Système de recommandation musical hybride.

Stagiaire : Arnaud Delpeyroux
Maître de stage : Pierre Hanna
Enseignant référant : Serge Chaumette

03 Septembre 2018

Table des matières

Introduction

Ce mémoire de stage recherche de fin d'étude est réalisé dans le cadre du Master 2 informatique pour l'image et le son de l'Université de Bordeaux. Le stage a été effectué au sein de l'équipe image et son du Labri sous la direction de Monsieur Pierre Hanna, Maître de Conférences à l'Université de Bordeaux.

Le sujet du stage est : "Système de recommandation musicale hybride". Ce travail a été fait en collaboration avec une entreprise de jeux vidéo bordelaise. Ces développeurs ont mis en place un moteur de recommandation sur un de leur jeu vidéo musical. Ce jeu musical propose aux joueurs différents niveaux, chacun étant composé d'une musique et d'une vidéo. Le stage a pour objectif d'étudier l'impact de l'ajout du contenu audio et vidéo dans le système de recommandation existant.

Ce mémoire présentera le contexte du stage. Puis, une présentation de la recommandation sera faite ainsi que deux de ses différentes approches, le filtrage collaboratif et l'approche basée contenu. Un point sera ensuite fait sur le contenu audio et vidéo et sur l'extraction de ceux-ci. Puis on traitera de la malédiction de la dimensionnalité et comment l'éviter. Ensuite, les stratégies utilisées pour ajouter le contenu dans le système seront présentées. Suivra une partie sur les différents protocoles d'évaluation d'un moteur de recommandation. Suite à cela, on présentera les expériences mises en place pour valider nos hypothèses et les résultats obtenus. Enfin, on ouvrira notre travail avec des perspectives d'amélioration.

1 Contexte

Comme dit précédemment, ce stage a pris place au LaBRI du 26 Mars 2018 au 26 Juillet 2018 et plus particulièrement au Scrime sous la direction de Monsieur Pierre Hanna. Le Scrime est un Groupement d'Intérêt Scientifique (`GIS`) et Artistique administré par le LaBRI.

Le Scrime est un rassemblement de chercheurs en informatique et d'artistes travaillant autour de la thématique de la musique et de l'informatique musicale. Le Scrime a pour objectif de favoriser la collaboration entre artistes et scientifiques.

Les projets portés par le Scrime sont :

- `Score` : Il s'agit d'un séquenceur permettant aux régisseurs et créateurs de spectacles vivants de concevoir des performances basées sur le concept du temps souple. Le temps souple est une conception du temps durant lequel les évènements ont des temps de début et de fin non absolus. Le séquenceur Score permet de définir des événements et des enchaînements d'événements à l'aide de conditions logiques et temporelles.
- `SEGMent2` : `SEGMent2` est un moteur de jeux vidéo permettant de créer des `point and click` de manière intuitive à l'aide de diagrammes. L'objectif de ce moteur de jeu est de permettre la création de jeux éducatifs par des non-programmeurs.
- `Projet3Dôme` : Ce projet a pour objectif d'aborder la problématique de la spatialisation 3D du son avec notamment la mise en oeuvre de techniques ambisoniques dans un dispositif de restitution sonore spatialisé (un dôme de haut parleurs).

Le développeur avec lequel nous sommes en collaboration distribue un jeu vidéo musical proposant aux joueurs des niveaux basés sur des musiques accompagnées de vidéos. L'ensemble de ces niveaux peuvent être sélectionnés directement par les joueurs à travers les menus en jeu. Un moteur de recommandation basique a déjà été mis en place dans le jeu pour proposer des morceaux adaptés aux joueurs. Néanmoins, ce moteur existant ne prend en compte que l'utilisation que les joueurs ont des niveaux. En effet, il ne regarde que l'historique d'utilisation des niveaux par les joueurs pour construire les recommandations. Ainsi, ils nous demandent d'évaluer l'impact de l'ajout du contenu audio et vidéo dans le système de recommandation. Pour ce faire, il nous ait fourni les données d'entrainement et d'évaluation du système existant ainsi que les musiques et vidéos proposées aux joueurs. L'objectif idéal étant d'améliorer le système existant en proposant de meilleures recommandations afin de proposer un service personnalisé au joueur. Ceci dans le but d'améliorer la rétention des joueurs par le jeu.

2 Système de recommandation

Un système de recommandation est un système qui a pour objectif de présenter aux utilisateurs d'un service les éléments qui sont susceptibles de les intéresser individuellement. Les systèmes de recommandation fonctionnent à l'aide de différentes données :

- Les objets que l'on cherche à recommander. Ils sont représentés par leurs caractéristiques, c'est ce que l'on appelle leur contenu. Ce contenu dépend grandement des objets considérés.
- Les utilisateurs et leurs profils. Les utilisateurs peuvent être représentés par une liste d'interactions avec plusieurs objets (notes mises à des objets, utilisation ou non d'un objet, ...). Ils peuvent aussi être représentés par leurs caractéristiques propres tel que l'âge, le genre, la nationalité, ... Leur représentation dépend du type de système de recommandation utilisé.
- Les transactions. Les transactions sont les interactions entre les utilisateurs et les objets. Elles prennent souvent la forme d'entrées de journal d'utilisation. Elles peuvent prendre plusieurs formes. Cela peut seulement être un lien entre un objet et un utilisateur, c'est-à-dire l'information binaire que l'utilisateur a interagi avec un objet (un utilisateur lit une nouvelle, regarde un film ou écoute une musique). Ou cela peut prendre la forme d'une appréciation qu'un utilisateur donne à un objet (un utilisateur note un film ou un morceau de musique). Dans ce cas, on parle de données explicites, l'utilisateur donne explicitement son avis sur un objet. Mais les transactions peuvent aussi être implicites, comme dans le cas où un utilisateur écoute une musique et passe à la suivante après quelques secondes. Dans ce cas, la transaction consiste en une information d'utilisation (le temps d'écoute, de lecture, ...). Cette information implicite doit être traitée pour juger si l'utilisateur apprécie ou non l'objet.

