différents. Dubey et al. [DCB11] font appel à la notion de marche aléatoire sur des chemins dans les HIN, de même que [NMM18 ; Jia+19]. Les mesures de diversité, telles que nous les étudions dans le chapitre 2, visent à estimer la (dis-)similarité des éléments qui sont recommandé dans un système. Certains auteurs considèrent que des éléments sont similaires s'ils ont été choisis par les mêmes utilisateurs [HZ11 ; ZCM02], d'autres s'ils sont du même type [Zie+05]. On parle fréquemment aussi de *nouveauté* ou de *similarité intra-liste* pour caractériser ces « divergences ». Certains travaux s'intéressent à la *surprise* que constitue une recommandation, par rapport aux choix passés d'un utilisateur, alors que d'autres regardent surtout la *personnalisation*, c'est-à-dire la différence des recommandations qui sont faites à un utilisateur par rapport à un autre [Zho+10]. Mesurer la diver-

sité à un intérêt applicatif, pour détecter par exemple des changements de préférences (le *concept drift*, [LCB16]). Mais l'intérêt de ces indicateurs réside aussi dans la quantification de phénomènes sociologiques comme la perte de diversité des « bulles de filtres » ou « chambre d'écho » [FH07 ; Par11].

Recommandation avec un HIN.

Un processus de recommandation consiste à proposer à un utilisateur des éléments qu'il n'a pas sélectionnés auparavant. Dans un HIN $\mathcal{H} = (\mathcal{G}, \mathcal{A}, \mathcal{R}, \varphi, \psi)$, cela revient à créer de nouvelles arêtes : $F : \mathcal{H} \mapsto E_{rec} \in V \times V$. L'ensemble des arêtes E_{rec} est composé des liens entre les utilisateurs et les éléments recommandés. On peut munir le HIN d'une relation R_{rec} telle que $\psi(e) = R_{rec}$ pour $e \in E_{rec}$, afin que les arêtes de recommandation soient utilisées dans les méta-chemins. L'intérêt de ce procédé est un peu le même que le calcul d'embeddings avec Deepwalk [PAS14] (précurseur des GNNs) : on calcule des marches aléatoires sur le graphe (ici, on les contraint à emprunter les méta-chemins), ce qui permet d'estimer des proximités (ou similarités) entre nœuds [Sun+11 ; XZY14 ; Yu+13 ; Shi+18]. Cela va nous permettre d'estimer la diversité des recommandations proposées, avec un *framework* graphique générique.

***Graph Neural Networks* et recommendation**

Les Graph Neural Networks ont été développés pour apprendre des représentations complexes sur des géométries non-euclidiennes comme les graphes [Vel+18 ; HYL17 ; KW17 ; Ros+20]. Des centaines d'articles sont publiés chaque trimestre en ce moment, pour améliorer itérativement les architectures de GNNs et leurs performances, donc cet état de l'art est partiel, proche des travaux que nous présentons plus loin.

Pour les systèmes de recommandation, les GNN offrent un moyen efficace d'apprendre les interactions entre utilisateurs et éléments, en tirant parti de la structure du graphe sous-jacent. Les modèles de GNN à passage de messages (MP-GNNs, pour *message-passing GNNs*) sont aujourd'hui aussi efficaces que les meilleures méthodes de factorisation de matrices pour de la recommandation top-k [He+20 ; Wan+19 ; Sun+20 ; Che+20b]. L'évaluation de cette recommandation top-k repose généralement sur des mesures fondées sur le classement, telles que la précision moyenne (Average Precision [CMS10]), le NDCG (Normalized Discounted Cumulative Gain [JK02]) ou le *rappel* à k (ou *recall@k*). Cependant, pour un processus d'apprentissage, ces indicateurs ont l'inconvénient de ne pas être différentiables (à cause du rang) et ne peuvent pas être directement utilisés (par exemple dans une descente stochastique de gradient). Les GNN sont donc généralement entraînés avec des fonctions de coût différentiables comme la fonction de perte *Bayesian Personalized Ranking* (BPR) [Ren+09]. La BPR est assez efficace, mais n'est pas conçue pour optimiser directement les indicateurs d'évaluation, et elle a tendance à mettre l'accent sur les erreurs en bas du classement, plutôt qu'en haut (ce qui n'est pas optimal pour de la recommandation) [Bro+20]. Plusieurs modèles ont été conçus ainsi : [Wan+19 ; Che+20b ; Wan+20]. Les techniques d'entraînement ont été progressivement améliorées, soit en modifiant la loss BPR [Wu+21a], soit en générant des (faux) exemples négatifs artificiellement plus délicats à distinguer des positifs [Hua+21], forçant le système à « mieux » apprendre.

L'étude des fonctions de pertes fondées sur le classement a été étudiée dans d'autres domaines que la recommandation, par exemple en recherche d'information avec des SVM structurés [Yue+07]. Plus récemment, l'optimisation directe des mesures d'évaluation a connu un regain de popularité dans le domaine de l'apprentissage profond [Bru+19 ; PB21 ; Bro+20 ; Ram+21b].

Chapitre 2

Diversité des recommandations

Éduque [ta fille Ijeawele] à la différence. Fais de la différence une chose ordinaire. Fais de la différence une chose normale. Apprends-lui à ne pas attacher d'importance à la différence. Et il ne s'agit pas là de se montrer juste ou même gentille, mais humaine et pragmatique. Parce que la réalité de notre monde, c'est la différence. Et en l'éduquant à la différence, tu lui donnes les moyens de survivre dans un monde de diversité.

Chimamanda Ngozi Adichie[1]

Sommaire

2.1 Un modèle de graphes pour évaluer la diversité **74**
2.1.1 Définitions 74
2.1.2 Les *vraies* diversités 75
2.1.3 Marches aléatoires et Heterogeneous Information Networks 77
2.2 Diversités individuelle et collective **77**
2.3 Expériences **80**
2.4 Conclusion **83**

Les systèmes de recommandation ont d'abord été pensés pour proposer à leurs utilisateurs et utilisatrices des suggestions qui leur conviendraient le mieux, à même de correspondre à leurs besoins ou leurs goûts. Cependant, on s'est progressivement aperçu que, paradoxalement, les résultats de ces recommandations optimisant ce critère n'étaient peut-être pas toujours prévisibles et désirables,

[1]Chimamanda Ngozi Adichie. *Chère Ijeawele, ou Un manifeste pour une éducation féministe.* Trad. par Marguerite Capelle. Paris : Gallimard, 2017.

à l'échelle individuelle comme collective (voir page 63). La diversification des recommandations est ainsi étudiée, depuis plusieurs années, pour comprendre les effets de ces principes, et les articuler avec les questions de gouvernance algorithmique.

Dans ce chapitre, nous proposons une étude technique, quantitative, du phénomène de diversité. Nous recourons à un framework reposant sur un modèle de graphes, les *Heterogeneous Information Networks*. Nous proposons des indicateurs[2], avec des propriétés spécifiques. Ce cadre permet d'unifier des résultats issus de différentes disciplines, et d'observer certains phénomènes jusque-là peu accessibles (ou, du moins, avec un regard différent).

2.1 Un modèle de graphes pour évaluer la diversité

2.1.1 Définitions

On considère un système avec un ensemble I d'éléments (*items*), un ensemble T de types, et une relation d'appartenance $\tau \subseteq I \times T$ qui indique comment les éléments sont classés dans ces types. Un élément $i \in I$ est du type $t \in T$ si et seulement si $(i, t) \in \tau$. Un élément peut appartenir à plusieurs types. La diversité peut être définie à partir de trois propriétés des systèmes comportant une classification d'éléments en types [Sti07] : la *variété* (le nombre de types d'items dans le système), l'*équilibre* (la distribution des items parmi les types) et la *disparité* (à quel point les types d'items diffèrent).

Une *mesure de diversité* est une fonction $D : I \times T \to \mathbb{R}^+$ qui associe à ce système une valeur réelle positive d, $D : \tau \mapsto d \in \mathbb{R}^+$. L'*abondance* d'un type $t \in T$ est le nombre d'éléments de ce type : $a_\tau(t) = |\{i \in I : (i, t) \in \tau\}|$, et son *abondance proportionnelle* est $p_\tau(t) = \frac{a_\tau(t)}{|\tau|}$. Les mesures de diversité D que nous considérons sont des applications de Δ^* dans $\mathbb{R}^+$, où $\Delta^* = \cup_{k \geq 0} \Delta^k$ est l'ensemble des distributions de probabilités sur les espaces discrets de taille $k + 1$:

$$\Delta^k = \left\{(p_1, \ldots, p_{k+1}) \in \mathbb{R}^{k+1} : \forall i \leq k,\ p_i \in [0, 1] \text{ et } \textstyle\sum_{i \leq k+1} p_i = 1\right\}.$$

Historiquement, selon les domaines, des indicateurs de diversité comme l'entropie de Shannon [Sha48b], l'index de Gini [Gin21] ou l'index d'Herfindahl-Hirschmann [Rho93] ont permis de quantifier la diversité. La richesse [Mac65 ; GC11] est communément utilisée en écologie, elle mesure simplement le nombre de types effectifs utilisés pour classer des éléments. Elle capture la notion de *variété* :

$$R(p) = |\{i \in \{1, 2, ..., |T|\} : p_i > 0\}|.$$

L'entropie de Shannon est liée aux propriétés de variété et d'équilibre du système. Elle caractérise l'incertitude de prédiction quant au type d'un élément tiré au hasard :

$$H(p) = -\sum_{i=1}^{|T|} p_i \log_2 p_i.$$

[2]Terminologie : je parle d'*indicateur* de diversité, mais utilise aussi abondamment le terme *mesure de diversité*, peut-être un peu trop calqué sur le terme anglais *measure*. Certains utilisent aussi *métrique*, calqué sur *metric*.

Nous pouvons illustrer ces concepts avec une bibliothèque contenant des ouvrages (les éléments) qui appartiennent à des genres littéraires (les types). Si une bibliothèque contient des romans, des bandes dessinées et des livres de voyage, alors sa richesse est de 3, quelles que soient les proportions des livres dans chaque catégorie. Si l'on connaît l'abondance proportionnelle des types (« genres ») de livres dans une étagère, l'entropie de Shannon est le nombre moyen de questions binaires (« est-ce un roman policier ? », « est-ce un dictionnaire ? ») à poser sur le type d'un élément pour obtenir la réponse.

2.1.2 Les *vraies* diversités

Il existe des fonctions mathématiques appelées *vraies diversités* (*true diversities*) qui sont connues depuis les années 1970 [Hil73 ; Jos06]. Ces fonctions appartiennent à la famille des *reciprocal self-weighted quasilinear means*, issues de travaux de Kolmogorov. On trouve des variantes dans différents domaines, comme l'indice de concentration de Hannah-Kay [HK77] ou l'entropie de Rényi d'ordre α en théorie de l'information [Rén61]. Elles peuvent être regroupées sous une formulation générique, dépendant d'un paramètre α :

Définition 1. *La vraie diversité d'ordre α, notée D_α, est l'application $D_\alpha : \Delta^* \to \mathbb{R}^+$, telle que, pour $p = (p_1, \ldots, p_k) \in \Delta^*$ et $\alpha \in \mathbb{R}^+$,*

$$D_\alpha(p) = \left(\sum_{i=1}^{k} p_i^\alpha\right)^{\frac{1}{1-\alpha}} \quad \textit{if } \alpha \neq 1, \quad \textit{et} \quad D_1(p) = \left(\prod_{i=1}^{k} p_i^{p_i}\right)^{-1}, \quad \textit{avec } {p_i}^{p_i} := 1 \textit{ si } p_i = 0.$$

La *richesse* d'une distribution p peut se définir comme la limite de $D_\alpha(p)$ quand $\alpha \to 0^+$. La richesse est donc la vraie diversité d'ordre 0. La vraie diversité d'ordre 1 $D_1(p)$, ou *diversité de Shannon*, est liée à l'entropie de Shannon $H(p)$: $D_1(p) = 2^{H(p)}$ (si l'entropie est en base 2). Enfin, avec $D_\alpha \xrightarrow{\alpha\to\infty} 1/\max\{p_1, \ldots, p_k\}$, on peut définir $D_\infty(p) := \frac{1}{\max\{p_1,\ldots,p_k\}}$, et conclure que $D_\infty(p) = 1/\mathrm{BPI}(p)$, si BPI désigne l'index de Berger-Parker [BP70]. Le tableau 2.1 résume ces relations entre des mesures connues et la famille des vraies diversités.

Ordre (α)	Nom	Vraie diversité	Expression	Relation aux autres mesures
0	Diversité de richesse	$D_0(p)$	$\lvert\{i \in \{1, \ldots, k\} : p_i > 0 \in\}\rvert$	Richesse [Mac65 ; GC11].
1	Diversité de Shannon	$D_1(p)$	$\left(\prod_{\substack{i=1 \\ p_i \neq 0}}^{k} p_i^{p_i}\right)^{-1}$	Exponentiation de l'entropie de Shannon [Sha48b ; Sha48a] : $H(p) = \log_2(D_1(p))$, with H in base 2.
2	Diversité de Herfindahl	$D_2(p)$	$\left(\sum_{i=1}^{k} p_i^2\right)^{-1}$	Réciproque de l'index de Herfindahl-Hirschman [Rho93] : $\mathrm{HHI}(p) = 1/D_2(p)$.
∞	Diversité de Berger	$D_\infty(p)$	$\left(\max_{i\in\{1,\ldots,k\}}\{p_i\}\right)^{-1}$	Réciproque de l'index de Berger-Parker [BP70] : $\mathrm{BPI}(p) = 1/D_\infty(p)$.

Tableau 2.1 : *Vraies diversités d'ordre 0, 1, 2, et ∞ et relations à des indicateurs classiques.*

La Figure 2.1 permet de mieux comprendre les propriétés de ces mesures de diversités :

- toutes les vraies diversités ont des valeurs identiques pour des distributions uniformes avec le même nombre de types effectifs (cette valeur est alors le nombre de types), ce sont les lignes horizontales.
- Quand la distribution en types n'est plus uniforme, ces mesures suivent des allures variées (courbes décroissantes). Le paramètre α indique comment l'équilibre (la non-uniformité) est pris en compte : quand α est faible, les inégalités d'abondances proportionnelles ne sont pas prises en compte. Quand α est grand, celles-ci ont un fort effet sur la diversité. Dans le cas extrême $\alpha \to \infty$ (c'est-à-dire pour la diversité de Berger), seule l'abondance la plus élevée est considérée.
- Les courbes rouge et bleu traduisent comment α peut moduler l'importance relative de la *variété* et de l'*équilibre* : une distribution à 6 types peut être considérée comme *moins* diverse qu'une autre à 4 types, si la valeur de α est judicieusement[3], choisie.

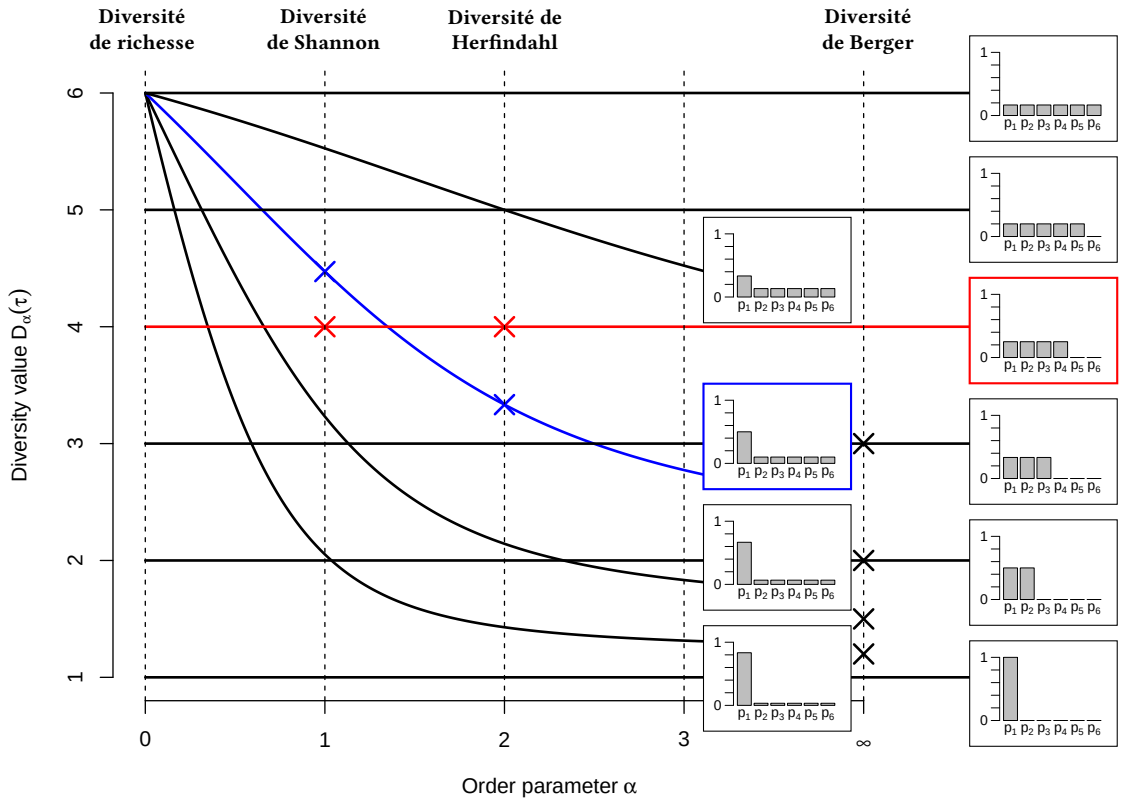


Figure 2.1 : *Les valeurs de plusieurs vraies diversités, en fonction d'α, pour différentes distributions.*

Les vraies diversités fournissent donc un continuum de mesures, pondérant la variété et l'équilibre des distributions. On peut les étendre, pour former la famille des *vraies diversités relatives*, avec laquelle il est possible de prendre pour référence une distribution autre qu'une distribution uniforme. Les vraies diversités relatives satisfont les mêmes propriétés que les vraies diversités.

2.1.3 Marches aléatoires et Heterogeneous Information Networks

Sur un graphe, on peut calculer de nombreuses distributions de probabilités, comme celles résultant de marches aléatoires sur les sommets du graphe. On peut ensuite calculer différentes mesures de diversité. Le framework que nous présentons dans cette section effectue simultanément les deux, le parcours du graphe et l'estimation de la diversité.

Nous considérons les marches aléatoires restreintes à des méta-chemins Π dans des HIN [LC10b; LC10a]. Ce sont des chaînes de Markov dont les probabilités de transition sont définies par $\Pr(X_i = v_i \mid X_{i-1} = v_{i-1}) = P_{E_{r_i}}(v_i \mid v_{i-1})$, pour $v_{i-1} \in V_{\text{src}}(E_{r_i})$ et $v_i \in V_{\text{dst}}(E_{r_i})$. Ce qui nous intéresse particulièrement dans notre contexte de recommandation est la distribution de probabilité de $V_{\text{dst}}(\Pi)$, le sommet d'arrivée d'une telle marche aléatoire. On définit aussi la distribution de probabilités conditionnelle sur un méta-chemin.

Définition 2. *Soit un méta chemin $\Pi = (E_{r_1}, \ldots, E_{r_k})$ de longueur k et une variable aléatoire $X_0 \in V_{src}(\Pi)$ représentant la position de départ d'une marche aléatoire sur les sommets $V_{src}(\Pi)$, la* marche aléatoire restreinte à Π *est la suite de $k+1$ variables aléatoires $(X_0, X_1, \ldots, X_k)$ résultant des transitions aléatoires entre les sommets de Π :*

$$X_0 \xrightarrow{E_{r_1}} X_1 \xrightarrow{E_{r_2}} X_2 \xrightarrow{E_{r_3}} \cdots \xrightarrow{E_{r_k}} X_k,$$

où, pour tout i, $X_i \in V_{dst}(E_{r_i})$.

Définition 3. *La distribution de probabilité conditionnelle de $X_k \in V_{dst}(\Pi)$, c'est-à-dire le sommet d'arrivée de la marche aléatoire restreinte à Π, sachant qu'elle a commencé en $v_0 \in V_{src}(\Pi)$ (donc $X_0 = v_0$), est noté $p_\Pi(v_k \mid v_0)$ pour $v_k \in V_{dst}(\Pi)$ et peut se calculer récursivement comme suit :*

$$\begin{aligned} p_\Pi(v_k \mid v_0) &= \Pr(X_k = v_k \mid X_0 = v_0) \\ &= \sum_{v_1 \in V_{dst}(E_{r_1})} p_{(E_{r_2}, \ldots, E_{r_k})}(v_k \mid v_1)\, p_{E_{r_1}}(v_1 | v_0). \end{aligned}$$

On note $p_{\Pi|v_0}(v_k)$ la distribution $p_\Pi(v_k|v_0)$ sur les sommets de $V_{dst}(\Pi)$.

2.2 Diversités individuelle et collective

Nous présentons ci-dessous comment définir des diversités individuelle et collective dans notre framework. Le sens de ces diversités sera variable, en terme applicatif, selon ce que représentera le HIN sous-jacent. La navigation dans le HIN pourrait également permettre de définir d'autres diversités, plus complexes et peut-être plus délicates à interpréter, comme des diversités *rétrograde* ou *projetée* [Ram+21a]. Ces diversités sont liées à la fonction de masse sur les nœuds de $V_{\text{end}} = V_{\text{dst}}(\Pi)$, à partir d'une marche aléatoire partie d'un nœud de $V_{\text{start}} = V_{\text{src}}(\Pi)$ et selon un méta chemin du HIN.

La diversité collective V_{end} de sommets V_{start} le long du méta chemin Π est la vraie diversité de la distribution de probabilités des sommets V_{end} qui résulte de la marche aléatoire débutée sur un sommet aléatoire de V_{start} et restreinte au méta chemin Π.

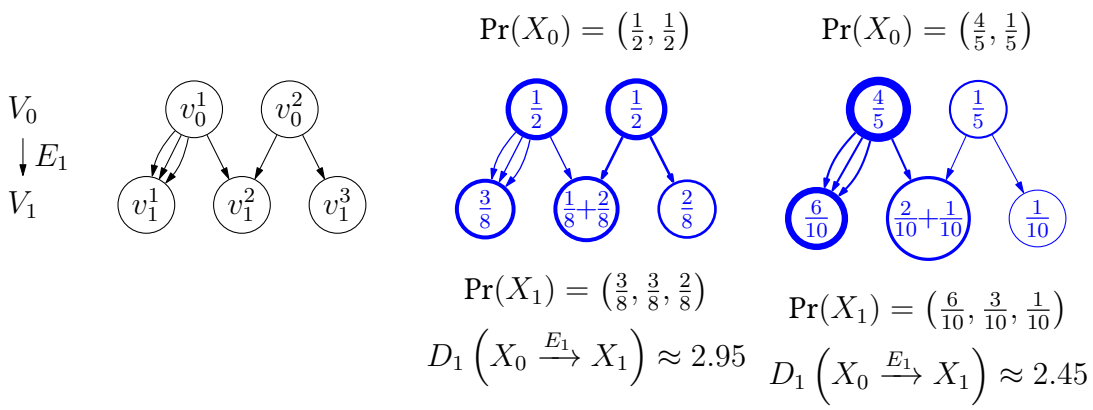


Figure 2.2 : *Calculs de diversité collective de V_1 depuis V_0 sur des méta-chemins.*

Définition 4. *Étant donnée la distribution de probabilités* $\Pr(X_0)$ *de partir d'un nœud au hasard* $X_0 \in V_{start}$*, on définit la* diversité collective V_{end} de V_{start} selon Π *comme étant la vraie diversité de la distribution de probabilité du sommet d'arrivée de la marche aléatoire restreinte. On la note* $D_\alpha\left(X_0 \xrightarrow{\Pi} X_k\right)$.

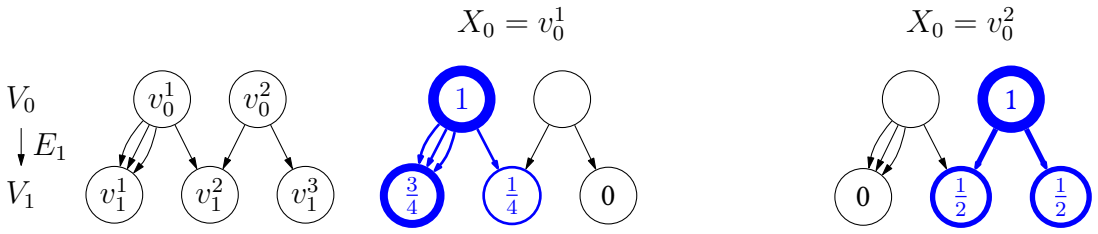
Cette diversité collective dépend de la distribution de probabilité initiale $\Pr(X_0 = v_0)$ et des probabilités de transition $p_{E_{r_i}}(v_i \mid v_{i-1})$ pour chaque $E_{r_i} \in \Pi$. La figure 2.2 illustre un exemple de cette diversité collective sur un HIN de 5 nœuds et 6 arêtes, pour deux distributions initiales $\Pr(X_0)$. Il y a deux types de sommets $V_0 = \{v_0^1, v_0^2\}$ et $V_1 = \{v_1^1, v_1^2, v_1^3\}$ et un seul type d'arêtes E_1, de V_0 vers V_1.

Les probabilités conditionnelles des marches aléatoires sur des méta-chemins permettent de capturer l'information sur la structure du réseau et les nœuds atteignables à partir de V_{start}. En particulier, on définit une diversité individuelle V_{end} depuis v_0 le long de Π. La Figure 2.3 illustre le calcul de cette diversité individuelle pour deux nœuds de V_{start}.

Définition 5. *À partir d'un nœud initial* $v_0 \in V_{start}$*, on appelle* diversité individuelle de V_{end} depuis v_0 selon Π *la vraie diversité de la distribution de probabilité des nœuds d'arrivée de la marche aléatoire restreinte à* Π*. On la note* $D_\alpha\left(X_0 \xrightarrow{\Pi} X_k \mid X_0 = v_0\right)$*, elle se calcule comme suit :* $D_\alpha\left(X_0 \xrightarrow{\Pi} X_k \mid X_0 = v_0\right) \quad = \quad D_\alpha(p_{\Pi|v_0})$.

On s'intéresse aussi à l'agrégation de ces diversités individuelles, en calculant une diversité individuelle moyenne, moyenne géométrique des diversités individuelles (d'un ensemble de nœuds donné).

Définition 6. *Pour un type de nœuds* V_{start}*, on définit la* diversité individuelle moyenne de V_{end} depuis V_{start} selon Π *comme suit :* $D_\alpha\left(X_0 \xrightarrow{\Pi} X_k \mid X_0\right) \quad = \quad \prod_{v_0 \in V_{start}} D_\alpha(p_{\Pi|v_0})^{\Pr(X_0 = v_0)}$



$$\Pr(X_1 \mid X_0 = v_0^1) = \left(\tfrac{3}{4}, \tfrac{1}{4}, 0\right) \qquad \Pr(X_1 \mid X_0 = v_0^2) = \left(0, \tfrac{1}{2}, \tfrac{1}{2}\right)$$

$$D_1\left(X_0 \xrightarrow{E_1} X_1 \mid X_0 = v_0^1\right) \approx 1.75 \quad D_1\left(X_0 \xrightarrow{E_1} X_1 \mid X_0 = v_0^2\right) = 2$$

$$\begin{array}{lclcl}
\Pr(X_0) = \left(\frac{1}{2}, \frac{1}{2}\right) & \Rightarrow & D_1\left(X_0 \xrightarrow{E_1} X_1 \mid X_0\right) & \approx & (1.75)^{\frac{1}{2}}\,(2)^{\frac{1}{2}} \approx 1.87 \\
\Pr(X_0) = \left(\frac{4}{5}, \frac{1}{5}\right) & \Rightarrow & D_1\left(X_0 \xrightarrow{E_1} X_1 \mid X_0\right) & \approx & (1.75)^{\frac{4}{5}}\,(2)^{\frac{1}{5}} \approx 1.80
\end{array}$$

Figure 2.3 : *Exemple de calculs de diversité individuelle et diversité individuelle moyenne.*

La moyenne des diversités individuelles dépend évidemment de $\Pr(X_0)$, et elle est minimale (elle vaut alors 1) lorsque toutes les diversités individuelles le sont. La figure 2.3 illustre quelques calculs simples de cette mesure.

Ces diversités individuelle et collective sont des mesures complémentaires, qui capturent différentes propriétés d'un système. Comme l'illustre la Figure 2.4, il est possible d'avoir une diversité individuelle moyenne qui soit basse, tout en ayant une diversité collective élevée (en haut à droite), ou l'inverse (en bas à gauche).

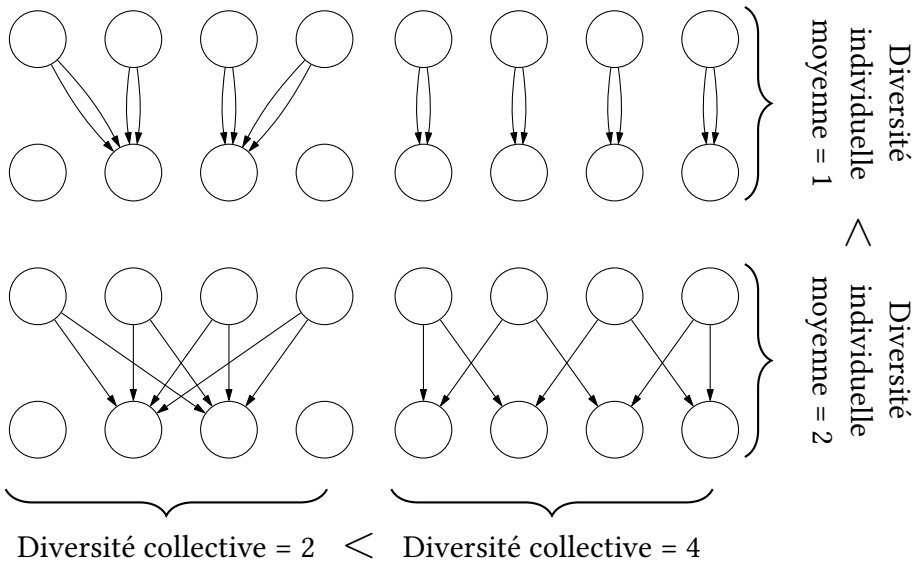


Figure 2.4 : *Différents HIN avec 2 types de sommets et illustrant valeurs relatives des diversités « collective » et « individuelle moyenne ».*

2.3 Expériences

Les mesures de diversité que nous avons introduites précédemment visent à évaluer la diversité dans des systèmes de recommandation. Nous présentons ci-dessous quelques expérimentations simples pour montrer comment la topologie d'un HIN a un impact sur la diversité des recommandations. Pour cela, nous calculons les probabilités résultant de marches aléatoires sur des méta-chemins, en partant d'un utilisateur et arrivant sur un ensemble d'éléments. On recommande les éléments avec la plus grande probabilité (et qui n'ont pas encore été vus/choisis), avec l'idée que la marche aléatoire est une approximation de la proximité (et donc des préférences de l'utilisateur) [NMM18 ; Jia+19]. On peut pondérer certaines parties du HIN, pour accorder par exemple un poids fort au méta-chemin relatif aux informations démographiques sur les utilisateurs et une faible importance aux informations relatives aux types des *items*, ou l'inverse.

Formellement, on considère K méta-chemins différents Π_k (avec $k = 1, \ldots, K$) commençant à $S \in \mathcal{A}$ et terminant à $T \in \mathcal{A}$ (pour *source* et *target*). On se donne les paramètres $\alpha_1, \ldots, \alpha_K$, tels que $\sum \alpha_k = 1$ et $0 \leq \alpha_k \leq 1$. Pour chaque $s \in S$ (typiquement, un utilisateur), on calcule un score $\mathcal{S}$ qui est une fonction de masse sur T (typiquement, l'ensemble des éléments du système) :

$$\mathcal{S}(t|s) = \sum_{k=1}^{K} \alpha_k p_{\Pi_k}(s), \text{ pour } t \in T. \tag{2.1}$$

Avec ce score $\mathcal{S}(t|s)$, on recommande les éléments les mieux classés de T.

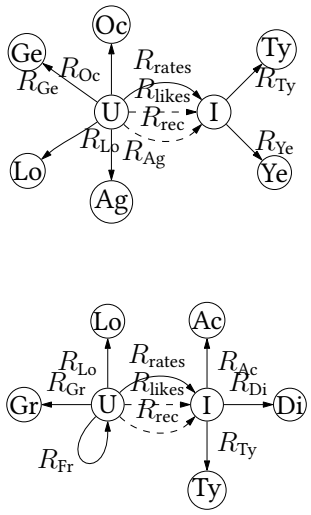
Tableau 2.2 : *Données ML100K et Douban Movie.*

MovieLens 100K (ML100K)					
Relation	Groupe source	Taille	Groupe cible	Taille	Liens
R_{Ty}	film	1664	genre	19	2863
R_{Ye}	film	1664	année	72	1664
R_{rates}	utilisateur	943	film	1664	99965
R_{Oc}	utilisateur	943	profession	21	943
R_{Ag}	utilisateur	943	âge	61	943
R_{Ge}	utilisateur	943	genre	2	943
R_{Lo}	utilisateur	943	ville	795	943
R_{likes}	utilisateur	943	film	1664	82495

Douban Movie (DM)					
Relation	Groupe source	Taille	Groupe cible	Taille	Liens
R_{Ac}	film	6971	acteur	3004	15584
R_{Di}	film	6971	réalisateur	789	3314
R_{Ty}	film	6971	genre	36	15598
R_{Gr}	utilisateur	3022	groupe	2269	7054
R_{Lo}	utilisateur	3022	ville	244	2491
R_{rates}	utilisateur	3022	film	6971	195493
R_{Fr}	utilisateur	3022	utilisateur	3022	779
R_{likes}	utilisateur	3022	film	6971	182069

Pour les données expérimentales, on se sert de deux jeux de données bien connus en recommandation, MovieLens100k (ML100K) [HK16] et Douban Movie (DM) [ZL09], tous les deux centrés sur des films. Ils sont composés de notes (1 à 5) attribuées aux films par les utilisateurs, et la relation rating représente le fait qu'un utilisateur ait noté un film. Nous considérons qu'un utilisateur aime un film s'il lui attribue une note de 3 ou plus, et nous désignons la relation correspondante par le terme *likes*. La Figure 2.5 et le Tableau 2.2 présentent les caractéristiques de ces données (tailles, types et relations), ainsi que les schémas HIN que nous utilisons. La relation *likes* et la relation de recommandation potentielle sont en pointillés sur ces figures. Les différences entre schémas viennent des caractéristiques disponibles dans les données.

La diversité calculée sur le méta-chemin $\Pi_1 = R_{\text{rec}} R_{\text{Ty}}$ représente la diversité des genres (types) de films dans une liste de recommandation et capture la même intuition que l'*intra-list similarity* [Zie+05]. D'autres méta-chemins sur ces schémas permettent d'observer d'autres formes de



U : Utilisateur
Ty : Genre (type) de film
Ye : Année (year) de sortie
Oc : Profession (Occupation)
Ge : Genre
Lo : Ville (Location)
Ag : Age
Ac : Acteur
Di : Réalisateur (director)
Gr : Groupe

I : Item
R_{Ty} : Être de ce genre
R_{Ye} : Être sorti telle année
R_{Oc} : Avoir pour profession
R_{Ge} : Être de genre
R_{Lo} : Habiter tel endroit
R_{Ag} : Avoir tel âge
R_{Ac} : Joue dans
R_{Di} : est réalisé par
R_{Gr} : est membre de
R_{rates} : Met la note
R_{likes} : Aime
R_{rec} : Est recommandé

Figure 2.5 : *Schémas pour les HIN utilisés sur MovieLens100K et Douban Movie.*

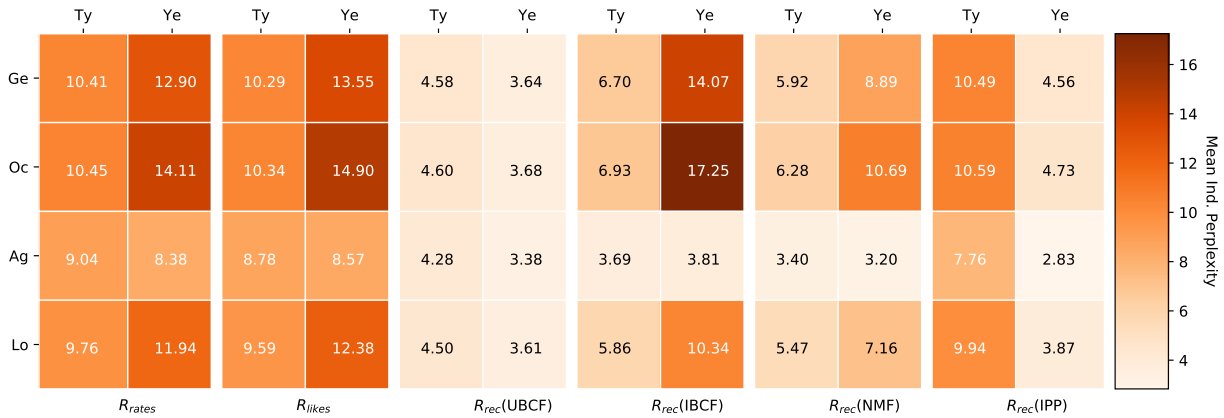


Figure 2.6 : *Diversités individuelles moyennes pour des méta-chemins $\mathcal{P}_{MI}(R_S^{-1} R_X R_T)$ pour ML100K.*

diversité, arbitrairement complexes : par exemple, $\Pi_2 = R_{\text{rec}} R_{\text{likes}}^{-1} R_{\text{likes}} R_{\text{Ty}}$ exprime la diversité des genres de films aimés par les utilisateurs qui ont aussi aimé les films qui leur avaient été recommandés. Le méta-chemin $\Pi_3 = R_{\text{likes}} R_{\text{likes}}^{-1} R_{\text{rec}}^{-1} R_{\text{Ag}}^{-1}$ exprime lui la diversité des âges des utilisateurs qui ont aimé des films qui leur avaient été recommandés.