Le système de recommandation cherche à modéliser les goûts des utilisateurs pour pouvoir leur proposer les objets les plus adaptés. Afin de modéliser l'intérêt des utilisateurs, plusieurs approches existent :

- Le filtrage collaboratif : L'intérêt de l'utilisateur est modelisé par ses interactions avec les objets mais aussi grâce aux interactions des autres utilisateurs avec les objets. Le principe est de trouver les goûts des utilisateurs avec l'utilisation qu'ils ont du système en trouvant des utilisateurs semblables qui ont des usages similaires.
- L'approche basée contenu : Dans ce cas, on utilise les contenus des objets, les caractéristiques qui définissent un objet, pour comprendre ce qu'aime un utilisateur et quels objets il est susceptible d'aimer. Ainsi, on va recommander aux utilisateurs des objets semblables en terme de contenu à ceux qu'ils aiment déjà.
- L'approche démographique. Le système va chercher à recommander des objets aux utilisateurs, selon leur profil démographique. On définit les utilisateurs par leur âge, leur nationalité, leur genre, leur métier, ... Cependant, ce sont des systèmes non personnels, ils ne proposent pas de recommandations individuelles et personnalisées, mais des recommandations dépendantes de la catégorisation de l'utilisateur.
- L'approche basée communautaire : les recommandations sont faites selon les proches (les amis ou connaissances) des utilisateurs. Ce type de système fonctionne donc grâce aux relations sociales des utilisateurs.

Les moteurs de recommandations sont utilisés dans un grand nombre de domaines et sont de plus en plus populaires. Ils tendent à personnaliser au mieux les services pour améliorer la rétention de ceux-ci. En effet, la principale hypothèse est que si l'on propose aux utilisateurs ce qu'ils souhaitent, alors ils continueront d'utiliser le service. Néanmoins, il est difficile de supposer ce que l'utilisateur veut. D'autant plus que le choix des recommandations a des influences sur l'expérience des utilisateurs. Ainsi, vaut-il mieux recommander des choses nouvelles aux utilisateurs ou des objets "sûrs" pour lesquels nous sommes certains que l'utilisateur les apprécie. Dans le premier cas, on cherche à faire découvrir à l'utilisateur des objets nouveaux avec le risque qu'il les aime moins que les objets que l'utilisateur a l'habitude d'utiliser. Ou bien, on peut conforter les utilisateurs dans leurs goûts pour ne pas prendre de risques.

Un autre problème des systèmes de recommandation est le respect de la vie privée. Dans le cas où seul les informations d'utilisation sont utilisées, le problème ne se pose pas. Néanmoins, certains moteurs de recommandation prennent en compte le contexte d'utilisation (heure de la journée, jour de la semaine) pour améliorer les recommandations. Ce contexte peut être plus ou moins complexe, cela peut être le moment d'utilisation ou l'activité que fait l'utilisateur lors de l'utilisation du système. Par exemple, pour la musique on sait qu'un utilisateur écoute une musique différente lorsqu'il travaille. Or ceci indique que l'on possède de telles informations sur les utilisateurs. Il faut donc faire attention aux données utilisées et à préserver la vie privée des utilisateurs.

3 Filtrage collaboratif

Le filtrage collaboratif se base sur l'environnement social des utilisateurs. L'objectif du filtrage collaboratif est de prédire l'intérêt qu'un utilisateur aura pour un objet à l'aide des informations suivantes :

- L'usage de l'utilisateur : l'ensemble des interactions entre l'utilisateur et les objets existants. Cela peut être des notes que l'utilisateur met aux des objets ou une information binaire d'utilisation (l'utilisateur a écouté telle musique, a joué à tel jeux ou niveau, ...).
- L'usage de tous les autres utilisateurs.

Le principe du filtrage collaboratif réside dans le fait que certains utilisateurs ou certains objets se ressemblent et ont un usage semblable. Cela implique que l'on peut utiliser ces similitudes pour prédire l'intérêt qu'un utilisateur pourrait avoir pour un objet. Le filtrage collaboratif se décompose en plusieurs étapes (voir la figure 1) :

- Construction du modèle : il existe, à cette étape, deux possibilités. Soit le modèle correspond à une matrice contenant l'ensemble des interactions entre les utilisateurs et les objets (approche `memory based`). Cette matrice est donc de taille `nombre d'utilisateurs` × `nombre d'objets`. Soit, un modèle construit à partir de cet historique avec par exemple l'utilisation d'un `SVD` ou autres méthodes de décompositions (approche `model based`).
- Utilisation du modèle pour trouver les plus proches voisins : à cette étape, on cherche à trouver les plus proches voisins pour chaque utilisateur ou pour chaque objet. Il s'agit respectivement de l'approche basée utilisateur et de l'approche basée objet.
- Utiliser les plus proches voisins pour prédire l'intérêt des utilisateurs et d'établir les recommandations.

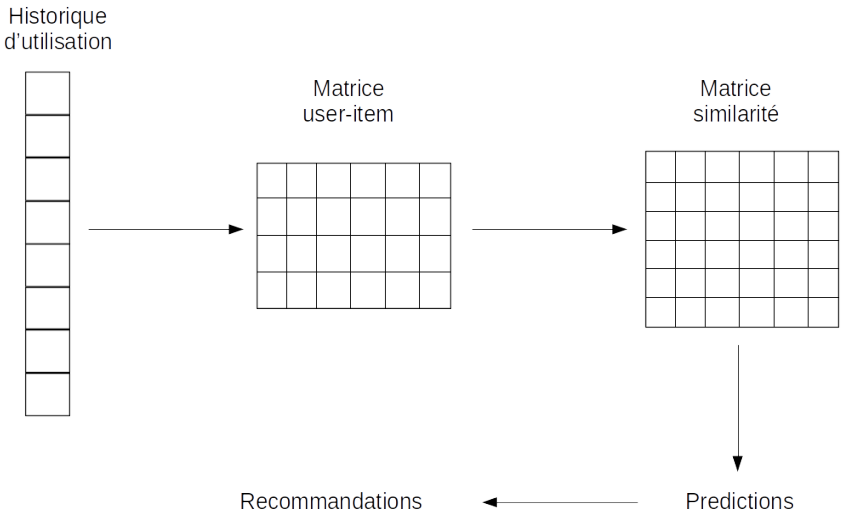


FIG. 1 – *Filtrage collaboratif.*

Dans les parties suivantes on définira toutes les formules dans le cas de l'approche objet. Néanmoins, le principe entre approche objet et approche utilisateur est le même et les formules sont extrêmement similaires.

Durant toute cette partie on utilisera l'exemple suivant 2 pour imager les différentes étapes :

	Matrix	Titanic	Die Hard	Forest Gump	Wall-E
Axel	5	1		2	2
Lucie	1	5	2	5	5
Eric	2		3	5	4
Diane	4	3	5	3	

FIG. 2 – *Exemple d'utilisateurs ayant noté des films de 1 à 5.*

Cet exemple représente quatre utilisateurs (Axel, Lucie, Eric et Diane) et cinq objets (ici les films Matrix, Titanic, Die Hard, Forest Gump et Wall-E). Certains utilisateurs ont vu certains films et leur ont attribué des notes entre un et cinq. Ces notes sont donc les transactions entre les utilisateurs et les objets. La matrice ci-dessus n'est pas pleine, car tous les utilisateurs n'ont pas vu tous les films.