Dans nos expériences, nous utilisons deux systèmes de recommandation standards, proposant des listes de 5 éléments : un filtrage collaboratif *user-based*, *User-Based Collaborative Filtering*, et une *baseline* de popularité (*Implicit Pure Popularity*, ou IPP, qui recommande les 5 films les plus aimés pas encore vus par l'utilisateur). Les Figures 2.6 et 2.7 présentent des résultats expérimentaux de diversité sur les jeux de données ML100K et Douban. On considère des méta-chemins de longueur 3, partant de S et arrivant à T (respectivement, les utilisateurs et les éléments) : $\Pi = R_S^{-1} R_X R_{\text{T}}$, où X désigne l'une des relations *rating*, *liked* or *recommended*. Par exemple, $\mathcal{P}_{\text{MI}}(R_{Lo}^{-1} R_{rec}(UBCF) R_{\text{Ty}})$ désigne la diversité individuelle moyenne des genres de films recommandés en utilisant UBCF, moyennée sur les lieux de résidence (information sur les utilisateurs). On calcule et présente la diversité individuelle moyenne sur ces chemins, notée $\mathcal{P}_{\text{MI}}(\Pi)$.

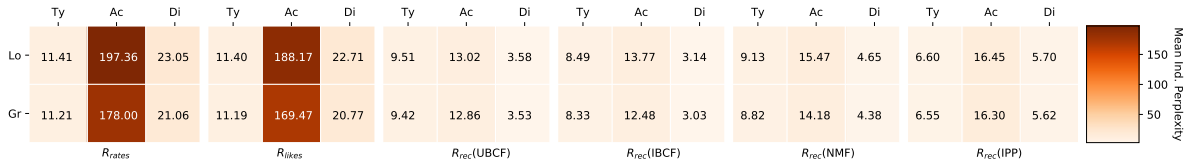


Figure 2.7 : *Diversités individuelles moyennes pour des méta-chemins $\mathcal{P}_{MI}(R_S^{-1} R_X R_T)$ pour DM.*

On remarque d'abord que les $\mathcal{P}_{\text{MI}}(\Pi)$ utilisant *recommended* sont plus basses que celles avec *likes*, ce qui vient du fait que les listes recommandées sont de longueur 5, alors que les utilisateurs ont souvent aimés plus d'éléments. De même, $\mathcal{P}_{\text{MI}}(R_S^{-1} R_X R_{\text{Ac}})$ est bien plus grand que $\mathcal{P}_{\text{MI}}(R_S^{-1} R_X R_{\text{Ty}})$ ou $\mathcal{P}_{\text{MI}}(R_S^{-1} R_X R_{\text{Di}})$, reflétant le fait que les films ont plus d'acteurs que de réalisateurs ou de genres. En d'autres termes, la *variété* des éléments a un effet direct sur la diversité mesurée. Normaliser ces résultats ne serait probablement pas une bonne idée, on souhaite préserver cette contribution de la variété : si un méta-chemin utilisant les nœuds d'acteurs permet d'accéder à un plus grand nombre d'éléments que via les réalisateurs, on souhaite que la diversité mesurée soit plus grande. On constate d'autres phénomènes, comme le fait que la diversité des années de sortie des films par groupe d'âge sont basses ($R_{\text{Ag}}^{-1} R_{\text{likes}} R_{\text{Ye}}$), ce qui tendrait à indiquer que les utilisateurs d'un groupe d'âge donné choisissent préférentiellement des films d'une période limitée dans le temps.

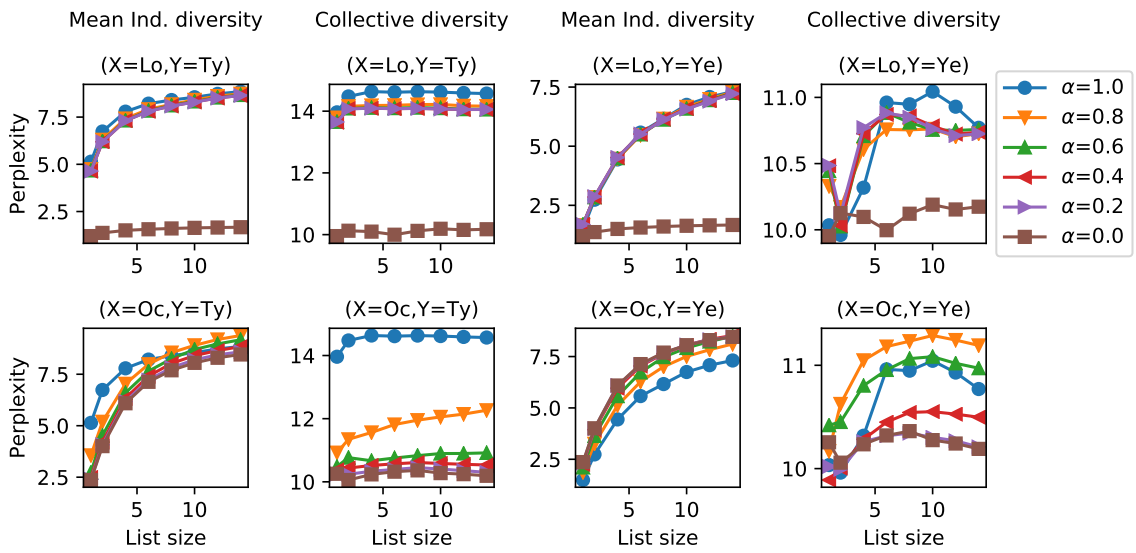


Figure 2.8 : *Diversité individuelle moyenne et collective pour MovieLens 100K, avec des listes de recommandation de tailles variables et des méta-chemins pondérés par α.*

Avec le même protocole, on mesure les diversités individuelles moyennes et les diversités collectives des listes de recommandation, en terme de types d'éléments recommandés. On calcule $\mathcal{P}_{\text{MI}}(\Pi)$ et $\mathcal{P}_{\text{Col}}(\Pi)$ sur le méta-chemin $\Pi = R_{\text{rec}} R_{\text{Ty}}$. La Figure 2.8 met en évidence les différences entre les diversités individuelle moyenne et collective (les résultats sont similaires sur les données Douban). La diversité individuelle moyenne est augmente systématiquement avec la taille de la liste, ce qui est attendu : quand l'utilisateur voit davantage d'éléments, il a accès à davantage de types (la variété croit, pas nécessairement l'équilibre). Cependant, cette tendance est bornée par le nombre de types.

En revanche, la diversité collective peut stagner voire décroître. Ainsi, bien que chaque utilisateur reçoive des recommandation plus diversifiées, l'ensemble des utilisateurs se voit recommander des éléments moins divers. D'autres phénomènes sont visibles sur ces expériences. Inclure des données utilisateurs peut améliorer la justesse, mais cela réduit la diversité collective. C'est particulièrement visible sur les données ML100K, où les combinaisons de paramètres avec le meilleur score F_1 sont celles avec les diversités les plus faibles (individuelle et collective). En revanche, ces données utilisateurs ont un effet limité sur la diversité individuelle moyenne, ce qui est un peu surprenant : les données utilisateurs ont tendance à lier des utilisateurs avec peu en commun (ou seulement leur genre), on s'attendrait à ce que cette diversité augmente.

2.4 Conclusion

L'étude des mesures de diversité de ce chapitre a une portée générale, qui dépasse le cadre des seuls systèmes de recommandation. Elle s'appuie sur un ensemble de travaux dans diverses communautés de recherche. Notre *framework* utilisant les HIN offre des grandes possibilités pour analyser des systèmes de recommandation et la diversité de leurs suggestions. Nos expériences, pour intéressantes qu'elles soient, sont restées de portée limitée dans la communauté : l'analyse de la topologie du réseau sous-jacent à un système de recommandation reste encore embryonnaire et on lui privilégie d'autres analyses, aux conclusions plus immédiatement pratiques.

En terme de perspectives, la diversité des recommandations est, aujourd'hui plus que jamais, un sujet important pour la société. Des équipes de recherche universitaires proposent régulièrement des études autour de la diversité, de la *fairness* (voir récemment [Tre+22 ; Bur+22 ; Mic+22]). Il semble que l'on manque toujours cruellement de protocoles d'évaluation ouverts et des données s'étendant sur une longue durée, pour évaluer les effets de ces systèmes à moyen et long terme. Et si le travail se poursuit dans le monde académique, on peut déplorer les efforts limités des grands acteurs industriels – qui, eux, ont accès à ces données – sur ces questions.

Chapitre 3

Apprendre sur des graphes, pour recommander

Perfecting oneself is as much unlearning
as it is learning.

Edsger Dijkstra

Sommaire

3.1 Entraînement des modèles *GNN* en recommandation **86**
3.1.1 Problématique 86
3.1.2 Optimisation directe avec le rang 87
3.1.3 Échantillonnage d'exemples négatifs 88
3.1.4 Protocoles expérimentaux inductif et transductif 89
3.2 Expériences **89**
3.2.1 Protocoles 90
3.2.2 Résultats 91
3.2.3 Analyses quantitative et qualitative, du modèle 92
3.3 Conclusion **93**

Dans ce chapitre, nous présentons des travaux autour des modèles de Graph Neural Networks, une extension des techniques d'apprentissage profond aux modèles de graphes. C'est une thématique de recherche un peu plus récente pour moi et des travaux sur des architectures GNN pour les graphes *dynamiques* sont en cours (avec Yannis Karmim et Nicolas Thome), ils ne seront pas détaillés ici. De même, une étude sur l'usage d'un modèle GNN pour améliorer une tâche de recherche d'information (légender des images en analysant le graphe DOM d'une page avec un GNN), est absente de ce manuscrit [Dan+21].

Les *Graph Neural Networks* (GNN) sont devenus en quelques années des architectures incontournables pour les tâches de recommandation top-k, où l'on souhaite ordonner l'ensemble des éléments

à proposer à un utilisateur, en vue de choisir les k premiers de la liste. Ces modèles surpassent parfois les modèles à factorisation de matrices grâce à leur capacité à agréger de l'information d'un contexte élargi. S'il existe pléthore d'architectures différentes pour proposer de bonnes recommandations selon les données et le contexte souhaité, les travaux que nous présentons dans ce chapitre se concentrent sur l'*entraînement* de ces architectures, sur trois aspects.

Tout d'abord, la fonction de perte utilisée pour guider l'entraînement des modèles GNN classiques repose souvent sur les principes du *Bayesian Personalized Ranking* (BPR) [Ren+09], alors que l'évaluation se fait avec le NDCG@k ou le rappel@k. Ces deux mesures sont basées sur le rang des éléments, et nous nous demandons si l'on pourrait améliorer la performance d'un modèle en incorporant, dès son entraînement, la fonction objectif qui servira d'évaluation (ou, au moins, une approximation de celle-ci).

Ensuite, nous étudions l'échantillonnage d'éléments négatifs. En effet, pour entraîner un modèle à apprendre, celui-ci doit chercher à distinguer, pour un utilisateur donné, des exemples *positifs* et des exemples *négatifs*, que l'utilisateur aura, respectivement, « vu », et « pas vu ». Pourtant, on doit échantillonner un ensemble fini d'éléments non vus j pour un élément positif i, car il n'est pas envisageable de considérer *tous* les triplets (u, i, j). Dans un catalogue de grande taille, avec des utilisateurs qui n'ont en général vu qu'une fraction réduite de tous les éléments, il est très facile de tirer aléatoirement des exemples négatifs qui n'ont rien en commun avec les positifs (c'est la majorité du catalogue). Cependant, ces exemples aléatoires ne ressemblent en général pas du tout aux exemples positifs vus par l'utilisateur : la machine n'a aucun mal à les distinguer, et cela ne donne pas un bon apprentissage. Diverses techniques ont été proposées pour trouver ou créer des négatifs « difficiles », nous en proposons une nouvelle à partir d'une marche aléatoire (*Personalized PageRank*).

Enfin, on se pose également la question du protocole expérimental. Le jeu de données utilisé en apprentissage est en général découpé de façon dite « transductive », ce qui permet un bon apprentissage mais est, peut-être, un peu plus éloigné de la réalité de la recommandation qu'un entraînement en « induction ». Dans notre approche graphes, le protocole *inductif*, sépare les données selon les utilisateurs, ce qui a pour effet d'introduire de nouveaux utilisateurs dans la phase de test et de mettre en évidence le pouvoir de généralisation des systèmes de recommandation évalués (voir la figure 3.2 et la section 3.1.4).

Après une présentation de ces idées, nous développons une partie expérimentale et de discussion des résultats.

3.1 Entraînement des modèles *GNN* en recommandation

3.1.1 Problématique

Dans un contexte de recommandation, on considère un graphe biparti non dirigé, $\mathcal{G} = (\mathcal{U}, \mathcal{V}, \mathcal{E})$ avec $|\mathcal{U}|$ utilisateurs, $|\mathcal{V}|$ éléments et $|\mathcal{E}|$ arêtes. La phase d'apprentissage, ou d'entraînement du modèle, associe chaque nœud de $\mathcal{U} \times \mathcal{V}$ à une représentation h, un *embedding*.

[0]Dans un processus d'apprentissage, une fonction de perte, en anglais *loss function*, calcule l'erreur commise par la machine à une étape donnée, et l'on tâche de réduire cette erreur dans l'itération suivante.

La plupart des modèles de recommandation cherchent à prédire un score pour chaque élément : ils sont dits *point-wise*. De nombreux modèles comme la BPR reposent sur une comparaison deux-à-deux d'éléments (*pairwise*, par paires) de façon à construire un espace d'*embeddings* dans lequel les représentations des utilisateurs sont plus proches des items positifs, $\mathcal{V}^+$, que des items négatifs, $\mathcal{V}^-$. Si l'on considère un utilisateur u d'embedding h_u, un item positif p et un item négatif j, on cherche à avoir $s_p > s_j$, avec $s_p = h_u \cdot h_p^T$, et $s_j = h_u \cdot h_j^T$, où s_p et s_j sont les scores de similarité entre les embeddings. Dans notre cas, on va explorer une troisième voie, une approche *list-wise*, en intégrant dans la fonction de perte une approximation du rang des items considérés.

3.1.2 Optimisation directe avec le rang

Reprenant des idées issues de la recherche d'information [Bro+20], nous avons proposé d'entraîner les GNN *directement* avec une fonction de perte qui prend en compte des approximations du classement adaptées à une descente de gradient. Pour cela, on commence par réécrire la position d'un élément dans un classement (son rang) en fonction des scores ayant permis le classement :

$$\text{rank}(p) = 1 + \sum_{j \in \mathcal{V}, j \neq p} H(s_j - s_p) \tag{3.1}$$

où $\text{rank}(p)$ est la position de l'élément p, s_p et s_j sont les scores de p et j, et H est la fonction de Heaviside, indicatrice de $\mathbb{R}^+$, classiquement définie comme suit :

$$H = \begin{cases} 1 \text{ si } s_j - s_p > 0 \\ \text{non définie pour } s_j - s_p > 0 \\ 0 \text{ si } s_j - s_p > 0 \end{cases} \tag{3.2}$$

Pour connaître le rang d'un élément positif p, on compte donc le nombre d'items j qui ont une similarité avec l'utilisateur courant plus grande que p.

Durant l'entraînement, on cherche à minimiser le nombre d'éléments *négatifs* qui ont un score plus grand que les *positifs*. Plusieurs mesures fondées sur le rang existent, chacune pondérant de façon spécifique les erreurs commises dans le classement. Avec la formulation du rang de l'équation 3.1, on comprend pourquoi ce n'est pas différentiable : la fonction de Heaviside ne l'est pas, ses gradients sont nuls ou non définis. On peut utiliser par exemple la sigmoïde, pour approcher les valeurs de la fonction de Heaviside : $\sigma(x;\ \tau) = \frac{1}{1+\exp\frac{-x}{\tau}}$, où $\tau \in \mathbb{R}$ est un paramètre de « température », qui permet de contrôler la pente de la sigmoïde (plus τ est petit, plus cette pente est importante et la sigmoïde sature plus vite). Le rang peut alors être approché de la façon suivante :

$$\text{rank}_s(p, \tau) = 1 + \sum_{j \in \mathcal{V}} \sigma(s_j - s_p;\ \tau). \tag{3.3}$$

Il devient alors maintenant différentiable et utilisable dans une descente de gradient. On note $\mathcal{L}_{\text{ITEM}}$ notre fonction de perte :

$$\mathcal{L}_{\text{ITEM}} = 1 - \frac{\text{DCG}_s}{\text{iDCG}}, \text{ où } \text{DCG}_s = \sum_{p \in \mathcal{V}^+} \frac{1}{\log_2(1 + \text{rank}_s(p))} \tag{3.4}$$

Le principe complet de cette recommandation est illustré sur la Figure 3.1. Un utilisateur cible u_q (en violet) est utilisé comme *requête* (on est ici proche d'un fonctionnement de recherche d'information, comme celui de Learning to rank [Cao+07]). Les éléments positifs, en vert, doivent avoir des embeddings plus proches que les éléments négatifs, en rouge. Après avoir calculé la perte, avec l'équation 3.4, on rétropropage les gradients dans le réseau pour mettre à jour les poids dans le GNN et les embeddings des utilisateurs et éléments.

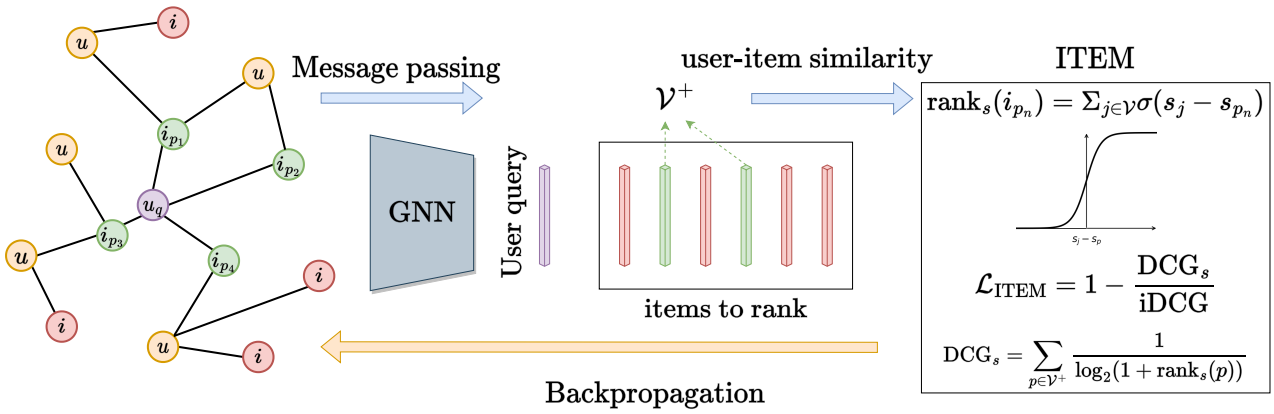


Figure 3.1 : *Principe de la recommandation avec une approximation du rang.*

3.1.3 Échantillonnage d'exemples négatifs

Pour chaque utilisateur pour lequel on souhaite produire des recommandations, on dispose en général d'un nombre relativement limité d'éléments positifs, et d'un grand nombre de négatifs. L'entraînement des GNN se fait par lots (*batchs*), c'est-à-dire en travaillant sur un sous-ensemble des éléments, parce qu'il est techniquement trop coûteux d'utiliser tous les éléments. On doit donc échantillonner, tirer au hasard des éléments positifs et négatifs, pour constituer les lots au sein desquels on calculera une approximation *satisfaisante* du classement, c'est-à-dire qui reflètera le classement réel [Ram+21b]. La plupart des modèles de recommandation ont commencé avec un échantillonnage simple, c'est-à-dire uniforme. Mais cela limite les performances car en faisant cela on tombe statistiquement sur des négatifs peu intéressants, loin de notre utilisateur (en terme de représentations). Ils sont donc facile à discriminer.

Nous proposons d'utiliser un PageRank Personnalisé (PPR) [Pag+98b] pour calculer des proximités entre nœuds, et pondérer l'échantillonnage. On s'attend à ce que les éléments avec un PPR élevé soient plus difficiles à classer, car plus proches de notre utilisateur. L'échantillonnage de négatifs est un sujet actif de longue date, et des approches récentes ont proposé de créer des éléments négatifs difficiles *synthétiques* [Hua+21]. L'avantage du PPR est qu'on peut le calculer hors-ligne, c'est-à-dire avant l'apprentissage proprement dit : ainsi, le processus est rapide et permet de créer

des lots adaptés à la fonction de perte $\mathcal{L}_{\text{ITEM}}$. Le score PPR utilisé est le suivant, après normalisation par une fonction softmax :

$$p(i_j^-|u_q) \quad \sim \quad \frac{e^{\text{ppr}_{u_q}(i_j^-)}}{\sum_{k\in\mathcal{V}^-} e^{\text{ppr}_{u_q}(i_k^-)}}. \tag{3.5}$$

3.1.4 Protocoles expérimentaux inductif et transductif

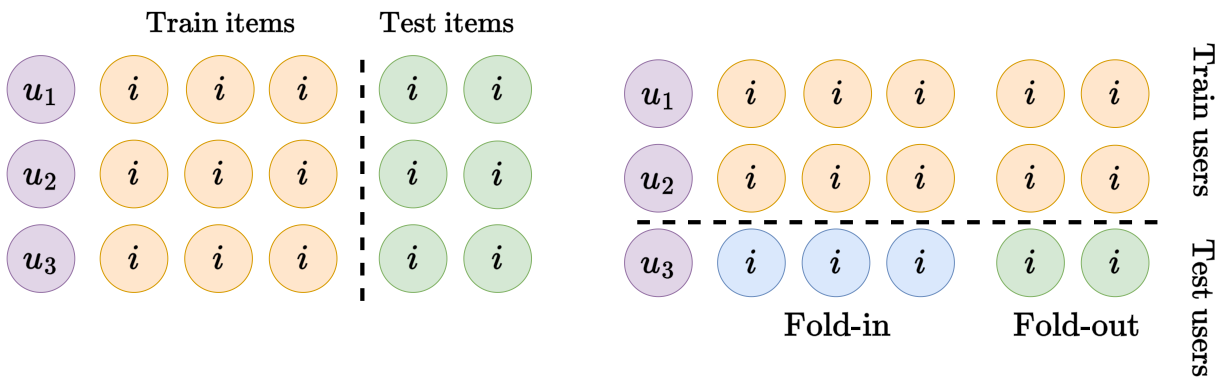


Figure 3.2 : *Protocole experimental en transductif et inductif.*

Le protocole standard d'évaluation des GNN sur les tâches de recommandation est *transductif*, c'est-à-dire que les éléments et les utilisateurs sont les mêmes lors de l'entraînement et de l'évaluation (illustré sur la partie gauche de la Figure 3.2. Dans ce cas, le découpage se fait sur les interactions : pour un utilisateur donné, on dispose d'une fraction ρ (les nœuds orange) de ses interactions pour faire le passage de messages, et l'on évalue le GNN sur les nœuds verts (fraction $1-\rho$). Comme les nœuds sont les mêmes à l'entraînement et lors de la validation, le système ne peut pas proposer de recommandation à des nouveaux utilisateurs, il se limite à proposer de nouveaux éléments aux utilisateurs déjà présents.

Pour être plus proche de la réalité des applications des GNNs à des systèmes de recommandation et de leurs problèmes récurrents de démarrage à froid, nous avons proposé un protocole expérimental inductif proche de celui de [Men+20] et [KW16], permettant d'inférer une représentation pour de nouveaux utilisateurs à partir de quelques éléments. Plus précisément, durant l'entraînement, le GNN a accès à une fraction μ (u_1, u_2 sur la Figure 3.2) des utilisateurs et de leur historique d'interaction. Pour l'évaluation, on considère la fraction $1-\mu$ (représentée par u_3 dans Figure 3.2) des utilisateurs. En ayant accès à une fraction η de leurs interactions, les éléments du *fold-in* (en bleu sur la Figure 3.2), on doit recommander des éléments parmi les $1-\eta$ restants (le *fold-out*, en vert).

3.2 Expériences

Pour compléter cette partie théorique, nous présentons quelques résultats expérimentaux pour valider l'idée que l'entraînement des GNN serait meilleur avec des fonctions de perte dédiées au classement (pour de la recommandation top-k), et que le protocole inductif serait peut-être plus réaliste que le transductif.

3.2.1 Protocoles

Nous utilisons notamment les données ML100k de MovieLens, ainsi qu'une version plus grande de ces données, ML1M (1 million d'interactions, au lieu de 100 000), ainsi que des données provenant de Yelp [Asg16] et Amazon [Wan+19]. Les détails des données sont dans le Tableau 3.1.

Données	#Utilisateurs	#Items	#Interactions (arêtes)
MovieLens-100K [HK16]	610	8957	100k
MovieLens-1M [HK16]	6022	3043	1m
Yelp-2018 [Asg16]	31668	38048	1.5m
Amazon-book [Wan+19]	52643	91599	2.9m

Tableau 3.1 : *Jeux de données.*

Les mesures que nous utilisons sont le NDCG@20 et le Rappel@20. Le NDCG se calcule comme suit :

$$\text{NDCG} = \frac{\text{DCG}}{\text{iDCG}}, \text{ où } \begin{cases} \text{DCG} = \sum\limits_{p \in \mathcal{V}^+} \frac{1}{\log_2(1+\text{rank}(p))} \\ \text{iDCG} = \max\limits_{\text{classement}} \text{DCG} \end{cases} \tag{3.6}$$

Le NDCG@k est une restriction du NDCG ne prenant en compte que k éléments dans le classement.

Le protocole transductif que nous utilisons ici est le même que celui de LightGCN [He+20] et de MixGCF [Hua+21], deux articles de référence pour la tâche top-k avec des architectures GNN. L'ensemble d'apprentissage comporte 80% des interactions, le reste sert pour le test et la validation. En inductif, on prend 80% des utilisateurs pour l'entraînement, 10% pour le test et 10% pour la validation. Les utilisateurs d'entraînement gardent leur historique d'interaction. En test et validation, on découpe en ensembles dits *fold-in* et *fold-out*, et on garde 80% pour le fold-in, 20% pour le fold-out.

De nombreuses *baselines* classiques en GNN au moment de ces travaux ont été utilisées pour nos expériences : GCN [KW17], GAT [Vel+18], GIN [Xu+19] et LightGCN [He+20]. On compare leur entraînement classique avec la fonction de perte BPR et avec $\mathcal{L}_{\text{ITEM}}$. On se compare également à d'autres approches du filtrage collaboratif, comme MF-BPR [Ren+09], ENMF [Che+20a], CML [Hsi+17], des approches d'embedding pures (DeepWalk [PAS14], LINE [Tan+15], Node2Vec [GL16]), des modèles VAE (auto-encodeurs variationnels) comme Mult-VAE [Lia+18], et des MP-GNNs NGCF [Wan+19], DGCF [Wan+20], LightGCN [He+20], NIA-GCN [Sun+20], LR-GCCF [Che+20b], des modèles hybrides SimpleX [Mao+21]. Pour l'échantillonnage de négatifs, on confronte notre méthode à NeuralNDCG [PB21], SGL-ED [Wu+21b] qui combine la BPR avec de l'auto-supervision et MixGCF [Hua+21] et ses négatifs synthétiques. Tous sont comparés sur l'architecture LightGCN.

3.2.2 Résultats

En comparant notre approche à des méthodes de l'état de l'art, avec une architecture LightGCN et un protocole transductif, on observe qu'on dépasse toutes les architectures de GNN testées, avec des améliorations parfois très significatives (*cf.* Tableau 3.2), que ce soit sur des approches génériques ou des approches comme NeuralNDCG [PB21] ou MixGCF, à l'état de l'art sur cette tâche (au moment de l'étude).

Méthode	MovieLens-1M		Yelp-2018		Amazon-Book	
	R@20	N@20	R@20	N@20	R@20	N@20
MF-BPR [Ren+09]	21.53	21.75	5.49	4.45	3.38	2.61
CML [Hsi+17]	17.30	15.63	6.22	5.36	5.22	4.28
ENMF [Che+20a]	23.15	20.69	6.24	5.15	3.59	2.81
DeepWalk [PAS14]	13.48	10.57	4.76	3.78	3.46	2.64
LINE [Tan+15]	23.36	22.26	5.49	4.46	4.10	3.18
Node2Vec [GL16]	14.75	11.86	4.52	3.60	4.02	3.09
Mult-VAE [Lia+18]	29.23	23.84	6.41	4.97	4.46	3.33
SimpleX [Mao+21]	28.02	26.70	7.01	5.75	**5.83**	**4.68**
NGCF [Wan+19]	25.13	25.11	5.79	4.77	3.44	2.63
LR-GCCF [Che+20b]	22.31	21.24	5.61	3.43	3.35	2.65
NIA-GCN [Sun+20]	23.59	22.42	5.99	4.91	3.69	2.87
LightGCN [He+20]	25.76	24.27	6.28	5.15	4.23	3.17
DGCF [Wan+20]	26.40	25.04	6.54	5.34	4.22	3.24
SGL-ED [Wu+21b]	26.34	24.87	6.75	5.55	4.78	3.79
NeuralNDCG [PB21]	29.45	25.13	6.50	5.23	4.38	3.27
MixGCF [Hua+21]	27.35	24.56	7.17	5.84	4.51	3.41
ITEM	**29.80**	**27.20**	**7.39**	**5.88**	5.23	4.02

Tableau 3.2 : *Comparaison d'ITEM avec diverses méthodes de l'état de l'art, avec le protocole transductif. Les meilleurs résultats sont en* ***gras****.*

Les GNN se prêtent bien aux protocoles inductifs, contrairement aux méthodes à factorisation de matrices : leur fonctionnement repose sur de la propagation d'information dans un réseau et cela permet de généraliser à des utilisateurs non vus pendant l'entraînement. Nos expériences montrent que $\mathcal{L}_{\text{ITEM}}$ accroît leurs performances, dépassant les méthodes à l'état de l'art en contexte inductif comme Mult-VAE [Lia+18], ou MixGCF (*cf.* Tableau 3.3).

On a montré également que notre stratégie d'échantillonnage de négatifs, quoique assez élémentaire, peut améliorer les performances obtenues avec notre fonction de perte $\mathcal{L}_{\text{ITEM}}$, comme l'indiquent les résultats du Tableau 3.4. On utilise pour ces tests simples une architecture standard LightGCN, en protocole transductif. MixGCF reste coûteux, en tentant de calculer les négatifs lors de l'entraînement, ce qui fait baisser ces performances car on ne peut en échantillonner suffisamment.

Méthode	MovieLens-100K		Yelp-2018		Amazon-Book	
	R@20	N@20	R@20	N@20	R@20	N@20
Mult-VAE [Lia+18]	30.14	28.28	10.15	8.18	10.86	9.2
GCN [KW17]	28.74	27.68	7.34	5.76	8.85	7.61
GAT [Vel+18]	31.01	28.92	9.04	7.32	9.88	8.17
GIN [Xu+19]	29.71	27.58	7.34	5.76	9.62	8.05
LightGCN [He+20]	30.79	29.73	7.88	6.34	9.56	8.02
NeuralNDCG [PB21]	31.12	30.07	9.14	7.62	9.54	8.19
MixGCF [Hua+21]	32.07	30.62	9.85	8.21	10.11	9.63
ITEM	**33.84**	**32.63**	**10.54**	**8.70**	**11.03**	**9.89**

Tableau 3.3 : *Comparaison de notre méthode avec diverses approches de l'état de l'art, en protocole inductif. Les meilleurs résultats sont en **gras**.*

$\mathcal{L}_{\text{ITEM}}$	MixGCF	PPR	Yelp-2018		Amazon-book	
			R@20	NDCG@20	R@20	NDCG@20
×	×	×	6.26	5.14	4.23	3.17
+	×	×	7.25	5.70	5.09	3.87
+	+	×	7.21	5.71	4.86	3.53
+	×	+	**7.39**	**5.88**	**5.23**	**4.02**

Tableau 3.4 : *Étude des composants de notre méthode. On utilise + pour $\mathcal{L}_{\text{ITEM}}$, ou × pour la BPR. On compare aussi l'échantillonnage des négatifs, soit aléatoire, soit avec MixGCF, soit avec le PPR de l'équation 3.5. Toutes les expériences utilisent LightGCN [He+20] en protocole **transductif**.*

3.2.3 Analyses quantitative et qualitative, du modèle

Une étude de vitesse comparant l'entraînement de LightGCN avec la BPR, MixGCF et $\mathcal{L}_{\text{ITEM}}$ sur les données Yelp-2018 (1.5 millions d'interactions) montre que notre méthode converge significativement plus vite qu'une méthode reposant sur une BPR, avec ou sans MixGCF. On a en particulier besoin de nettement moins d'itérations pour constituer les lots d'éléments positifs et négatifs qui servent au calcul approché du NDCG.

Qualitativement, on compare rapidement sur la Figure 3.3 les classements proposés par notre méthode et ceux obtenus avec la BPR. Les éléments positifs sont liserés de vert, les autres de rouge : on constate que le classement proposé avec par notre méthode comporte moins de négatifs (8 contre 11) et que les éléments positifs sont présents dans le haut de la liste. C'est évidemment un test rapide, qui mériterait une systématisation pour être robuste. Les performances de notre entraînement avec $\mathcal{L}_{\text{ITEM}}$ semblent donc intéressantes, pour ce contexte de recommandation top-k avec des GNN, tant en terme de vitesse et de légèreté dans la mise en œuvre, que dans la qualité des recommandation proposées.

(a) *ITEM* **(b)** *BPR*

Figure 3.3 : *Classement des 20 premiers éléments recommandés à un utilisateur sur MovieLens-100k (en inductif) avec l'architecture LightGCN, et les fonctions de perte BPR ou $\mathcal{L}_{ITEM}$. Les éléments positifs sont liserés de vert, les autres de rouge.*

3.3 Conclusion

Les travaux présentés dans ce chapitre constituent une étude préliminaire autour des techniques qui pourraient améliorer des architectures GNN pour effectuer de la recommandation. On a observé en détail plusieurs aspects : introduire une fonction de perte différente pour mieux guider l'entraînement, échantillonner différemment de éléments négatifs pour que la machine ait un apprentissage plus difficile, et travailler avec un protocole expérimental différent, plus réaliste. Les performances obtenues sont à l'état de l'art du domaine, mais il faudrait bien entendu des tests reproduits dans des contextes élargis pour en mesurer vraiment l'impact.

En terme de perspectives, on peut constater que la communauté est toujours très active pour étudier les architectures GNN et leur apport aux systèmes de recommandation. L'absence de réelle standardisation des protocoles d'évaluation suscite des débats sur la significativité des résultats [Ane+23], la communauté a besoin d'améliorer ce point et ce devrait être un enjeu des prochaines années.

Un enjeu important tient aussi au rapprochement de la communauté « réseaux complexes » avec la communauté « apprentissage » : il y a une finesse d'étude des graphes (quels qu'ils soient) dans la première communauté qui pourrait bénéficier aux techniques d'apprentissage. Les modèles d'apprentissage GNN sont encore souvent des variantes de modèles existants, en incorporant de façon trop limitée les indicateurs développés pour analyser les graphes. Bien sûr, quelques chercheurs éminents font le pont entre ces communautés [Wan+21], mais il y a encore beaucoup à faire, pour avoir des équipes couplant les spécialistes des deux disciplines et en tirer pleinement tous les bénéfices.

Chapitre 4

Conclusion et perspectives

People think that computer science is the art of geniuses but the actual reality is the opposite, just many people doing things that build on each other, like a wall of mini stones.

Donald E. Knuth

4.1 Contributions

Dans cette seconde partie, j'ai présenté des travaux sur les systèmes de recommandation avec des modèles de graphes, en détaillant deux contributions : une étude de la diversité et une analyse de l'entraînement d'architectures modernes d'apprentissage profond.

Dans le chapitre 2, nous avons analysé la notion de diversité. Nous avons proposé un *framework* qui rassemble des indicateurs venus de différentes disciplines. Il repose sur les Heterogeneous Information Networks, un modèle qui étend la notion classique de graphes. Avec des marches aléatoires contraintes à emprunter certains méta-chemins, nous avons défini des mesures de diversité individuelle et collective, avant de montrer par quelques expériences, que l'on pouvait reproduire des analyses classiques de diversité mais aussi d'obtenir une finesse souvent inaccessible aux autres approches.