3.1 Memory based

L'approche basée mémoire consiste à construire le modèle en utilisant une matrice contenant l'ensemble des interactions entre les utilisateurs et les objets. Comme dit plus tôt, cette matrice est de taille `nombre d'utilisateurs` × `nombre d'objets`. Cette matrice est la matrice `user-item` R 3 où r_{ij} correspond à l'interaction entre l'utilisateur i et l'objet j. Cette interaction peut être une note donnée par l'utilisateur à l'objet ou peut être une information binaire d'utilisation ou bien même une valeur calculée à l'aide d'informations implicites.

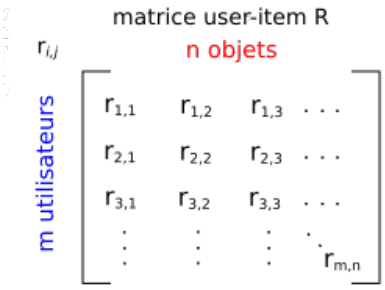


FIG. 3 – *Matrice user-item.*

Cette matrice sert donc de point d'entrée au système de recommandation. Selon les cas, cette matrice peut être creuse. En effet, dans un système réel, il se peut que beaucoup d'utilisateurs n'aient pas interagi avec beaucoup d'objets. Ceci conduit à une matrice qui peut être creuse.

Grâce à la matrice `user-item`, il est possible de définir les objets à l'aide de vecteurs d'utilisation. Ainsi, un objet est défini par une liste de valeurs qui représente si l'objet a interagi avec tel ou tel utilisateur (approche basée objet). Dans le cas inverse, on peut représenter les utilisateurs par des vecteurs d'utilisation (approche basée utilisateur). Ce seront donc ces représentations qui seront utilisées pour calculer les similarités.

La figure 2 correspond à la matrice `user-item` pour notre exemple. C'est une matrice creuse contenant toutes les transactions.

3.2 Model based

Dans le cas de l'approche basée modèle, on essaye de construire un modèle pour chaque utilisateur et chaque objet. Ces modèles seront utilisés pour prédire l'intérêt d'un utilisateur pour un objet. En effet, il suffit de confronter les deux modèles ensemble pour voir s'ils correspondent. La méthode courante pour construire de tels modèles est l'utilisation d'une décomposition en valeurs singulières (ou `SVD`, de l'anglais : `Singular Values Decomposition`). La `SVD` est une technique de factorisation de matrice. Le principe de cette approche est de décomposer la matrice `user-item` R en trois matrices U, V et Σ tel que :

$$R = U * \Sigma * V^T$$

Dans notre cas, cette méthode ne peut pas être utilisée directement. En effet, la `SVD` sur la matrice `user-item` n'est pas définie car cette matrice (ici R) est creuse, beaucoup de valeurs ne sont pas définies. Ceci vient du fait que pour construire les matrices U et V il faut pouvoir calculer les vecteurs propres de $R^T R$, or $R^T R$ n'existe pas, car R est creuse. Néanmoins, il reste un espoir. Il est possible de construire les matrices U et V en résolvant un problème d'optimisation. On peut trouver les matrices U et V si on l'on trouve les vecteurs p_u et q_i (les vecteurs p_u sont les lignes de U et q_i les colonnes de V^T) tel que :

- $r_{ui} = p_u * q_i$
- Les vecteurs p_u et q_i sont mutellement orthogonaux.

La seconde contrainte peut être ignorée et l'on peut reconstruire U et V avec une matrice incomplète R, comme montré par Simon Funk [1]. Ainsi il faut résoudre le problème suivant :

$$min_{p_u, q_i} \sum_{r_{ui} \in R} (r_{ui} - p_u * q_i)^2$$

Enfin, ce problème est résolu en utilisant une descente de gradient stochastique (`SGD`).

L'avantage de la `SVD` est qu'elle permet de représenter indépendemment les utilisateurs et les objets en réduisant le nombre de dimensions du problème. Aussi, les objets et les utilisateurs sont représentés par leur usage. Enfin, il est facile à l'aide des matrices U et V de trouver une transaction non existante entre un objet et un utilisateur donné.

3.3 Calcul de la similarité

Durant cette partie, nous ne détaillerons que les similarités entre objets (les films dans notre exemple). Néanmoins, le principe est similaire entre utilisateurs et les formules varient peu.

La seconde étape du filtrage collaboratif est de calculer les similarités entre tous les objets dans l'objectif de trouver les plus proches voisins pour chaque objet. Les similarités ainsi calculées prendront en compte l'usage et non l'objet en lui-même. Ainsi, si deux objets similaires (deux films du même auteur et du même genre, deux morceaux de musiques du même artiste, ...) ont été complètement utilisés par des utilisateurs différents, alors ils seront considérés comme très différents malgré qu'ils soient semblables en terme de contenu.

On détaillera ici deux fonctions de similarité, la similarité cosinus et la similarité cosinus centré.

Similarité cosinus L'objectif de la similarité cosinus est de calculer la similarité entre deux vecteurs de même dimension. Le principe est d'au lieu d'utiliser une distance d'ordre N, on va utiliser la valeur de l'angle entre les deux vecteurs. Ainsi deux vecteurs colinéaires seront similaires alors que deux vecteurs avec une direction inverse seront non similaires. Ainsi cette similarité est définie telle que pour deux vecteurs x_a et x_b :

$$cos(x_a,x_b) = \frac{x_a^T x_b}{\|x_a\|\|x_b\|}$$

Si l'on considère deux objets i et j on définira la similarité cosinus entre ces deux objets telle que :

$$CV(i,j) = cos(x_i,x_j) = \frac{\sum_{i\in U_{ij}} r_{ui} r_{uj}}{\sqrt{\sum_{i\in U_{ij}} r_{ui}^2 \sum_{j\in U_{ij}} r_{uj}^2}}$$

Ou U_{ij} correspond à l'ensemble des utilisateurs ayant interagi avec les objets i et j. Néanmoins, on peut choisir de considérer l'ensemble des utilisateurs et remplir le reste de la matrice avec des zéros. Dans le cas où les interactions ne peuvent être zéro alors cela n'influe pas les calculs et simplifie la méthode.

Si l'on calcule la similarité cosinus entre tous les objets de l'exemple on obtient la matrice de la figure 4.