Dans le chapitre 3, nous avons travaillé sur une architecture à la pointe de l'état de l'art en systèmes de recommandation, les *Graphs Neural Networks* (GNN). Là encore, le travail s'articule autour d'une modélisation avec des graphes d'un problème pratique. Nous avons examiné les modalités d'entraînement des GNN pour la recommandation et proposé d'incorporer une approximation du rang dans la mesure d'entraînement. Nous avons ensuite étudié l'échantillonnage des éléments négatifs. Notre solution repose sur le calcul d'un indicateur classique en graphes, le Personalized PageRank, qui donne des négatifs plus difficiles à discriminer, avec une approche particulièrement rapide et efficace dans la constitution des lots d'apprentissage. Enfin, nous avons proposé d'adopter un protocole d'apprentissage dit *inductif*, de façon à être plus réaliste quant aux performances des GNN. Nos résultats expérimentaux montrent que ce protocole se couple bien avec notre fonction de perte et donne de bons résultats.

4.2 Perspectives

Mes travaux sur les GNN apportent une brique de plus à l'édifice des connaissances sur ces architectures, mais il y en a beaucoup qui ont été proposées ces dernières années. Ainsi, il est encore tôt pour dire quelles seront les optimisations de l'entraînement des GNN, ou plus généralement les architectures d'apprentissage profond sur les graphes, qui seront les standards dans quelques années. La proximité des GNN avec les modèles Transformers, aujourd'hui très « à la mode » (car très puissants), devraient offrir encore de nombreuses années de recherches. Nombre de questions sont encore très ouvertes, en particulier la prise en compte de la dynamique des systèmes. Si la communauté des réseaux complexes étudie et propose des modèles de graphes dynamiques depuis de nombreuses années, l'apprentissage sur graphes dynamiques en est encore à ses débuts. Que ce soit sur les modèles, développés seulement à partir de 2019, et qui sont donc encore à perfectionner, ou sur les protocoles d'évaluations, pas encore standardisés, il reste nombre de travaux majeurs à effectuer pour faire mûrir ces architectures – et l'on aura compris que j'espère y participer pleinement.

Les travaux du projet ANR Algodiv, au sein duquel s'inséraient mes recherches sur la diversité, ont permis d'éclairer plusieurs points relatifs à la recommandation de contenus en ligne, comme l'intérêt des approches « graphes », ou que, pour les plateformes les plus en vue, la recommandation a tendance à réduire l'espace exploré par les utilisateurs, ou encore la nécessité d'une meilleure gouvernance de ces pratiques algorithmico-commerciales.

Quelques années après, nous pouvons constater que le Web continue de changer, faisant évoluer les pratiques et les conclusions. Il semble que l'on assiste actuellement à un phénomène de désagrégation et d'individualisation massives [War23]. L'omniprésence de la publicité sur Facebook, les changements profonds subis par Twitter/X, ou des plateformes comme Tiktok qui font d'autres choix algorithmiques : les pratiques tendent à s'individualiser et chacun tend à ne plus savoir ce que les autres voient. Ainsi, les vidéos les plus populaires sur Netflix ou Tiktok (comme la série The Night Agent, ou une routine de maquillage) ne sont plus celles qui suscitent le plus de discussions : l'influence peut être massive, avec d'impressionnants chiffres d'audience à l'appui, mais elle reste en *silo*. La célébrité semble ne plus toujours devenir « commune », comme cela a pu être le cas dans la décennie précédente. Cela pose évidemment d'intéressantes questions sur la diversité : cette individualisation amène-t-elle une diversité collective meilleure ? Est-ce que cela réduit *vraiment* la diversité individuelle ? Et, pour les universitaires, quelles vont être les possibilités de mesurer à l'avenir ces phénomènes, pourront-ils se passer de la collaboration et des données desdites plateformes ? Et si l'on va plus loin, pour la société, on peut en revenir à la conjecture de Chavalarias : « le modèle économique actuel de la Big Tech, fondé sur la marchandisation de l'influence sociale, est-il incompatible avec la pérennité de nos démocraties ? » [Cha23]. Ces évolutions de nos systèmes numériques demandent qu'on trouve collectivement les modalités pour assurer qu'une société rassemble, autant que possible, des personnes aux références communes, sur lesquelles fonder un débat public de qualité.

Annexes

Bibliographie

[W3C04] WWW Consortium (W3C). *MusicXML W3C XML Schema Definition (XSD).* https ://www.musicxml.com/for-developers/musicxml-xsd/. Date de consultation : 15 Novembre 2023. 2004. [Cité page 14]

[Ada+04] Norman H. Adams, Mark A. Bartsch, Jonah B. Shifrin et Gregory H. Wakefield. « Time series alignment for music information retrieval ». In : *International Society for Music Information Retrieval Conference (ISMIR).* 2004, p. 303-311. [Cité page 13]

[Agg16] Charu C Aggarwal. *Recommender systems. The textbook.* T. 1. Springer, 2016. [Cité pages 59, 66]

[Agg+99] Charu C Aggarwal, Joel L Wolf, Kun-Lung Wu et Philip S Yu. « Horting hatches an egg : A new graph-theoretic approach to collaborative filtering ». In : *Proc. 5th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining.* ACM. 1999, p. 201-212. [Cité page 68]

[Ale06] Gerald Alexanderson. « About the cover : Euler and Königsberg's Bridges : A historical view ». In : *Bulletin of the American Mathematical Society* 43.4 (2006), p. 567-573. [Cité page 61]

[All+09] Julien Allali, Pascal Ferraro, Pierre Hanna, Costas Iliopoulos et Matthias Robine. « Toward a general framework for polyphonic comparison ». In : *Fundamenta Informaticae* 97.3 (2009), p. 331-346. [Cité page 15]

[And+20] Ashton Anderson, Lucas Maystre, Ian Anderson, Rishabh Mehrotra et Mounia Lalmas. « Algorithmic effects on the diversity of consumption on Spotify ». In : *Proceedings of The Web Conference 2020.* 2020, p. 2155-2165. [Cité page 62]

[Ane+23] Vito Walter Anelli, Daniele Malitesta, Claudio Pomo, Alejandro Bellogín, Eugenio Di Sciascio et Tommaso Di Noia. « Challenging the myth of graph collaborative filtering : a reasoned and reproducibility-driven analysis ». In : *Proceedings of the 17th ACM Conference on Recommender Systems.* 2023, p. 350-361. [Cité page 93]

[Ank69] Paul Anka. *My Way.* Interpret : Frank Sinatra. 1969. [Cité page 33]

[Asg16] Nabiha Asghar. « Yelp dataset challenge : Review rating prediction ». In : *arXiv preprint arXiv :1605.05362* (2016). [Cité page 90]

[Ass+99] Gérard Assayag, Camilo Rueda, Mikael Laurson, Carlos Agon et Olivier Delerue. « Computer-Assisted Composition at IRCAM : From PatchWork to OpenMusic ». In : *Computer Music Journal* 23.3 (1999), p. 59-72. [Cité page 14]

[AP02] Jean-Julien Aucouturier et Francois Pachet. « Music similarity measures : What's the use ? » In : *Ismir.* T. 7. 2002, p. 339-340. [Cité page 16]

[BMA15] Eytan Bakshy, Solomon Messing et Lada A Adamic. « Exposure to ideologically diverse news and opinion on Facebook ». In : *Science* 348.6239 (2015), p. 1130-1132. [Cité page 62]

[Bal96] Mira Balaban. « The Music Structures Approach to Knowledge Representation for Music Processing ». In : *Computer Music Journal* 20.2 (1996), p. 96-111. [Cité page 13]

[BA99] Albert-László Barabási et Réka Albert. « Emergence of scaling in random networks ». In : *Science* 286.5439 (1999), p. 509-512. [Cité page 61]

[Bar07] Michael J. Barber. « Modularity and community detection in bipartite networks ». In : *Phys. Rev. E - Statistical, Nonlinear, and Soft Matter Physics* 76.6 (2007). [Cité page 69]

[Bec16] Stephen J Beckett. « Improved community detection in weighted bipartite networks. » In : *Royal Society Open Science* 3.1 (2016). [Cité page 69]

[BKV07] Robert M Bell, Yehuda Koren et Chris Volinsky. « The Bellkor solution to the Netflix prize ». In : *KorBell team's report to Netflix* (2007). [Cité page 60]

[Bel11] Juan Pablo Bello. « Measuring Structural Similarity in Music ». In : *IEEE Transactions on Audio, Speech, and Language Processing* 19 (2011), p. 2013-2025. [Cité page 16]

[Ber+04] Adam Berenzweig, Beth Logan, Daniel PW Ellis et Brian Whitman. « A large-scale evaluation of acoustic and subjective music-similarity measures ». In : *Computer Music Journal* (2004), p. 63-76. [Cité page 16]

[Ber58] Claude Berge. *La théorie des graphes*. Paris : Dunod, 1958. [Cité page 61]

[BP70] Wolfgang H Berger et Frances L Parker. « Diversity of planktonic foraminifera in deep-sea sediments ». In : *Science* 168.3937 (1970), p. 1345-1347. [Cité page 75]

[Ber78] Hector Berlioz. *Mémoires*. Paris : Calmann-Lévy, 1878. [Cité page 53]

[Ber+11] José F Bernabeu, Jorge Calera-Rubio, José M Iñesta et David Rizo. « Melodic identification using probabilistic tree automata ». In : *Journal of New Music Research* 40.2 (2011), p. 93-103. [Cité page 15]

[Ber+14] Daniel Bernardes, Mamadou Diaby, Raphaël Fournier, Françoise Fogelman-Soulié et Emmanuel Viennet. « A Social Formalism and Survey for Recommender Systems ». In : *SIGKDD Explorations* 16.2 (2014), p. 20-37. doi : 10.1145/2783702.2783705. [Cité page 63]

[Ber15] Aurélien Berra. « Pour une histoire des humanités numériques ». In : *Critique* 8 (2015), p. 613-626. [Cité page 9]

[BSH23] Samarth Bhargav, Anne Schuth et Claudia Hauff. « When the Music Stops : Tip-of-the-Tongue Retrieval for Music ». In : *arXiv preprint arXiv :2305.14072* (2023). [Cité page 54]

[Blo+08] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte et Etienne Lefebvre. « Fast unfolding of communities in large networks ». In : *Journal of Statistical Mechanics : Theory and Experiment* 2008.10 (2008), P10008. [Cité pages 61, 69]

[Bog+11] Dmitry Bogdanov, Joan Serrà, Nicolas Wack, Perfecto Herrera et Xavier Serra. « Unifying low-level and high-level music similarity measures ». In : *IEEE Transactions on Multimedia* 13.4 (2011), p. 687-701. [Cité page 16]

[BMZ11] Johan Bollen, Huina Mao et Xiaojun Zeng. « Twitter mood predicts the stock market ». In : *Journal of Computational Science* 2.1 (2011), p. 1-8. [Cité page 61]

[Bro+20] Andrew Brown, Weidi Xie, Vicky Kalogeiton et Andrew Zisserman. « Smooth-AP : Smoothing the Path Towards Large-Scale Image Retrieval ». In : *ECCV*. 2020. [Cité pages 70, 71, 87]

[Bru+19] Sebastian Bruch, Masrour Zoghi, Michael Bendersky et Marc Najork. « Revisiting approximate metric optimization in the age of deep neural networks ». In : *ACM SIGIR*. 2019. [Cité page 71]

[BS12] Jürgen Buder et Christina Schwind. « Learning with personalized recommender systems : A psychological view ». In : *Computers in Human Behavior* 28.1 (2012), p. 207-216. [Cité page 60]

[BFD15] John Ashley Burgoyne, Ichiro Fujinaga et J Stephen Downie. « Music information retrieval ». In : *A new companion to digital humanities* (2015), p. 213-228. [Cité pages 53, 55]

[Bur02] Robin Burke. « Hybrid recommender systems : Survey and experiments ». In : *User modeling and user-adapted interaction* 12.4 (2002), p. 331-370. [Cité page 65]

[Bur+22] Robin Burke, Nicholas Mattei, Vladislav Grozin, Amy Voida et Nasim Sonboli. « Multi-agent Social Choice for Dynamic Fairness-aware Recommendation ». In : *Adjunct Proceedings of the 30th ACM Conference on User Modeling, Adaptation and Personalization*. UMAP '22 Adjunct. Barcelona, Spain : ACM, 2022, p. 234-244. doi : 10.1145/3511047.3538032. [Cité page 83]

[BVM14] Robin Burke, Fatemeh Vahedian et Bamshad Mobasher. « Hybrid recommendation in heterogeneous networks ». In : *International Conference on User Modeling, Adaptation, and Personalization*. Springer. 2014, p. 49-60. [Cité page 69]

[Bur20] Lou Burnard. *Du Literary and linguistic computing aux Digital Humanities : retour sur 40 ans de relations entre sciences humaines et informatique*. Date de consultation : 15 Décembre 2023. 2020. [Cité page 8]

[BVB02] MuseScore BVBA. *MuseScore*. Date de consultation : 15 Novembre 2023. 2002. [Cité page 15]

[CR18] Jorge Calvo-Zaragoza et David Rizo. « End-to-End Neural Optical Music Recognition of Monophonic Scores ». In : *Applied Sciences* 8 (avr. 2018), p. 606. doi : 10.3390/app8040606. [Cité page 14]

[Cao+07] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai et Hang Li. « Learning to Rank : From Pairwise Approach to Listwise Approach ». In : *ACM ICML*. 2007. [Cité page 88]

[CCM18] Dominique Cardon, Jean-Philippe Cointet et Antoine Mazières. « La revanche des neurones. L'invention des machines inductives et la controverse de l'intelligence artificielle ». fr. In : *Réseaux* 211.5 (2018). Place : Paris Publisher : La Découverte, p. 173-220. doi : 10.3917/res.211.0173. [Cité page 55]

[CP07] Dominique Cardon et Christophe Prieur. « Les réseaux de relations sur Internet : un objet de recherche pour l'informatique et les sciences sociales ». In : *Humanités numériques* 1 (2007), p. 147-164. [Cité page 63]

[Cas+08] MA Casey, Remco Veltkamp, Masataka Goto, Marc Leman, Christophe Rhodes et Malcolm Slaney. « Content-based music information retrieval : current directions and future challenges ». eng. In : *Proceedings of the IEEE* 96.4 (2008), p. 668-696. [Cité page 16]

[CJ06] Chuan-Wang Chang et Hewijin Christine Jiau. « An Efficient Numeric Indexing Technique for Music Retrieval System ». In : *Proc. IEEE Intl Conf. on Multimedia and Expo (ICME)*. 2006, p. 1981-1984. [Cité page 15]

[Cha12] David Chavalarias. « La société recommandée ». In : *Conflits des interprétations dans la société de l'information : éthique et politique de l'environnement*. Lavoisier, 2012, p. 26-39. [Cité page 63]

[Cha23] David Chavalarias. *Toxic data*. Paris : Flammarion, 2023. [Cité pages 63, 96]

[Che18] Andrée Chedid. *Rythmes*. Paris : Gallimard, 2018. [Cité page 43]

[Che+20a] Chong Chen, Min Zhang, Yongfeng Zhang, Yiqun Liu et Shaoping Ma. « Efficient neural matrix factorization without sampling for recommendation ». In : *ACM Transactions on Information Systems* 38.2 (jan. 2020), p. 14. doi : 10.1145/3373807. [Cité pages 90, 91]

[Che+20b] Lei Chen, Le Wu, Richang Hong, Kun Zhang et Meng Wang. « Revisiting Graph based Collaborative Filtering : A Linear Residual Graph Convolutional Network Approach ». In : *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence* (jan. 2020), p. 27-34. doi : 10.1609/aaai.v34i01.5330. [Cité pages 70, 90, 91]

[CC05] Elaine Chew et Yun-Ching Chen. « Real-time pitch spelling using the spiral array ». In : *Computer Music Journal* 29.2 (2005), p. 61-76. [Cité page 10]

[Neuma] Cnam. *Neuma on-line digital library*. 2009. eprint : http://neuma.huma-num.fr/. [Cité pages 15, 40]

[CD17] Andrea Cogliati et Zhiyao Duan. « A Metric for Music Notation Transcription Accuracy ». In : *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*. 2017, p. 407-413. [Cité page 15]

[Col+06] Vittoria Colizza, Alain Barrat, Marc Barthélemy et Alessandro Vespignani. « The role of the airline transportation network in the prediction and predictability of global epidemics ». In : *Proceedings of the National Academy of Sciences of the United States of America* 103.7 (2006), p. 2015. [Cité page 61]

[XQuery] World Wide Web Consortium. *XQuery 3.0 : An XML Query Language*. https ://www.w3.org/TR/xquery-30. 2007. [Cité page 44]

[Con+11] Camélia Constantin, Cédric du Mouza, Zoé Faget et Philippe Rigaux. « The Melodic Signature Index for Fast Content-based Retrieval of Symbolic Scores Camelia Constantin ». In : *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011, Miami, Florida, USA, October 24-28, 2011*. 2011, p. 363-368. [Cité page 15]

[CCM63] Grosvenor W Cooper, Grosvenor Cooper et Leonard B Meyer. *The rhythmic structure of music*. University of Chicago Press, 1963. [Cité page 10]

[CAS16] Paul Covington, Jay Adams et Emre Sargin. « Deep neural networks for youtube recommendations ». In : *Proceedings of the 10th ACM conference on Recommender Systems*. 2016, p. 191-198. [Cité page 60]

[CMS10] W Bruce Croft, Donald Metzler et Trevor Strohman. *Search engines : Information retrieval in practice*. T. 520. Addison-Wesley Reading, 2010. [Cité page 70]

[CA10] Michael Scott Cuthbert et Christopher Ariza. « Music21 : A Toolkit for Computer-Aided Musicology and Symbolic Music Data ». In : *International Society for Music Information Retrieval Conference (ISMIR)*. 2010, p. 637-642. [Cité page 47]

[DM15] Marin Dacos et Pierre Mounier. *Humanités numériques*. Research Report. Institut français, mars 2015. [Cité page 9]

[Dan+21] Chen Dang, Hicham Randrianarivo, Raphaël Fournier-S'niehotta et Nicolas Audebert. « Web Image Context Extraction with Graph Neural Networks and Sentence Embeddings on the DOM tree ». In : *GEM @ ECML-PKDD 21*. Sept. 2021. [Cité pages 3, 85, 118, 120]

[DFT22] Victor David, Raphaël Fournier-S'niehotta et Nicolas Travers. « Parameterisation of Reasoning on Temporal Markov Logic Networks ». 2022. doi : 10.48550/ARXIV.2211.16414. [Cité pages 3, 118, 120]

[DFT23] Victor David, Raphaël Fournier-S'niehotta et Nicolas Travers. « NeoMaPy : a Framework for Computing MAP inference on Temporal Knowledge Graphs ». In : *IJCAI23 - Demo*. 2023. [Cité pages 3, 118, 120]