	Matrix	Titanic	Die Hard	Forest Gump	Wall-E
Matrix	1	0.55	0.67	0.69	0.51
Titanic	0.55	1	0.69	0.77	0.68
DieHard	0.67	0.69	1	0.82	0.53
Forest Gump	0.69	0.77	0.82	1	0.92
Wall-E	0.51	0.68	0.53	0.92	1

FIG. 4 – *Similarités cosinus.*

Similarité cosinus centré La similarité cosinus centré est très semblable à la similarité cosinus. Néanmoins, dans le cas du calcul de la similarité entre objets, on compare les notes données par des utilisateurs différents. Or, différents utilisateurs n'ont pas les mêmes échelles de valeur et donc des mêmes notes mise par des utilisateurs différents n'ont pas le même sens. Afin de prendre ceci en compte, il faut soustraire aux notes lors du calcul la moyenne des notes données par les utilisateurs. On obtient ainsi la définition suivante :

$$AC(i,j) = \frac{\sum_{i\in U_{ij}} (r_{ui} - \overline{r}_u)(r_{uj} - \overline{r}_u)}{\sqrt{\sum_{i\in U_{ij}} (r_{ui} - \overline{r}_u)^2 \sum_{j\in U_{ij}} (r_{uj} - \overline{r}_u)^2}}$$

Si l'on calcule la similarité cosinus centré entre tous les objets de l'exemple, on obtient la matrice de la figure 4.

	Matrix	Titanic	Die Hard	Forest Gump	Wall-E
Matrix	1	-0.32	0	-0.50	-0.52
Titanic	-0.32	1	-0.75	-0.49	-0.12
DieHard	0	-0.75	1	-0.21	-0.70
Forest Gump	-0.50	-0.49	-0.21	1	0.50
Wall-E	-0.52	-0.12	-0.70	0.50	1

FIG. 5 – *Similarités cosinus centrés.*

3.4 Prédictions et recommandations

À l'aide des similarités obtenues, on peut pour chaque objet trouver ses plus proches voisins. Ces plus proches voisins seront utiles pour prédire l'intérêt d'un objet pour un utilisateur.

3.4.1 Prédiction basée objet

Afin de prédire l'intérêt qu'un utilisateur va avoir pour un objet, nous allons utiliser les notes mises par l'utilisateur aux plus proches voisins notés de l'objet. Le principe étant de déduire d'objets similaires l'intérêt de l'utilisateur. En effet, si l'on sait que l'utilisateur porte un grand intérêt à des objets similaires à l'objet considéré, alors la probabilité qu'il soit intéressé par l'objet est grande. Dans le cas contraire, s'il ne porte pas d'intérêt pour les objets voisins, alors son intérêt pour l'objet sera faible. La formule suivante permet de prédire la valeur d'une transaction de cette manière :

$$\hat{r}_{ui} = \frac{\sum_{j \in N_u(i)} w_{ij} r_{uj}}{\sum_{j \in N_u(i)} |w_{ij}|}$$

Avec w_{ij} la similarité entre les objets i et j.

De plus, on peut voir que les transactions prises en compte dans la formule sont pondérées par la similitude entre l'objet i et les objets j correspondants. Cela permet de donner plus de poids aux objets les plus proches.

4 Recommandations basées contenu

Contrairement au filtrage collaboratif, l'approche basée contenu n'utilise pas l'environnement social de l'utilisateur pour lui construire des recommandations. Le principe du basé contenu est d'utiliser le contenu des objets à recommander et l'usage de l'utilisateur pour connaitre quels sont les objets qui l'intéressent. L'objectif, afin de construire les recommandations du basé contenu, est d'utiliser l'historique d'utilisation de l'utilisateur et la représentation du contenu des objets pour calculer un modèle utilisateur. Ce modèle correspond à l'intérêt de l'utilisateur pour les différentes caractéristiques des objets. Ainsi, à l'aide de ce modèle, il est aisé de savoir si un objet plaira ou non à un utilisateur.

On appelle contenu toute information qui définie l'objet. Le contenu dépend nécessairement de la nature de l'objet. Par exemple, dans le cas d'un film, on pourra définir le contenu comme le genre, le réalisateur, les acteurs jouant dans le film ou l'année de production, mais aussi des informations que l'on peut extraire depuis le flux vidéo. Dans tous les cas, ces informations représentent le film et décrivent ce qu'il est et ce qu'il contient. Dans le cas des films, des vidéos ou de la musique, l'utilisation du contenu a toujours été problématique. En effet, extraire des informations de films, musiques ou vidéos prend beaucoup de temps. De plus, il peut être difficile de choisir quoi extraire. Il faut en effet que les informations représentent bien l'objet et qu'elles soient discriminantes. Si les informations ne représentent pas bien les objets, alors on ne pourra pas avoir confiance dans les recommandations et si elles ne sont pas discriminantes, alors on ne pourra pas différencier deux objets objectivement différents.

Un système de recommandation basé contenu est constitué des blocs suivants :

- Analyse du contenu. Il faut en effet un bloc capable d'extraire ou de traiter le contenu d'un objet. Ce contenu peut être annoté à l'objet ou extrait depuis l'objet lui-même.
- Apprentissage du profil utilisateur. Il faut pouvoir, à l'aide de l'historique d'utilisation de l'utilisateur et du contenu des objets utilisés, construire une représentation des goûts de l'utilisateur. Habituellement, le profil est construit à l'aide de techniques d'apprentissage machine.
- Filtrer les objets. Ainsi, à l'aide du profil utilisateur il faut filtrer les objets en comparant la représentation des objets et le profil de l'utilisateur. Le résultat est une information binaire ou continue obtenue à l'aide de fonctions de similarité diverses.

L'approche basée contenu a la possibilité de pouvoir recommander des objets nouvellement ajoutés contrairement au filtrage collaboratif. En effet, même sans usage concernant un objet, il est possible d'utiliser son contenu pour le comparer aux profils utilisateurs. Cependant, dans le cas d'un nouvel utilisateur, l'approche basée contenu ne peut rien prédire. En effet, sans utilisation un utilisateur ne peut construire son profil et donc il n'existe pas pour le système. Le même problème se pose pour le filtrage collaboratif.

Le problème du `cold start` utilisateur n'est pas le seul défaut de l'approche basée contenu. En effet, cette approche a tendance à être sur-spécialisée. Elle va finir par toujours recommander le même type d'objet aux mêmes utilisateurs. Chaque utilisateur ayant un profil bien à lui, seuls les objets correspondant bien au profil et donc tous similaires lui seront recommandés. Aussi, cette approche est très dépendante des informations de contenu utilisées et il peut être difficile de décrire le contenu des objets. D'autant plus que l'extraction de ce contenu peut être difficile et longue.