[Dem+09] Erik D. Demaine, Shay Mozes, Benjamin Rossman et Oren Weimann. « An optimal decomposition algorithm for tree edit distance ». In : *ACM Trans. Algorithms* 6.1 (2009), 2 :1-2 :19. [Cité page 38]

[Des05] Antoinette Deshoulières. « Chanson : Ah ! que je sens d'inquiétude ! » In : *L'Enchantement des chagrins – Poésies complètes*. Sous la dir. de Catherine Hémon-Fabre et Pierre-Eugène Leroy. Paris : Bartillat, 2005. [Cité page 22]

[DK11] Christian Desrosiers et George Karypis. « A comprehensive survey of neighborhood-based recommendation methods ». In : *Recommender systems handbook*. Springer, 2011, p. 107-144. [Cité page 69]

[DJR22] Martin Digard, Florent Jacquemard et Lydia Rodriguez-de la Nava. « Automated Transcription of Electronic Drumkits ». In : *4 th International Workshop on Reading Music Systems*. 2022, p. 37. [Cité page 54]

[DR04] Shyamala Doraisamy et Stefan M. Rüger. « A Polyphonic Music Retrieval System Using N-Grams ». In : *International Society for Music Information Retrieval Conference (ISMIR)*. 2004, p. 204-209. [Cité page 15]

[Dow99] J. Stephen Downie. « Evaluating a simple approach to music information retrieval : Conceiving melodic n-grams as text ». Thèse de doctorat. Univ. Western Ontario, 1999. [Cité page 15]

[DCB11] Avinava Dubey, Soumen Chakrabarti et Chiranjib Bhattacharyya. « Diversity in ranking via resistive graph centers ». In : *Proceedings of the 17th ACM SIGKDD international conference on Knowledge Discovery and Data mining*. ACM. 2011, p. 78-86. [Cité page 69]

[Ela] Elastic co. *ElasticSearch*. https://www.elastic.co/elasticsearch. [Cité page 33]

[Ell54] Jacques Ellul. *La Technique ou l'Enjeu du siècle*. Armand Colin, 1954. [Cité page 62]

[ERP17] Jeff Ens, Bernhard E. Riecke et Philippe Pasquier. « The Significance of the Low Complexity Dimension in Music Similarity Judgements ». In : *International Society for Music Information Retrieval Conference (ISMIR)*. 2017, p. 31-38. [Cité page 16]

[ER60] Paul Erdős et Alfréd Rényi. « On the evolution of random graphs ». In : *Publications of the Mathematical Institute of the Hungarian Academy of Sciences* 5.1 (1960), p. 17-60. [Cité page 61]

[FR10] Zoé Faget et Philippe Rigaux. « A Database Approach to Symbolic Music Content Management ». In : *Intl. Symp. on Exploring Music Contents (CMMR)*. 2010, p. 303-320. [Cité page 15]

[Fei21] Laurent Feisthauer. « Annotation automatisée des métadonnées structurelles dans les partitions musicales : cas des modulations et des cadences pour la forme sonate ». Thèse de doctorat. École doctorale Sciences pour l'ingénieur (Lille), mai 2021. [Cité page 54]

[FH07] Daniel M Fleder et Kartik Hosanagar. « Recommender systems and their impact on sales diversity ». In : *Proceedings of the 8th ACM conference on Electronic commerce*. ACM. 2007, p. 192-199. [Cité page 70]

[Fob+13] Dominique Fober, Stéphane Letz, Yann Orlarey et Frédéric Bevilacqua. « Programming Interactive Music Scores with INScore ». In : *Sound and Music Computing*. Stockholm, Sweden, juill. 2013, p. 185-190. [Cité page 13]

[FOL12] Dominique Fober, Yann Orlarey et Stéphane Letz. « SCORES LEVEL COMPOSITION BASED ON THE Guido MUSIC NOTATION ». In : *Proceedings of the International Computer Music Conference*. Sous la dir. d'ICMA. 13 sept. 2012, p. 383-386. [Cité page 13]

[Fos20] Francesco Foscarin. « La partition musicale : un objectif ambitieux pour la transcription automatique de la musique ». Theses. Conservatoire national des arts et metiers - CNAM, déc. 2020. [Cité page 14]

[FAF21] Francesco Foscarin, Nicolas Audebert et Raphaël Fournier-S'Niehotta. « PKSpell : Data-Driven Pitch Spelling and Key Signature Estimation ». In : *International Society for Music Information Retrieval Conference (ISMIR)*. Online, India, nov. 2021. [Cité pages 56, 117, 119]

[Fos+18a] Francesco Foscarin, David Fiala, Florent Jacquemard, Philippe Rigaux et Virginie Thion. « Gioqoso, an online Quality Assessment Tool for Music Notation ». In : *4th International Conference on Technologies for Music Notation and Representation (TENOR'18)*. 2018. [Cité page 14]

[FFJ19a] Francesco Foscarin, Raphaël Fournier-S'niehotta et Florent Jacquemard. « A diff procedure for XML music score files ». In : *6th International Conference on Digital Libraries for Musicology (DLfM 2019)*. Nov. 2019. [Cité pages 19, 25, 50, 117, 119]

[FFJ19b] Francesco Foscarin, Raphaël Fournier-S'niehotta et Florent Jacquemard. « Computation and visualization of differences between two XML music score files ». In : *20th International Society for Music Information Retrieval Conference, ISMIR 2019 (Late-breaking Demo)*. 2019. [Cité pages 19, 117, 119]

[Fos+18b] Francesco Foscarin, Raphaël Fournier-S'niehotta, Florent Jacquemard et Philippe Rigaux. « Évaluation de la correction rythmique des partitions numérisées ». In : *JIM 2018 - Journées d'Informatique Musicale*. Amiens, France, mai 2018, p. 87-95. [Cité pages 117, 119]

[Fos+20] Francesco Foscarin, Andrew Mcleod, Philippe Rigaux, Florent Jacquemard et Masahiko Sakai. « ASAP : a dataset of aligned scores and performances for piano transcription ». In : *International Society for Music Information Retrieval Conference*. CONF. 2020, p. 534-541. [Cité page 55]

[FRT18] R. Fournier-S'niehotta, P. Rigaux et N. Travers. « Modeling Music as Synchronized Time Series : Application to Music Score Collections ». In : *Information Systems* (73 2018), p. 35-49. [Cité pages 29, 47]

[Fou12] Raphaël Fournier-S'niehotta. « Détection et analyse d'une thématique rare dans de grands ensembles de requêtes : l'activité pédophile dans le P2P ». Thèse de doctorat. UPMC, 2012. [Cité page 1]

[FRT16a] Raphaël Fournier-S'niehotta, Philippe Rigaux et Nicolas Travers. « A Digital Score Library Based on MEI ». In : *(MEC'16) Music Encoding Conference*. https://music-encoding.org/community/conference/. Montréal, Canada, mai 2016, p. 1-4. [Cité page 19]

[FRT16b] Raphaël Fournier-S'niehotta, Philippe Rigaux et Nicolas Travers. « Is There a Data Model in Music Notation ? » In : *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'16*. Sous la dir. de Richard Hoadley, Chris Nash et Dominique Fober. Cambridge, UK : Anglia Ruskin University, mai 2016, p. 85-91. [Cité pages 18, 21]

[FRT16c] Raphaël Fournier-S'niehotta, Philippe Rigaux et Nicolas Travers. « Querying Music Notation ». In : *23rd International Symposium on Temporal Representation and Reasoning, TIME 2016, Kongens Lyngby, Denmark, October 17-19, 2016*. Sous la dir. de Curtis E. Dyreson, Michael R. Hansen et Luke Hunsberger. IEEE, 2016, p. 51-59. doi : 10.1109/TIME.2016.13. [Cité pages 18, 19, 117, 119]

[FRT16d] Raphaël Fournier-S'niehotta, Philippe Rigaux et Nicolas Travers. « Querying XML Score Databases : XQuery is not Enough ! » In : *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016, New York City, United States, August 7-11, 2016*. Sous la dir. de Michael I. Mandel, Johanna Devaney, Douglas Turnbull et George Tzanetakis. 2016, p. 723-729. [Cité pages 18, 21, 117, 119]

[FRT16e] Raphaël Fournier-S'niehotta, Philippe Rigaux et Nicolas Travers. « Vers un Traitement Algébrique de la Notation Musicale ». In : *JIM 2016 - Journées d'Informatique Musicale*. T. 23. Albi, France, mars 2016, p. 1-9. [Cité pages 19, 21]

[FRT17] Raphaël Fournier-S'niehotta, Philippe Rigaux et Nicolas Travers. « Modeling Music as Synchronized Time Series : Application to Music Score Collections ». In : *(IS'18) Information Systems* (2017), p. 1-36. doi : 10.1016/j.is.2017.12.003. [Cité pages 18, 21, 117, 119]

[GSD08] Joachim Ganseman, Paul Scheunders et Wim D'haes. « Using XQuery on MusicXML Databases for Musicological Analysis ». In : *International Society for Music Information Retrieval Conference (ISMIR)*. 2008. [Cité page 15]

[Gau+16] Noé Gaumont, Tiphaine Viard, Raphaël Fournier, Quina Wang et Matthieu Latapy. « Analysis of the temporal and structural features of threads in a mailing-list ». In : *Complex Networks VII - Proceedings of the 7th Workshop on Complex Networks CompleNet 2016, Dijon, France, March 23-25, 2016*. 2016. doi : 10.1007/978-3-319-30569-1_8. [Cité pages 3, 117, 119]

[Gil14] Tarleton Gillespie. « The Relevance of Algorithms ». In : *Media Technologies*. MIT Press, jan. 2014, p. 167-194. [Cité page 63]

[Gin21] Corrado Gini. « Measurement of inequality of incomes ». In : *The Economic Journal* 31.121 (1921), p. 124-126. [Cité page 74]

[Gir+14] Mathieu Giraud, Florence Levé, Florent Mercier, Marc Rigaudière et Donatien Thorez. « Towards modeling texture in symbolic data ». In : *International Society for Music Information Retrieval Conference-ISMIR*. 2014, p. 59-64. [Cité page 8]

[Gol05] Jennifer Ann Golbeck. *Computing and applying trust in web-based social networks*. University of Maryland, College Park, 2005. [Cité page 69]

[Goo01] Michael Good. « MusicXML : An internet-friendly format for sheet music ». In : *Xml conference and expo*. 2001, p. 03-04. [Cité page 12]

[GC11] Nicholas J Gotelli et Robert K Colwell. « Estimating species richness ». In : *Biological diversity : frontiers in measurement and assessment* 12 (2011), p. 39-54. [Cité pages 74, 75]

[Gou11] Elaine Gould. *Behind Bars*. Faber Music, 2011, p. 676. [Cité page 18]

[Gra73] Mark S Granovetter. « The strength of weak ties ». In : *American Journal of Sociology* 78.6 (1973), p. 1360-1380. [Cité page 61]

[Wik] Wikipedia. *Graph Theory*. https://en.wikipedia.org/wiki/Graph_theory#History. [Cité page 61]

[GL16] Aditya Grover et Jure Leskovec. « Node2vec : Scalable Feature Learning for Networks ». In : (2016). [Cité pages 90, 91]

[Gui19] Éric Guichard. « Les humanités numériques n'existent pas ». A paraître dans : llouz, Charles ; Huerta, Antoine, "Amériques-Europe, les humanités numériques en partage ? Enjeux, innovations et perspectives", Les Indes Savantes. Août 2019. [Cité page 9]

[Gui20] Christophe Guillotel-Nothmann. « Les signes musicaux et leur étude par l'informatique. Le statut épistémologique du numérique dans l'appréhension du sens et de la signification en musique ». In : *Revue musicale OICRM* 6.2 (2020), p. 45-72. [Cité page 9]

[HYL17] William L Hamilton, Zhitao Ying et Jure Leskovec. « Inductive Representation Learning on Large Graphs ». In : *NeurIPS*. 2017. [Cité page 70]

[HK77] Leslie Hannah et John Anderson Kay. *Concentration in modern industry : Theory, measurement and the UK experience*. Springer, 1977. [Cité page 75]

[Haq+19] Albert Haque, Michelle Guo, Prateek Verma et Li Fei-Fei. « Audio-linguistic Embeddings for Spoken Sentences ». In : *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*. IEEE, 2019, p. 7355-7359. doi : 10.1109/ICASSP.2019.8682553. [Cité page 56]

[Har69] Frank Harary. *Graph Theory*. Addison-Wesley Reading, 1969. [Cité page 61]

[HK16] F Maxwell Harper et Joseph A Konstan. « The Movielens datasets : History and context ». In : *ACM Transactions on Interactive Intelligent Systems (TIIS)* 5.4 (2016), p. 19. [Cité pages 80, 90]

[He+20] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang et Meng Wang. « LightGCN : Simplifying and Powering Graph Convolution Network for Recommendation ». In : *ACM SIGIR*. 2020. [Cité pages 70, 90-92]

[Hec78] Paul Heckel. « A technique for isolating differences between files ». In : *Communications of the ACM* 21.4 (1978), p. 264-268. [Cité pages 14, 48]

[Her+04] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen et John T Riedl. « Evaluating collaborative filtering recommender systems ». In : *ACM Transactions on Information Systems (TOIS)* 22.1 (2004), p. 5-53. [Cité page 67]

[Hew97] Walter B Hewlett. « MuseData : multipurpose representation ». In : *Beyond MIDI : The handbook of musical codes*. MIT press, 1997, p. 402-447. [Cité page 12]

[Hil73] Mark O Hill. « Diversity and evenness : a unifying notation and its consequences ». In : *Ecology* 54.2 (1973), p. 427-432. [Cité page 75]

[HS19] Petter Holme et Jari Saramäki. *Temporal network theory*. Springer, 2019. [Cité page 61]

[Hoo+98] Holger H Hoos, Keith A Hamel, Kai Renz et Jürgen Kilian. « The GUIDO Notation Format : A Novel Approach for Adequately Representing Score-Level Music ». In : *International Computer Music Conference (ICMC)*. Citeseer, 1998. [Cité page 12]

[Hsi+17] Cheng Kang Hsieh, Longqi Yang, Yin Cui, Tsung Yi Lin, Serge Belongie et Deborah Estrin. « Collaborative metric learning ». In : *26th International World Wide Web Conference, WWW 2017* (2017). [Cité pages 90, 91]

[Hua+21] Tinglin Huang, Yuxiao Dong, Ming Ding, Zhen Yang, Wenzheng Feng, Xinyu Wang et Jie Tang. « MixGCF : An Improved Training Method for Graph Neural Network-based Recommender Systems ». In : *SIGKDD* (2021). [Cité pages 70, 88, 90-92]

[HBM22] Nicolas Hubert, Armelle Brun et Davy Monticolo. « New Ontology and Knowledge Graph for University Curriculum Recommendation ». In : *ISWC 2022 - The 21st International Semantic Web Conference*. Hangzhou / Virtual, China, oct. 2022. [Cité page 64]

[Hub+22] Nicolas Hubert, Pierre Monnin, Armelle Brun et Davy Monticolo. « Knowledge Graph Embeddings for Link Prediction : Beware of Semantics ! » In : *DL4KG@ISWC 2022 : Workshop on Deep Learning for Knowledge Graphs, held as part of ISWC 2022 : the 21st International Semantic Web Conference*. Virtual, China, oct. 2022. [Cité page 64]

[Hud15] Paul Hudak. *The Haskell School of Music – From Signals to Symphonies*. (Version 2.6), jan. 2015. [Cité page 13]

[HM76] James Wayne Hunt et M Douglas MacIlroy. *An algorithm for differential file comparison*. Rapp. tech. Bell Laboratories Murray Hill, 1976. [Cité pages 14, 48]

[HZ11] Neil Hurley et Mi Zhang. « Novelty and diversity in top-n recommendation–analysis and evaluation ». In : *ACM Transactions on Internet Technology (TOIT)* 10.4 (2011), p. 14. [Cité page 69]

[Hur93] David Huron. *The humdrum toolkit : Software for music research*. https://humdrum.org. 1993. [Cité page 12]

[Hur97] David Huron. « Humdrum and Kern : selective feature encoding ». In : *Beyond MIDI : the handbook of musical codes*. MIT Press, Cambridge, MA, 1997. [Cité page 12]

[Jan+13] David Janin, Florent Berthaut, Myriam Desainte-Catherine, Yann Orlarey et Sylvain Salvati. « The T-Calculus : towards a structured programing of (musical) time and space ». In : *Proceedings of the first ACM SIGPLAN workshop on Functional art, music, modeling and design (FARM'13)*. 2013, p. 23-34. [Cité page 13]

[JK02] Kalervo Järvelin et Jaana Kekäläinen. « Cumulated gain-based evaluation of IR techniques ». In : *ACM TOIS* (2002). [Cité page 70]

[Jia+19] Ray Jiang, Silvia Chiappa, Tor Lattimore, Andras Agyorgy et Pushmeet Kohli. « Degenerate Feedback Loops in Recommender Systems ». In : *arXiv preprint arXiv :1902.10730* (2019). [Cité pages 69, 80]

[JDE07] Malo Cameron Jones, J. Stephen Downie et Andreas F. Ehmann. « Human Similarity Judgments : Implications for the Design of Formal Evaluations ». In : *International Society for Music Information Retrieval Conference (ISMIR)*. 2007, p. 539-542. [Cité page 16]

[Jos06] Lou Jost. « Entropy and diversity ». In : *Oikos* 113.2 (2006), p. 363-375. [Cité page 75]

[Kar+22] Yannis Karmim, Elias Ramzi, Raphaël Fournier-S'niehotta et Nicolas Thome. « LARGO : Direct Optimization of Ranking-based Metrics for Neural Graph Recommendation ». 2022. [Cité pages 3, 65, 118, 119]

[KW23] Emmanouil Karystinaios et Gerhard Widmer. « Roman Numeral Analysis with Graph Neural Networks : Onset-wise Predictions from Note-wise Features ». In : *arXiv preprint arXiv :2307.03544* (2023). [Cité page 56]

[KM11] Nauman Ali Khan et Mubashar Mushtaq. « Open issues on query by humming ». In : *Fourth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2011)*. IEEE. 2011, p. 147-152. [Cité page 16]

[KJR14] Ben King, Rahul Jha et Dragomir R Radev. « Heterogeneous networks and their applications : Scientometrics, name disambiguation, and topic modeling ». In : *Transactions of the Association for Computational Linguistics* 2 (2014), p. 1-14. [Cité page 69]

[KW16] Thomas N. Kipf et Max Welling. « Variational Graph Auto-Encoders ». In : *arXiv preprint arXiv :1611.07308* (2016). [Cité page 89]

[KW17] Thomas N. Kipf et Max Welling. « Semi-Supervised Classification with Graph Convolutional Networks ». In : *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. Toulon, France : OpenReview.net, 2017. [Cité pages 70, 90, 92]