5 Contenu audio et vidéo

5.1 Extraction des caractéristiques

5.1.1 Audio

Il est possible d'obtenir beaucoup d'informations sonores et musicales depuis un signal musical. Ces informations peuvent caractériser la musique à différents niveaux. Il existe des informations qui caractérisent le signal à très bas niveaux, puis des informations plus haut niveau. Ces informations haut niveau ont plus de sens pour nous. Cela peut être pour de la musique, le tempo, la tonalité, l'intensité de la musique ou d'autres caractéristiques musicales. Néanmoins, ces informations haut niveaux reposent souvent sur des informations extraites depuis le signal audio. En effet, les informations bas niveau, bien que peu compréhensibles, contiennent beaucoup d'informations. L'ensemble de ses informations que l'on peut extraire vont caractériser et représenter une musique.

Afin d'extraire ces caractéristiques, on utilise la bibliothèque libre `Essentia`[1]. Cette bibliothèque développée par l'équipe `Music Technology Group`[2] de l'université `Pompeu Fabra` à Barcelone permet d'analyser et d'extraire des caractéristiques depuis des signaux audios et/ou musicaux. Pour extraire plus de 4300 valeurs, on utilise l'utilitaire `essentia_streaming_extractor_music`. Cet utilitaire permet d'extraire l'ensemble des caractéristiques disponibles dans `Essentia` de manière rapide et automatisé.

Caracteristiques de bas niveau: Les caracteristiques bas niveaux sont des informations extraites directement depuis le signal audio. Comme dit précédemment, ces caractéristiques contiennent beaucoup d'informations et permettent de décrire le signal. Elles représentent par exemple le contenu du signal, sa forme et son comportement au cours du temps. Souvent les caracteristiques bas niveau sont aussi spectrales. Elles sont extraites depuis la représentation spectrale du signal audio. La représentation spectrale d'un signal audio est obtenue à l'aide de la transformé de Fourier rapide (FFT) appliquée à ce signal. Néanmoins, dans le cas d'un signal audio, il est rare d'appliquer une FFT à tout un signal. En effet, on considère souvent de petites unités de temps pour des raisons de performances, mais aussi pour garantir une certaine précision temporelle du spectre obtenu. Dans ce cas, on parle de transformé de Fourier à court terme (STFT).

Caracteristiques rythmiques et tonales: Le rythme et la tonalité sont des caractéristiques musicales. Elles n'existent que parce que l'on étudie des morceaux de musiques. Ces informations permettent de décrire de manière intelligible un signal musical et agrègent un grand nombre d'informations. Ce sont donc des informations de plus haut niveau. Ces caractéristiques sont souvent calculées à l'aide d'autres caracteristiques bas niveau.

Voici plusieurs exemples de caractéristiques extraites depuis le signal audio :

Les `Mel-frequency cepstral coefficients` (MFCCs): Les MFCCs sont des caractéristiques bas niveau extraites depuis le signal. Ce sont aussi des caractéristiques spectrales. C'est-à-dire qu'elles sont extraites depuis la représentation spectrale du signal. Il faut en effet effectuer une transformé de Fourier sur le signal pour extraire les MFCCs. Les MFCCs sont les

1. http://essentia.upf.edu/documentation/
2. https://www.upf.edu/web/mtg/contact

coefficients qui collectivement définissent le `mel-frequency cepstrum` (MFC). Le MFC est une représentation du spectre d'amplitude à court terme d'un signal. Cette représentation utilise des échelles logarithmiques basées sur le système psychoacoustique humain (échelle Mel). Ceci permet aux MFCCs de décrire le spectre dans une échelle qui a du sens pour la perception humaine. Le plus souvent [13] les MFCCs sont extraits de la manière suivante (voir figure 6). D'abord il faut découper le signal en trames auxquelles on applique une fonction de fenêtrage. Cette fonction modifie le signal pour accentuer ou diminuer certaines propriétés du spectre. Puis, pour chaque trame on applique une FFT sur le signal d'entrée pour obtenir le spectre d'amplitude. Ensuite, on filtre le spectre par un banc de filtres triangulaires espacés selon l'échelle Mel. Enfin, on effectue une transformé cosinus discrète du spectre pondéré pour obtenir les coefficients.

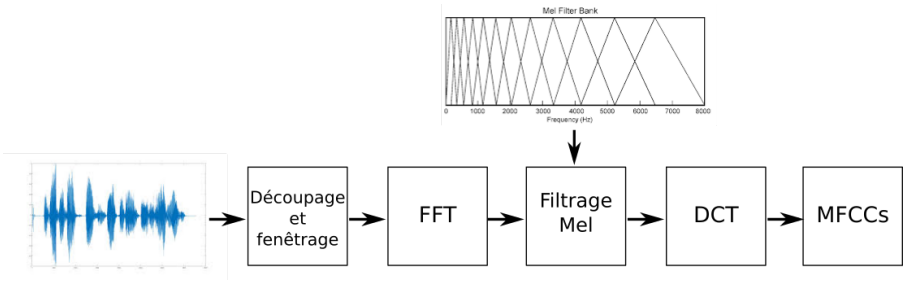


FIG. 6 – *Calcul des MFCCs.*

Les `Harmonic pitch class profiles` (HPCP): Les HPCP sont des caractéristiques tonales extraites depuis le signal audio. Ces caractéristiques permettent d'extraire la distribution des 12 tonalités de la gamme tempérée au cours du temps. Ces caractéristiques permettent donc de connaitre la tonalité du morceau et les classes de notes utilisées au cours de la musique. Il s'agit donc d'une information musicale. De plus, cette information est assez haut niveau et abstraite. Elle nous permet de mieux comprendre le signal en tant que morceau de musique. Ces caractéristiques sont calculées de la manière suivante [8] (voir figure 7). Tout d'abord, de la même manière qu'avec les MFCCs, il faut découper le signal en trame de petites durées. Puis pour chaque trame, on applique la transformé de Fourier pour obtenir le spectre d'amplitude qui sera filtré pour ne garder que les valeurs entre 100 et 5000 Hz. Dans ce spectre d'amplitude on cherche à détecter les pics (les maximas locaux). Ces pics correspondent aux fréquences les pus présentes dans le signal. Dans le cas d'un son, cela correspond aux différentes hauteurs ou notes perçues dans le signal. Ensuite, on calcule la fréquence de chacun de ces pics par rapport à une fréquence de référence (440Hz). Ces fréquences relatives nous permettent de faire correspondre les fréquences trouvées dans le signal avec les tonalités correspondantes. Enfin, pour que les valeurs obtenues ne dependent pas des valeurs d'amplitude du morceaux (s'il est fort ou faible), il faut normaliser les valeurs obtenues.