[Kla06] Anssi Klapuri. « Introduction to music transcription ». In : *Signal Processing Methods for Music Transcription* (2006), p. 3-20. [Cité page 10]

[Kle00] Jon M Kleinberg. « Navigation in a small world ». In : *Nature* 406.6798 (2000), p. 845-845. [Cité page 61]

[KB07] Ian Knopke et Donald Byrd. « Towards musicdiff : A foundation for improved optical music recognition using multiple recognizers ». In : *Dynamics* 85.165 (2007), p. 121. [Cité page 15]

[Kön90] Dénes König. *Theory of Finite and Infinite Graphs*. Boston, MA : Birkhäuser, 1990, p. 45-421. [Cité page 61]

[KSH17] Alex Krizhevsky, Ilya Sutskever et Geoffrey E. Hinton. « ImageNet Classification with Deep Convolutional Neural Networks ». In : *Commun. ACM* 60.6 (mai 2017), p. 84-90. doi : 10.1145/3065386. [Cité page 55]

[KP17] Matevž Kunaver et Tomaž Požrl. « Diversity in recommender systems – A survey ». In : *Knowledge-Based Systems* 123 (2017), p. 154-162. [Cité page 69]

[LCB16] Amaury L'Huillier, Sylvain Castagnos et Anne Boyer. « The New Challenges when Modeling Context through Diversity over Time in Recommender Systems ». In : *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization*. ACM. 2016, p. 341-344. [Cité page 70]

[Lam+08] Renaud Lambiotte, Vincent D Blondel, Cristobald de Kerchove, Etienne Huens, Christophe Prieur, Zbigniew Smoreda et Paul Van Dooren. « Geographical dispersal of mobile communication networks ». In : *Physica A : Statistical Mechanics and its Applications* 387.21 (2008), p. 5317-5325. [Cité page 61]

[LC10a] Ni Lao et William W Cohen. « Fast query execution for retrieval models based on path-constrained random walks ». In : *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2010, p. 881-888. [Cité page 77]

[LC10b] Ni Lao et William W Cohen. « Relational retrieval using a combination of path-constrained random walks ». In : *Machine learning* 81.1 (2010), p. 53-67. [Cité page 77]

[Le 05] Hervé Le Tellier. *Les amnésiques n'ont rien vécu d'inoubliable*. Bègles : Le Castor Astral, 2005. [Cité page 21]

[Lem00] Kjell Lemström. « String Matching Techniques for Music Retrieval ». Thèse de doctorat. University of Helsinki, Department of Computer Science, 2000. [Cité page 15]

[Les23] Jure Leskovec. *Notes du cours CS224W*. http://web.stanford.edu/class/cs224w/. 2023. [Cité page 69]

[Lex+21] Elisabeth Lex, Dominik Kowald, Paul Seitlinger, Thi Ngoc Trang Tran, Alexander Felfernig et Markus Schedl. « Psychology-informed recommender systems ». In : *Foundations and Trends® in Information Retrieval* 15.2 (2021), p. 134-242. [Cité page 60]

[Lia+18] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman et Tony Jebara. « Variational autoencoders for collaborative filtering ». In : *The Web Conference* (2018). [Cité pages 90-92]

[LSY03] G. Linden, B. Smith et J. York. « Amazon.com recommendations : item-to-item collaborative filtering ». In : *IEEE Internet Computing* 7.1 (2003), p. 76-80. doi : 10.1109/MIC.2003.1167344. [Cité page 60]

[LLC06] Project Petrucci LLC. *International Music Score Library Project.* Date de consultation : 15 Novembre 2023. 2006. [Cité page 15]

[Mac65] Robert H MacArthur. « Patterns of species diversity ». In : *Biological reviews* 40.4 (1965), p. 510-533. [Cité pages 74, 75]

[Mah05] Gustav Mahler. *Letters to his wife.* Faber Music, 2005. [Cité page 18]

[Mao+21] Kelong Mao, Jieming Zhu, Jinpeng Wang, Quanyu Dai, Zhenhua Dong, Xi Xiao et Xiuqiang He. « SimpleX : A Simple and Strong Baseline for Collaborative Filtering ». In : *International Conference on Information and Knowledge Management, Proceedings* (sept. 2021), p. 1243-1252. doi : 10.1145/3459637.3482297. [Cité pages 90, 91]

[ML16] Naoki Masuda et Renaud Lambiotte. *A Guide to Temporal Networks.* World Scientific, 2016. [Cité page 61]

[Mat22] Jawher Matmati. « Au-delà de la notation ». Mém. de mast. Conservatoire National Supérieur de Musique et de Danse de Paris, 2022. [Cité pages 14, 17]

[McA22] Julian McAuley. *Personalized machine learning.* Cambridge University Press, 2022. [Cité page 66]

[McC02] James McCartney. « Rethinking the computer music language : Super collider ». In : *Computer Music Journal* 26.4 (2002), p. 61-68. [Cité page 14]

[MF06] Cory McKay et Ichiro Fujinaga. « jSymbolic : A Feature Extractor for MIDI Files. » In : *ICMC.* 2006. [Cité page 8]

[MS18] Andrew Mcleod et Mark Steedman. « Evaluating Automatic Polyphonic Music Transcription ». In : *International Society for Music Information Retrieval Conference (ISMIR).* 2018, p. 42-49. [Cité page 15]

[MK43] Luigi Federico Menabrea et Ada King Countess of Lovelace. *Sketch of the Analytical Engine Invented by Charles Babbage, Esq.* Richard et John E. Taylor, 1843. [Cité page 7]

[Men+20] Zaiqiao Meng, Richard McCreadie, Craig Macdonald et Iadh Ounis. « Exploring Data Splitting Strategies for the Evaluation of Recommendation Models ». In : *Proceedings of the 10th ACM conference on Recommender Systems* (2020). [Cité page 89]

[Mic+22] Lien Michiels, Jens Leysen, Annelien Smets et Bart Goethals. « What Are Filter Bubbles Really ? A Review of the Conceptual and Empirical Work ». In : *Adjunct Proceedings of the 30th ACM Conference on User Modeling, Adaptation and Personalization.* UMAP '22 Adjunct. Barcelona, Spain : Association for Computing Machinery, 2022, p. 274-279. doi : 10.1145/3511047.3538028. [Cité page 83]

[Mil67] Stanley Milgram. « The small world problem ». In : *Psychology today* 2.1 (1967), p. 60-67. [Cité page 61]

[Möl+20] Judith Möller, Damian Trilling, Natali Helberger et Bram van Es. « Do not blame it on the algorithm : an empirical assessment of multiple recommender systems and their impact on content diversity ». In : *Digital media, political polarization and challenges to democracy.* Routledge, 2020, p. 45-63. [Cité page 62]

[MS90] Marcel Mongeau et David Sankoff. « Comparison of musical sequences ». In : *Computers and the Humanities* 24.3 (1990), p. 161-175. [Cité page 15]

[Mur09] Tsuyoshi Murata. « Modularities for bipartite networks ». In : *Proceedings of the 20th ACM conference on Hypertext and hypermedia - HT '09* (2009), p. 245. [Cité page 69]

[NMM18] Sharad Nandanwar, Aayush Moroney et M Narasimha Murty. « Fusing Diversity in Recommendations in Heterogeneous Information Networks ». In : *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM. 2018, p. 414-422. [Cité pages 69, 80]

[Nan+09] Alexandros Nanopoulos, Dimitrios Rafailidis, Maria M. Ruxanda et Yannis Manolopoulos. « Music search engines : Specifications and challenges ». In : *Information Processing & Management* 45.3 (2009), p. 392-396. doi : 10.1016/j.ipm.2009.02.002. [Cité page 15]

[NO04] Giovanna Neve et Nicola Orio. « Indexing and Retrieval of Music Documents through Pattern Analysis and Data Fusion Techniques ». In : *International Society for Music Information Retrieval Conference (ISMIR)*. 2004, p. 216-223. [Cité page 15]

[Ngo17] Chimamanda Ngozi Adichie. *Chère Ijeawele, ou Un manifeste pour une éducation féministe*. Trad. par Marguerite Capelle. Paris : Gallimard, 2017. [Cité page 73]

[NNc16] Han-Wen Nienhuys, Jan Nieuwenhuizen et contributors. *GNU Lilypond*. http ://lilypond.org/. 2016. [Cité page 12]

[OWA10] I Oppenheim, C Walshaw et J Atchley. *The abc standard 2.0*. https://abcnotation.com/wiki/abc:standard:v2.0. 2010. [Cité page 12]

[Pag+98a] Lawrence Page, Sergey Brin, Rajeev Motwani et Terry Winograd. *The pagerank citation ranking : Bring order to the web*. Rapp. tech. Technical report, stanford University, 1998. [Cité page 37]

[Pag+98b] Lawrence Page, Sergey Brin, Rajeev Motwani et Terry Winograd. « The PageRank Citation Ranking : Bringing Order to the Web ». In : *World Wide Web Internet And Web Information Systems* (1998). [Cité page 88]

[Par+08] Bryan Pardo, David Little, Rui Jiang, Hagai Livni et Jinyu Han. « The vocalsearch music search engine ». In : *Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries*. 2008, p. 430-430. [Cité page 16]

[Par11] Eli Pariser. *The filter bubble : How the new personalized web is changing what we read and how we think*. Penguin, 2011. [Cité pages 62, 70]

[PM12] Stéphanie Peltier et François Moreau. « Internet and the "long tail versus superstar effect" debate : evidence from the French book market ». In : *Applied Economics Letters* 19.8 (2012), p. 711-715. [Cité page 63]

[PAS14] Bryan Perozzi, Rami Al-Rfou et Steven Skiena. « DeepWalk : Online Learning of Social Representations ». In : *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2014, p. 701-710. doi : 10.1145/2623330.2623732. [Cité pages 70, 90, 91]

[PB21] Przemysław Pobrotyn et Radosław Białobrzeski. *NeuralNDCG : Direct Optimisation of a Ranking Metric via Differentiable Relaxation of Sorting*. 2021. [Cité pages 71, 90-92]

[PG23] Andrea Poltronieri et Aldo Gangemi. « The music note ontology ». In : *arXiv preprint arXiv :2304.00986* (2023). [Cité page 13]

[Por23] Xavier de la Porte. « Ecouter Donna Haraway parler de cyborgs sous un arbre ». In : *Le code a changé*. France Inter. 29 sept. 2023. [Cité page 63]

[PT11] O. Post et G.T. Toussaint. « The Edit Distance as a Measure of Perceived Rhythmic Similarity ». In : *Empirical Musicology Review* 6 (2011), p. 164-179. [Cité page 38]

[Pug+12] Laurent Pugin, Johannes Kepper, Perry Roland, Maja Hartwig et Andrew Hankinson. « Separating Presentation and Content in MEI. » In : *ISMIR*. 2012, p. 505-510. [Cité page 18]

[PZR14] Laurent Pugin, Rodolfo Zitellini et Perry Roland. « Verovio : A library for Engraving MEI Music Notation into SVG ». In : *International Society for Music Information Retrieval Conference (ISMIR)*. 2014, p. 107-112. [Cité page 50]

[Raf16] Colin Raffel. « Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching. 2016 ». Thèse de doctorat. Stanford University, 2016. [Cité page 55]

[Rai+07] Yves Raimond, Samer A. Abdallah, Mark B. Sandler et Frederick Giasson. « The Music Ontology ». In : *Proceedings of the 8th International Conference on Music Information Retrieval, ISMIR 2007, Vienna, Austria, September 23-27, 2007*. Sous la dir. de Simon Dixon, David Bainbridge et Rainer Typke. Austrian Computer Society, 2007, p. 417-422. [Cité page 13]

[Ram+21a] Pedro Ramaciotti Morales, Robin Lamarche-Perrin, Raphaël Fournier-S'niehotta, Rémy Poulain, Lionel Tabourier et Fabien Tarissan. « Measuring diversity in heterogeneous information networks ». In : *Theoretical Computer Science* 859 (2021), p. 80-115. doi : 10.1016/j.tcs.2021.01.013. [Cité pages 3, 65, 77, 117, 119]

[RTF20] Pedro Ramaciotti-Morales, Lionel Tabourier et Raphaël Fournier-S'niehotta. « Testing the Impact of Semantics and Structure on Recommendation Accuracy and Diversity ». In : *International Conference on Social Networks Analysis and Mining*. ASONAM. IEEE/ACM. Déc. 2020. [Cité pages 3, 117, 119]

[Ram+21b] Elias Ramzi, Nicolas Thome, Clément Rambour, Nicolas Audebert et Xavier Bitot. « Robust and Decomposable Average Precision for Image Retrieval ». In : *NeurIPS* (2021). [Cité pages 71, 88]

[RDM18] Sabbir M. Rashid, David De Roure et Deborah L. McGuinness. « A Music Theory Ontology ». In : *Proceedings of the 1st International Workshop on Semantic Applications for Audio and Music*. SAAM '18. Monterey, CA, USA : Association for Computing Machinery, 2018, p. 6-14. doi : 10.1145/3243907.3243913. [Cité page 13]

[Reb+12] Ana Rebelo, Ichiro Fujinaga, Filipe Paszkiewicz, Andre RS Marcal, Carlos Guedes et Jaime S Cardoso. « Optical music recognition : state-of-the-art and open issues ». In : *International Journal of Multimedia Information Retrieval* 1 (2012), p. 173-190. [Cité page 14]

[Ren+09] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner et Lars Schmidt-Thieme. « BPR : Bayesian Personalized Ranking from Implicit Feedback ». In : *Uncertainty in Artificial Intelligence*. 2009. [Cité pages 70, 86, 90, 91]

[Rén61] Alfréd Rényi. « On measures of entropy and information ». In : *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1 : Contributions to the Theory of Statistics*. The Regents of the University of California. 1961, p. 547-561. [Cité page 75]

[Res+94] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom et John Riedl. « Grouplens : An open architecture for collaborative filtering of netnews ». In : *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. 1994, p. 175-186. [Cité page 60]

[RF67] Jacques Revaux et Claude François. *Comme d'habitude*. 1967. [Cité page 33]

[Rho93] Stephen A Rhoades. « The Herfindahl-Hirschman index ». In : *Fed. Res. Bull.* 79 (1993), p. 188. [Cité pages 74, 75]

[RRS11] Francesco Ricci, Lior Rokach et Bracha Shapira. *Recommender systems handbook*. Springer, 2011. [Cité page 66]

[Rig+12] Philippe Rigaux, Lylia Abrouk, Hervé Audéon, Nadine Cullot, Cécile Davy-Rigaux, Zoé Faget, Elisabeth Gavignet, David Gross-Amblard, Alice Tacaille et Virginie Thion-Goasdoué. « The design and implementation of Neuma, a collaborative Digital Scores Library : Requirements, architecture, and models ». In : *International Journal on Digital Libraries* 12.2-3 (2012), p. 73-88. [Cité pages 15, 40]

[RCR20] Antonio Ríos-Vila, Jorge Calvo-Zaragoza et David Rizo. « Evaluating simultaneous recognition and encoding for optical music recognition ». In : *7th International Conference on Digital Libraries for Musicology*. 2020, p. 10-17. [Cité page 54]

[Riz10] David Rizo. « Symbolic music comparison with tree data structures ». Thèse de doctorat. Universidad de Alicante, 2010. [Cité page 15]

[Rol02] Perry Roland. « The music encoding initiative (MEI) ». In : *Proceedings of the First International Conference on Musical Applications Using XML*. T. 1060. 2002, p. 55-59. [Cité page 12]

[Rom+23] Hilda Romero-Velo, Susana Ladra, José R Paramá et Fernando Silva-Coira. « Indexing and Retrieval of Scores by Humming based on Extracted Features ». In : *VI Congreso Xove TIC : impulsando el talento científico. Octubre, 2023, A Coruña*. Universidade da Coruña, Servizo de Publicacións. 2023, p. 35-42. [Cité page 54]

[Ros+20] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti et Michael Bronstein. « Temporal Graph Networks for Deep Learning on Dynamic Graphs ». In : *arXiv preprint arXiv :2006.10637* (2020). [Cité page 70]

[RMM20] Camille Roth, Antoine Mazières et Telmo Menezes. « Tubes and bubbles topological confinement of YouTube recommendations ». In : *PloS One* 15.4 (2020). [Cité page 62]

[Sad15] Éric Sadin. *La vie algorithmique. Critique de la raison numérique*. Paris : L'Échappée, 2015. [Cité page 62]

[SDW06] Matthew J Salganik, Peter Sheridan Dodds et Duncan J Watts. « Experimental study of inequality and unpredictability in an artificial cultural market ». In : *Science* 311.5762 (2006), p. 854-856. [Cité page 63]

[San20] Janelle C Sands. « Efficient optical music recognition validation using MIDI sequence data ». Thèse de doctorat. Massachusetts Institute of Technology, 2020. [Cité page 54]

[San16] Arunav Sanyal. « Modulo7 : A Full Stack Music Information Retrieval and Structured Querying Engine ». Thèse de doctorat. Johns Hopkins University, 2016. [Cité page 16]

[SSU08] Susan Schreibman, Ray Siemens et John Unsworth. *A companion to digital humanities*. John Wiley & Sons, 2008. [Cité pages 8, 9]

[Ser+13] Xavier Serra, Michela Magas, Emmanouil Benetos, Magdalena Chudy, Simon Dixon, Arthur Flexer, Emilia Gómez Gutiérrez, Fabien Gouyon, Herrera Boyer et Sergi Jordà Puig. « Roadmap for music information research ». In : (2013). [Cité pages 53, 54]

[Sha48a] Claude E. Shannon. « A mathematical theory of communication ». In : *The Bell System Technical Journal* 27.3 (1948), p. 379-423. doi : 10.1002/j.1538-7305.1948.tb01338.x. [Cité page 75]

[Sha48b] Claude Elwood Shannon. « A mathematical theory of communication ». In : *Bell system technical journal* 27.3 (1948), p. 379-423. [Cité pages 74, 75]

[SF20] Elona Shatri et György Fazekas. « Optical music recognition : State of the art and major challenges ». In : *arXiv preprint arXiv :2006.07885* (2020). [Cité page 14]

[Shi+18] Chuan Shi, Binbin Hu, Wayne Xin Zhao et Philip S Yu. « Heterogeneous information network embedding for recommendation ». In : *IEEE Transactions on Knowledge and Data Engineering* 31.2 (2018), p. 357-370. [Cité pages 69, 70]

[Shi+16] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun et Philip S Yu. « A survey of heterogeneous information network analysis ». In : *IEEE Transactions on Knowledge and Data Engineering* 29.1 (2016), p. 17-37. [Cité page 69]

[Shi+15] Chuan Shi, Zhiqiang Zhang, Ping Luo, Philip S Yu, Yading Yue et Bin Wu. « Semantic path based personalized recommendation on weighted heterogeneous information networks ». In : *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM. 2015, p. 453-462. [Cité page 69]