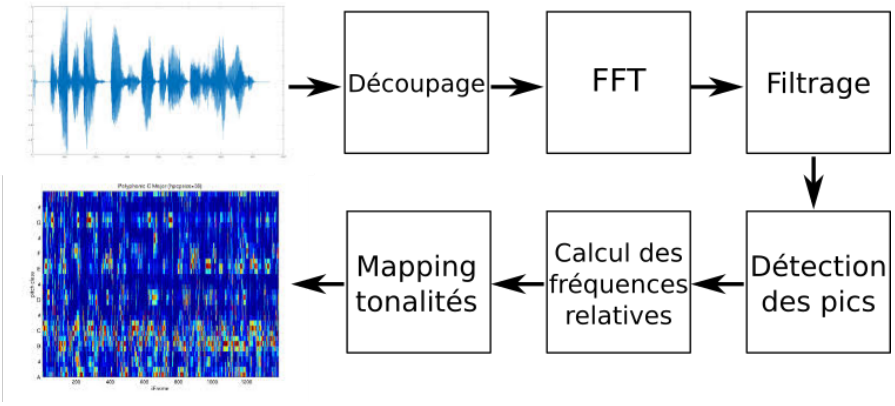


FIG. 7 – *Calcul des HPCP.*

La dansabilité: Comme son nom l'indique, cette caractéristique mesure si la musique est dansante ou non. Une grande valeur indique si la musique est très dansante et une valeur de 0 indique que la musique n'est pas dansante. C'est donc une caractéristique musicale de très haut niveau. Cette mesure peut être calculée en utilisant une variation de l'algorithme calculant la `Detrended Fluctuation Analysis` (DFA) [12].

5.1.2 Vidéo

Depuis la vidéo, on extrait plus de 540 caractéristiques. Ces valeurs sont extraites à l'aide d'Opencv footnoteOpenCv est une bibliothèque libre de traitement d'image et de flux vidéo depuis le flux vidéo. Comme pour l'audio, on extrait des informations de différents niveaux.

On extrait différents types d'informations de la vidéo. Tout d'abord, des informations de mouvement, puis des informations colorimétriques et enfin des informations de luminosité des images. Certaines de ces caractéristiques sont proposées dans l'article [5], où elles sont utilisées pour faire de la recommandation basée contenu de vidéos.

Dans notre cas l'ensemble de nos vidéos sont en très haute définition en 60 images par secondes. Afin d'accélérer le calcul des caractéristiques vidéo, on ne considère qu'une image sur cinq et on divise la taille des images par quatre en largeur et en longueur.

Mouvement: Le mouvement dans une vidéo correspond aux mouvements perçus des objets filmés ou animés. Ceci est représenté par le "mouvement des pixels" d'une image à l'autre de la vidéo. Ainsi afin de mesurer le mouvement, on cherche à trouver ce qu'on appelle les `motion vectors`. Ces `motion vectors` correspondent aux déplacements des pixels d'une image à l'autre. Néanmoins, il existe différents mouvements pouvant être mesuré. En effet, on peut dissocier deux mouvements principaux, le mouvement des objets animés ou filmés (acteurs, véhicules, animaux, ...) et les mouvements du point de vue (déplacement de la caméra, textttttraveling, dots). Dans notre cas, les vidéos sont considérées comme fixes, alors le seul mouvement présent sera celui des objets de la vidéo. Il ne sera donc pas nécessaire d'extraire le mouvement du point de vue. Ainsi pour mesurer ce mouvement, on utilise la méthode de Gunnar Farneback [6]. Cette méthode cherche à calculer les `motion vectors` en comparant deux images consécutives de la vidéo. À l'aide de ces `motion vectors`, on peut connaitre la direction du mouvement et son amplitude. Ceci nous aide à calculer la quantité de mouvement en considérant la moyenne de l'amplitude de l'ensemble de tous les `motion vectors` de toute la vidéo. Aussi, on construit

un histogramme des directions des `motion vectors` pour toute la vidéo. La première valeurs nous indique si la vidéo bouge beaucoup ou non. La seconde nous donne une distribution de la direction des mouvements.

Couleurs: Concernant les couleurs présentes dans la vidéo, on extrait deux informations différentes. La première est l'ensemble des teintes présentes dans la vidéo. On calcule donc un histogramme des couleurs présentes dans toute la vidéo. Pour ce faire, on change d'espace de couleur pour obtenir du `HSV` (dans cet espace de couleur, les images ne sont plus codées avec une combinaison de bleu, rouge et vert, mais à l'aide d'une composante de teinte -H-, une de saturation -S- et une d'intensité -V-) pour chaque image. Ensuite, il suffit d'utiliser le canal de teinte H pour remplir l'histogramme que l'on normalisera. On calcule aussi la variance moyenne de la couleur au sein de la vidéo. Pour ce faire, il faut pour chaque image de la vidéo représentée en `Luv` puis on calcule la matrice de covariance de l'image. Grâce au déterminant de cette matrice, on peut obtenir la variance de couleur au sein d'une image. Enfin, on moyenne ces variances obtenues pour toute la vidéo.

Luminosité: Concernant la lumière, on cherche à mesurer la quantité de lumière dans la vidéo. Dans le cas où la vidéo est lumineuse, cette valeur sera grande. Inversement, si la vidéo est sombre, alors la valeur de la mesure sera faible. Pour obtenir ce comportement, on mesure la `lightning key` tel que proposé dans l'article [5]. Cette valeur est obtenue en utilisant pour chaque image de la vidéo sa version en nuances de gris. En effet, l'information contenue dans une image en nuances de gris est l'information de luminosité ou d'intensité de l'image. Ainsi à l'aide de ces images, on peut calculer la moyenne et l'écart type des nuances de gris. En moyennant le produit de ces deux valeurs (moyenne et écart type) sur l'ensemble de la vidéo, on obtient la `lightning key` de la vidéo.

5.2 Mesure de la similarité

On peut utiliser ces représentations des objets pour calculer les similarités entre les objets. Il est en effet facilement possible d'appliquer les similitudes cosinus ou cosinus centré sur les représentations audio ou vidéo des objets. Néanmoins, étant donné le nombre de caractéristiques, on peut rencontrer un problème concernant les similitudes obtenues. Que ce soit dans le cas de l'audio ou de la vidéo, le nombre de caractéristiques extraites est très grand et donc les objets sont représentés dans un espace de très grande dimensionnalité. Or, calculer des distances ou des similarités entre des objets dans de tels espaces pose un problème de dimensionalité, le fléau de la dimensionnalité.

6 Fléau de la dimension

Quand on traite des objets avec un nombre de propriétés très grand on peut rencontrer un problème connu sous le nom de fléau de la dimensionnalité (`curse of the dimensionality`). Ce problème peut se présenter sous diverses formes. Dans notre cas, quand on cherche à calculer des distances entre objets dans un espace de grande dimensionnalité, on fini par avoir tous les objets équidistants les un des autres [2]. La seule solution à ce problème est de diminuer le nombre de dimensions. Ainsi, on a exploré trois techniques différentes pour réduire le nombre de dimensions de notre problème. Nous avons tout d'abord mis en place une sélection des caractéristiques. En effet, on a d'abord cherché à trouver quelles caractéristiques audio ou vidéo avaient le plus de sens pour ne prendre en compte que celles-ci. Puis dans un second temps, nous avons cherché à réduire le nombre de dimensions en compressant les données en utilisant un autoencoder et une analyse en composante principale.

6.1 Sélection des caractéristiques

Afin de sélectionner les caractéristiques les plus intéressantes, il nous faut définir ce qui rend une caractéristique pertinente ou non. Dans notre cas, il faut trouver un lien entre le contenu audio et vidéo et l'utilisation que les joueurs font des morceaux. C'est pour cela que nous nous sommes intéressés à la popularité des morceaux. Grâce à l'historique d'utilisation, on peut compter combien de fois chaque morceaux apparaît et donc leur attribuer une classe de popularité. Ainsi, à l'aide de ces classes de popularités et d'un arbre de décision, on peut trouver quelles caractéristiques sont les plus pertinentes.

Un arbre de décision est un outil classiquement utilisé pour résoudre des problèmes de classification [11]. C'est un algorithme d'apprentissage supervisé qui permet, à partir de données étiquetés en entrée, de construire un modèle et à l'aide de ce modèle de prédire l'étiquette d'un objet [3]. Cet algorithme permet aussi nativement d'obtenir l'importance de chaque caractéristique dans la classification. Ainsi, grâce à un arbre de décision, il est facile d'obtenir une liste des caractéristiques les plus pertinentes pour résoudre un problème de classification donné.

Ainsi en utilisant le contenu audio et/ou vidéo pour définir nos morceaux, on peut résoudre le problème de classification de la popularité pour sélectionner les caractéristiques. Néanmoins, ce problème de classification pose problème. Tout d'abord, l'ensemble des classes sont par définition déséquilibrées, elles ne comportent pas toutes le même nombre d'objets. En effet, il y a peu de morceaux très populaires et énormément de pas populaires.

Afin de construire les différentes classes de popularité, on a compté pour chaques morceaux le nombre de fois qu'ils ont été joués et nous les avons rangés dans ordre décroissant. On a donc obtenu la liste des morceaux dans l'ordre du plus joué au moins joué. Avec ceci, on a construit cinq classes de popularité. On a choisi de construire chaque classe de façon à ce que toutes les classes contiennent le même nombre de parties jouées. Il faut donc que la somme des parties jouées par l'ensemble des morceaux contenus dans chaque classe soit égale à toutes les autres sommes. Comme dit plus haut, la distribution des morceaux dans les classes est donc déséquilibrée. La classe contenant les morceaux les plus populaires n'ont pas besoin de beaucoup de morceaux pour que le nombre de parties jouées soit le même que la classe contenant les morceaux les moins populaires.

Pour essayer de compenser ce problème, on applique une pondération des objets lors de l'apprentissage de la classification. On cherche donc à favoriser les éléments les plus populaires

car moins nombreux et à défavoriser les moins populaires car beaucoup plus nombreux.

6.2 Réduction des dimensions

La réduction des dimensions consiste, au lieu de choisir quelles caractéristiques garder, comment les transformer pour passer vers un espace de plus petite dimension. Ainsi, on a mis en oeuvre deux approches différentes.

6.2.1 PCA

L'analyse en composante principale (`ACP` ou `PCA` en anglais) est un traitement statistique qui permet de transformer un ensemble de variables possiblement corrélées en un ensemble de dimensionnalité inférieure de variables décorrélées. Ainsi, l'objectif de l'ACP est de trouver une rotation de l'espace dans lequel est défini l'ensemble des objets représentés par les caractéristiques. En effet, chaque objet est représenté par une liste de caractéristiques et donc peuvent être définis dans un espace de grande dimensionnalité (autant de dimensions que de caractéristiques) par un nuage de points. Afin de trouver la meilleure transformation de l'espace contenant ce nuage de points, l'ACP cherche les directions dans l'espace qui maximise la variance de la distance entre la projection des objets sur cette direction et le centre de l'espace. Ainsi, l'algorithme cherche toutes les directions orthogonales qui maximisent les distances du centre de l'espace aux objets. On peut construire un nouvel ensemble de caractéristiques décorrélées. De plus, il peut exister des directions dont la variance peut être minime ou nulle, ces directions peuvent être ignorées. Ceci conduit à une diminution du nombre de dimensions de la nouvelle représentation des objets. Enfin, les caractéristiques sont projetées sur les directions restantes pour obtenir la réduction souhaitée.

6.2.2 Autoencoder

L'autoencoder est une méthode d'apprentissage profond permettant d'encoder un signal donné. C'est un réseau de neurones profond qui, à partir d'un signal, essaye d'apprendre une compression et une décompression qui vont minimiser la perte d'information entre le signal en entrée et celui en sortie. Son objectif est donc d'apprendre des caractéristiques discriminantes. Un autoencodeur est constitué de deux parties. La première est l'encodeur. Cette partie vient compresser le signal en entrée pour réduire le nombre de dimensions sur lequel il est défini. La seconde partie est le décodeur. Le décodeur prend en entrée la sortie de l'encodeur, c'est-à-dire le signal encodé et cherche à reconstruire le signal d'entrée à partir du signal encodé. L'entraînement de l'autoencoedur permet de manière non supervisé d'entraîner à la fois l'encodeur et le décodeur. Le point intéressant avec un autoencoder est que l'on peut l'entraîner entièrement avec un ensemble de données et utiliser seulement l'encodeur pour réduire en dimension un signal en entrée.

Nous avons donc mis en place un autoencoder le plus simplement possible. L'objectif de notre encodeur est donc de prendre en entrée la représentation d'un objet à recommander et de reconstruire une représentation de plus basse dimension de cette entrée. On a donc choisi une architecture simple composée de cinq couches cachées entièrement connectées (voir figure 8). Nous avons choisi une telle architecture pour aller au plus simple. Aussi, ce type d'architecture semble souvent utilisé pour les autoencodeurs [9].