[Sil+16] Diego Furtado Silva, Chin-Chia Michael Yeh, Gustavo E. A. P. A. Batista et Eamonn J. Keogh. « SiMPle : Assessing Music Similarity Using Subsequences Joins ». In : *International Society for Music Information Retrieval Conference (ISMIR)*. 2016. [Cité page 16]

[Apa] Apache Foundation. *Apache Solr*. https://solr.apache.org/. [Cité page 33]

[Sou52] Répertoire International des Sources Musicales (RISM). *Base de données du Répertoire International des Sources Musicales (RISM)*. Date de consultation : 15 Novembre 2023. 1952. [Cité page 15]

[St 20] Emily St John Mandel. *The Glass Hotel*. New York : HarperCollins, 2020. [Cité page 31]

[Sti07] Andy Stirling. « A general framework for analysing diversity in science, technology and society ». In : *Journal of the Royal Society Interface* 4.15 (2007), p. 707-719. [Cité page 74]

[Sun+20] Jianing Sun, Yingxue Zhang, Wei Guo, Huifeng Guo, Ruiming Tang, Xiuqiang He, Chen Ma et Mark Coates. « Neighbor Interaction Aware Graph Convolution Networks for Recommendation ». In : *SIGIR 2020 - Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (juill. 2020), p. 1289-1298. doi : 10.1145/3397271.3401123. [Cité pages 70, 90, 91]

[Sun+11] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S Yu et Tianyi Wu. « Pathsim : Meta path-based top-k similarity search in heterogeneous information networks ». In : *Proceedings of the VLDB Endowment* 4.11 (2011), p. 992-1003. [Cité page 70]

[SW21] Bo-Hsun Sung et Shih-Chieh Wei. « BECMER : A Fusion Model Using BERT and CNN for Music Emotion Recognition ». In : *2021 IEEE 22nd International Conference on Information Reuse and Integration for Data Science (IRI)*. 2021, p. 437-444. doi : 10.1109/IRI51335.2021.00068. [Cité page 56]

[Tan+15] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan et Qiaozhu Mei. « LINE : Large-scale Information Network Embedding ». In : *WWW 2015 - Proceedings of the 24th International Conference on World Wide Web* (mars 2015), p. 1067-1077. doi : 10.1145/2736277.2741093. [Cité pages 90, 91]

[Tan+12] Jie Tang, Sen Wu, Jimeng Sun et Hang Su. « Cross-domain collaboration recommendation ». In : *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2012, p. 1285-1293. [Cité page 69]

[THL13] Jiliang Tang, Xia Hu et Huan Liu. « Social recommendation : a review ». In : *Social Network Analysis and Mining* 3.4 (2013), p. 1113-1133. [Cité page 69]

[Tar19] Fabien Tarissan. *Au cœur des réseaux. Des sciences aux citoyens*. Paris : Le Pommier, 2019. [Cité page 61]

[TME93] Daniel Taupin, Ross Mitchell et Andreas Egler. « MusiXTEX. Using TEX to write polyphonic or instrumental music ». In : *EuroTEX* 92 (1993), p. 257-272. [Cité page 12]

[Tho+12] Verena Thomas, David Damm, Christian Fremerey, Michael Clausen, Frank Kurth et Meinard Müller. « PROBADO music : A multimodal online music library ». In : *ICMC*. Citeseer. 2012. [Cité page 16]

[TJB09] Andreas Töscher, Michael Jahrer et Robert M Bell. « The BigChaos solution to the Netflix grand prize ». In : *Netflix prize documentation* (2009), p. 1-52. [Cité page 60]

[Tre+22] Celina Treuillier, Sylvain Castagnos, Evan Dufraisse et Armelle Brun. « Being Diverse is Not Enough : Rethinking Diversity Evaluation to Meet Challenges of News Recommender Systems ». In : *FairUMAP 2022 - Fairness in User Modeling, Adaptation and Personalization*. Barcelone, Spain, juill. 2022. doi : 10.1145/3511047.3538030. [Cité page 83]

[Typ97] Rainer Typke. *Musipedia*. Date de consultation : 15 Novembre 2023. 1997. [Cité pages 15, 56]

[TWV05] Rainer Typke, Frans Wiering et Remco C Veltkamp. « A survey of music information retrieval systems ». In : *International Society for Music Information Retrieval Conference (ISMIR)*. 2005, p. 153-160. [Cité page 15]

[Var91] Fred Vargas. *L'homme aux cercles bleus*. Paris : Viviane Hamy, 1991. [Cité page 1]

[Vas+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser et Illia Polosukhin. « Attention is all you need ». In : *Advances in neural information processing systems* 30 (2017). [Cité page 56]

[Vel+18] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò et Yoshua Bengio. « Graph attention networks ». In : *ICLR* (2018). [Cité pages 70, 90, 92]

[Via+18] Tiphaine Viard, Raphaël Fournier, Clémence Magnien et Matthieu Latapy. « Discovering Patterns of Interest in IP Traffic Using Cliques in Bipartite Link Streams ». In : *International Workshop on Complex Networks (CompleNet'18)*. Mars 2018. doi : 10.1007/978-3-319-73198-8_20. [Cité pages 3, 118, 119]

[VF19a] Tiphaine Viard et Raphaël Fournier-S'niehotta. « Encoding temporal and structural information in machine learning models for recommandation ». In : *LEG @ ECML-PKDD 19*. Sept. 2019. [Cité pages 3, 118, 119]

[VF19b] Tiphaine Viard et Raphaël Fournier-S'niehotta. « Augmenting content-based rating prediction with link stream features ». In : *Computer Networks* 150 (2019), p. 127-133. doi : 10.1016/j.comnet.2018.12.002. [Cité pages 3, 118, 119]

[Vir11] Vladimir Viro. « Peachnote : Music Score Search and Analysis Platform. » In : *International Society for Music Information Retrieval Conference (ISMIR)*. 2011, p. 359-362. [Cité page 15]

[Wan+19] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng et Tat-Seng Chua. « Neural Graph Collaborative Filtering ». In : *ACM SIGIR* (2019). [Cité pages 70, 90, 91]

[Wan+20] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu et Tat-Seng Chua. « Disentangled Graph Collaborative Filtering ». In : *ACM SIGIR* (2020). [Cité pages 70, 90, 91]

[Wan+21] Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec et Pan Li. « Inductive representation learning in temporal networks via causal anonymous walks ». In : *arXiv preprint arXiv :2101.05974* (2021). [Cité page 93]

[War23] Charlie Warzel. « Nobody Knows What's Happening Online Anymore ». In : *The Atlantic* (18 déc. 2023). [Cité page 96]

[WS98] Duncan J Watts et Steven H Strogatz. « Collective dynamics of 'small-world' networks ». In : *Nature* 393.6684 (1998), p. 440-442. [Cité page 61]

[Wu+21a] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian et Xing Xie. « Self-supervised Graph Learning for Recommendation ». In : *ACM SIGIR* (2021). [Cité page 70]

[Wu+21b] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian et Xing Xie. « Self-supervised Graph Learning for Recommendation ». In : *ACM SIGIR*. 2021. [Cité pages 90, 91]

[Wu+23] Shangda Wu, Dingyao Yu, Xu Tan et Maosong Sun. « CLaMP : Contrastive Language-Music Pre-training for Cross-Modal Symbolic Music Information Retrieval ». In : *arXiv preprint arXiv :2304.11029* (2023). [Cité page 56]

[XZY14] Yun Xiong, Yangyong Zhu et Philip S Yu. « Top-k similarity join in heterogeneous information networks ». In : *IEEE Transactions on Knowledge and Data Engineering* 27.6 (2014), p. 1710-1723. [Cité page 70]

[Xu+19] Keyulu Xu, Weihua Hu, Jure Leskovec et Stefanie Jegelka. « How Powerful are Graph Neural Networks ? » In : *ICLR* (2019). [Cité pages 90, 92]

[YSL12] Xiwang Yang, Harald Steck et Yong Liu. « Circle-based recommendation in online social networks ». In : *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2012, p. 1267-1275. [Cité page 69]

[Yu+14] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick et Jiawei Han. « Personalized entity recommendation : A heterogeneous information network approach ». In : *Proceedings of the 7th ACM international conference on Web search and data mining*. ACM. 2014, p. 283-292. [Cité page 69]

[Yu+13] Xiao Yu, Xiang Ren, Yizhou Sun, Bradley Sturt, Urvashi Khandelwal, Quanquan Gu, Brandon Norick et Jiawei Han. « Recommendation in heterogeneous information networks with implicit user feedback ». In : *Proceedings of the 7th ACM conference on Recommender systems*. ACM. 2013, p. 347-350. [Cité page 70]

[Yue+07] Yisong Yue, Thomas Finley, Filip Radlinski et Thorsten Joachims. « A Support Vector Method for Optimizing Average Precision ». In : *ACM SIGIR*. 2007. [Cité page 71]

[ZL09] R. Zafarani et H. Liu. *Social Computing Data Repository at ASU*. 2009. [Cité page 80]

[Zen+21] Mingliang Zeng, Xu Tan, Rui Wang, Zeqian Ju, Tao Qin et Tie-Yan Liu. « MusicBERT : Symbolic Music Understanding with Large-Scale Pre-Training ». In : *Findings of the Association for Computational Linguistics : ACL/IJCNLP 2021, Online Event, August 1-6, 2021*. Sous la dir. de Chengqing Zong, Fei Xia, Wenjie Li et Roberto Navigli. T. ACL/IJCNLP 2021. Findings of ACL. Association for Computational Linguistics, 2021, p. 791-800. doi : 10.18653/V1/2021.FINDINGS-ACL.70. [Cité page 56]

[Zha+09] Bingjun Zhang, Jialie Shen, Qiaoliang Xiang et Ye Wang. « Compositemap : a novel framework for music similarity measure ». In : *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. 2009, p. 403-410. [Cité page 16]

[Zha+23] Huan Zhang, Emmanouil Karystinaios, Simon Dixon, Gerhard Widmer et Carlos Eduardo Cancino-Chacón. « Symbolic Music Representations for Classification Tasks : A Systematic Evaluation ». In : *arXiv preprint arXiv :2309.02567* (2023). [Cité page 56]

[ZS89] Kaizhong Zhang et Dennis Shasha. « Simple Fast Algorithms for the Editing Distance between Trees and Related Problems ». In : *J. Comput.* 18.6 (1989), p. 1245-1262. doi : 10.1137/0218082. [Cité pages 38, 49]

[Zha+19] Shuai Zhang, Lina Yao, Aixin Sun et Yi Tay. « Deep Learning Based Recommender System : A Survey and New Perspectives ». In : *ACM Comput. Surv.* 52.1 (fév. 2019). doi : 10.1145/3285029. [Cité page 60]

[ZCM02] Yi Zhang, Jamie Callan et Thomas Minka. « Novelty and redundancy detection in adaptive filtering ». In : *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2002, p. 81-88. [Cité page 69]

[Zho+10] Tao Zhou, Zoltán Kuscsik, Jian-Guo Liu, Matúš Medo, Joseph Rushton Wakeling et Yi-Cheng Zhang. « Solving the apparent diversity-accuracy dilemma of recommender systems ». In : *Proceedings of the National Academy of Sciences* 107.10 (2010), p. 4511-4515. [Cité page 69]

[ZFR22] Tiange Zhu, Raphaël Fournier-S'niehotta et Philippe Rigaux. « Computation and visualization of differences between two XML music score files ». In : *23th International Society for Music Information Retrieval Conference, ISMIR 2019 (Late-breaking Demo)*. 2022. [Cité pages 117, 119]

[Zhu+22] Tiange Zhu, Raphaël Fournier-S'niehotta, Philippe Rigaux et Nicolas Travers. « A Framework for Content-Based Search in Large Music Collections ». In : *Big Data and Cognitive Computing* 6.1 (2022). doi : 10.3390/bdcc6010023. [Cité pages 19, 21, 37, 40, 55, 117, 119]

[Zie+05] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan et Georg Lausen. « Improving recommendation lists through topic diversification ». In : *Proceedings of the 14th international conference on World Wide Web*. ACM. 2005, p. 22-32. [Cité pages 69, 80]

Résumé en français

Mes travaux récents s'articulent dans deux directions de recherche : l'informatique musicale symbolique et l'intelligence artificielle.

En informatique musicale symbolique, j'ai travaillé avec Philippe Rigaux et Nicolas Travers sur la modélisation de partitions musicales encodées en XML, vues comme des séries temporelles synchronisées, pour lesquelles nous avons proposé une algèbre de manipulation et un langage d'interrogation reposant sur XQuery [FRT16c ; FRT16d ; FRT17]. Ensuite, dans le cadre du doctorat de Francesco Foscarin, nous avons travaillé sur la question de la transcription automatique de la musique, c'est-à-dire la production d'une partition musicale (représentation lisible par les humains) à partir de documents MIDI. Nous avons proposé une nouvelle modélisation du contenu musical encodé dans une partition à l'aide d'arbres, puis utilisé des distances d'édition adaptées aux séquences et aux arbres pour comparer efficacement des partitions [FFJ19a ; FFJ19b ; Fos+18b]. Ces travaux visent une application équivalentes à l'outil `diff` pour du texte. Enfin, j'ai étudié un autre problème d'informatique musicale, avec de l'intelligence artificielle (réseaux neuronaux récurrents, l'estimation de clé (*key-signature estimation*) et la reconnaissance des hauteurs (*pitch-spelling*). Nous parvenons ainsi à améliorer la transcription depuis le format MIDI [FAF21].

Plus récemment, dans le cadre du projet Polifonia, j'étudie et propose des méthodes pour la recherche d'information de contenu musicaux symboliques. Nous travaillons, dans le doctorat de Tiange Zhu, à la mise au point d'un moteur de recherche[1]. Le cœur du moteur repose sur une modélisation efficace du contenu musical à partir de *n-grams*, ce qui permet l'intégration dans des moteurs de recherche de texte ainsi que l'interrogation de diverses dimensions de la musique [Zhu+22 ; ZFR22].

Mon second axe de recherche est centré sur la modélisation de graphes (éventuellement dynamiques), appliquée à divers problèmes dont les systèmes de recommandation.

J'ai étudié, avec plusieurs collègues, le modèle des Heterogeneous Interaction Networks (HIN), une généralisation des graphes. Nous avons examiné leur intérêt pour faire une classification exhaustive des indicateurs de diversité, et appliqué ces résultats aux listes produites par les systèmes de recommandation [Ram+21a ; RTF20]. Depuis 2021, j'encadre le doctorat de Yannis Karmim et nous approchons cette fois le problème de la prise en compte de l'aspect temporel de la recommandation par l'étude de techniques d'apprentissage profond, avec les modèles de *Graph Neural Networks*. Cela rejoint et poursuit des travaux antérieurs sur les graphes dynamiques [Gau+16 ;

[1] Voir la démonstration sur http://neuma-dev.huma-num.fr/.

Via+18 ; VF19a ; VF19b]. Nous avons également étudié diverses manières d'améliorer l'entraînement de ces architectures, en utilisant une fonction de perte incorporant des informations sur le classement, en modifiant l'échantillonnage des exemples et en reconsidérant le protocole d'évaluation inductif [Kar+22]. J'avais préalablement étudié ces modèles sur un problème de recherche d'information pour améliorer l'indexation d'images dans un moteur de recherche, à l'aide d'une détection efficace de leurs légendes dans des pages web [Dan+21].

Dans un deuxième temps, j'ai mené des travaux sur le raisonnement sur des faits représentés dans des graphes, les liens modélisant les conflits éventuels entre faits. Avec Victor David et Nicolas Travers, nous avons proposé une approche originale par rapport à l'état de l'art pour construire des ensembles de faits cohérents entre eux, en maximisant le nombre et les probabilités des faits conservés, au détriment d'un ensemble de faits laissés de côté [DFT22 ; DFT23]. Ce travail trouve son application dans la méthodologie historiographique (faits incertains), mais aussi en systèmes multi-agents (quand ceux-ci doivent raisonner dans des environnements avec des informations incomplètes).

Abstract (in English)

My recent work focuses on two research directions : symbolic music computing and artificial intelligence.

In symbolic music, I have worked with Philippe Rigaux and Nicolas Travers on the modeling of musical scores encoded in XML format, first seen as synchronized time series, for which we proposed an algebra and a query language based on XQuery [FRT16c ; FRT16d ; FRT17]. Then, during Francesco Foscarin's PhD, we worked on the issue of automatic music transcription, i.e. the production of a musical score (human readable representation) from MIDI documents. We have proposed a new modeling of the musical content encoded in a score using a tree-based representation, and then used editing distances adapted to sequences and trees to efficiently compare scores [FFJ19a ; FFJ19b ; Fos+18b]. This work aims at being for musical scores what the `diff` tool is to text. Finally, I have studied two other computer music problems, key-signature estimation and pitch-spelling, with artificial intelligence (recurrent neural networks). We improved the transcription from MIDI format [FAF21] into a score.

More recently, as part of the Polifonia project, I study and propose methods for information retrieval of symbolic music content. We work, in Tiange Zhu's Ph.D., on the development of a search engine [2]. The core of the engine relies on efficient modeling of music content from *n-grams*, which allows integration into standard text search engines, as well as querying various dimensions of music [Zhu+22 ; ZFR22].

My second line of research focuses on graph modeling (possibly dynamic), applied to various problems including recommender systems.

First, I studied Heterogeneous Interaction Networks (HIN), a generalization of graphs. With several colleagues, we have used this model to make an exhaustive classification of diversity metrics, and applied those results to the lists produced by the recommendation systems [Ram+21a ; RTF20]. I have also conducted work on the link streams dynamic graph model, to examine various properties (temporal clique search, local temporal densities) [Gau+16 ; Via+18], and then to study their interest in improving the performance of recommender systems (by efficiently taking into account the temporal dimension of the problem) [VF19a ; VF19b]. Since 2021, I supervise the PhD of Yannis Karmim and we approach this temporal aspect of the recommendation problem and study deep learning techniques, with the models of *Graph Neural Networks*. We have also studied various ways to improve the evaluation of these architectures [Kar+22]. I had previously studied these models on

[2] See the demonstration on http://neuma-dev.huma-num.fr/.

an information retrieval problem to improve the indexing of images in a search engine, using an efficient detection of their captions in web pages [Dan+21].

I also worked on reasoning about *facts*, represented in knowledge graphs, and conflicts between them. With Victor David and Nicolas Travers, we have proposed an original approach to build coherent sets of facts, by maximizing the number and the probabilities of the preserved facts, at the expense of a set of facts left aside [DFT22 ; DFT23]. This work finds its application in historiographic methodology (uncertain facts), but also in multi-agent systems (when those agents must reason in environments with incomplete information).