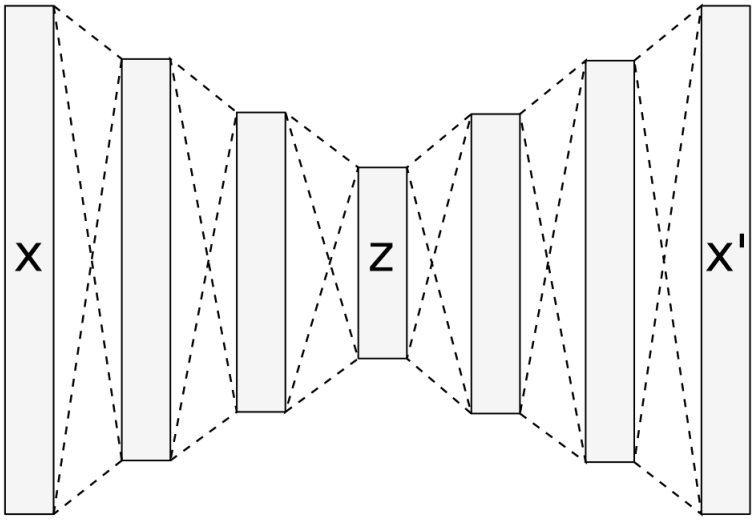


FIG. 8 – *Architecture de l'autoencoder.*

7 Stratégies d'hybridation

Après avoir vu comment construire des recommandations avec le filtrage collaboratif et comment extraire le contenu, il faut ajouter le contenu au système de recommandation existant. Pour des raisons de simplicité, on a choisi d'injecter le contenu directement dans la méthode de filtrage collaboratif. On a choisi de l'injecter au niveau du calcul de la similarité entre les objets (voir figure 9). Ainsi, on a calculé la similarité suivant différentes méthodes qui mélangent le contenu et l'usage.

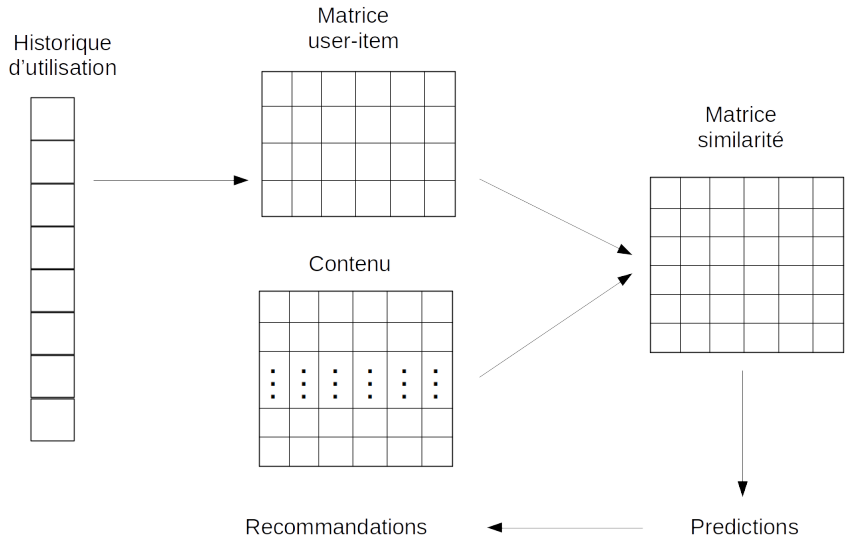


FIG. 9 – *Ajout du contenu dans la recommandation.*

Afin d'hybrider le système, nous avons testé plusieurs approches. La première est de calculer la similarité entre les objets à l'aide de toutes les représentations en les empilant les une sur les autres. Les autres consistent à calculer les similarités indépendamment pour chaque représentation et combiner ces similarités suivant différentes techniques.

7.1 Empilement des représentations

La première approche consiste donc à empiler les différentes représentations pour définir les objets. Ainsi, chaque objet est représenté par son usage suivi du contenu audio et enfin du contenu vidéo. On obtient donc un très long vecteur qui sera utilisé pour calculer les similarités (voir figure 10). L'avantage de cette méthode est la simplicité. Néanmoins, selon la taille des différentes représentations, certaines représentations peuvent prendre plus d'importance dans le calcul de la similarité.

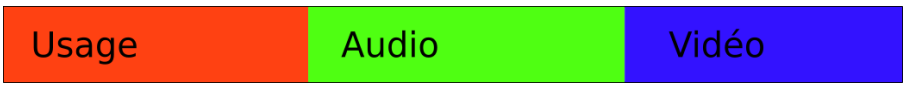


FIG. 10 – *Empilement des representations.*

7.2 Combinaison des similarités

La seconde approche est de calculer plusieurs similarités à l'aide de chaque représentation puis de combiner ces similarités à l'aide de différentes fonctions de mélanges. On a mis en place trois fonctions de mélange simples, un mélange pondéré, un minimum et un maximum. Dans la suite de ce chapitre, on notera S la similarité finale, S_u la similarité de l'usage, S_a la similarité du contenu audio et S_v la similarité du contenu vidéo.

7.2.1 Mélange

La première approche est d'effectuer une pondération de chaque similarité. Cette pondération permet de choisir l'importance de chaque représentation dans la similarité finale. Ceci a l'avantage de proposer de favoriser le contenu à l'usage et dans le contenu, favoriser l'audio ou la vidéo. La pondération s'effectue à l'aide la formule suivante :

$$S = \alpha_1 * S_u + (1 - \alpha_1) * \alpha_2 * S_a + (1 - \alpha_1) * (1 - \alpha_2) * S_v$$

Avec :

– α_1: le coefficient de pondération entre l'usage et le contenu.

– α_2: le coefficient de pondération entre le contenu audio et le contenu vidéo.

Ces deux coefficients doivent être respectivement compris entre 0 et 1. Le coefficient α_1 permet de pondérer l'utilisation du contenu et de l'usage. Ainsi, si $\alpha_1 = 0$ alors la similarité ne prendra en compte que le contenu et si $\alpha_1 = 1$ seul l'usage sera utilisé. Concernant α_2, il permet de choisir dans le contenu si l'on prend en compte plus l'audio ou la vidéo. Ainsi, si $\alpha_2 = 0$, alors le contenu vidéo sera seulement pris en compte alors que si $\alpha_2 = 1$, seul l'audio est pris en compte.

La difficulté d'utiliser cette méthode est de paramétrer la méthode. En effet, évaluer l'impact d'une pondération donnée à priori est difficile, voir impossible. Ainsi, il est nécessaire de tester un certain nombre de combinaisons et d'évaluer les résultats pour paramétrer au mieux la pondération.

7.2.2 Minimum

Une autre approche est de prendre les similarités minimum entre toutes les représentations (voir figure 11). L'intuition derrière cette méthode est que si l'on prend la similarité minimale pour tout couple d'objets, alors on peut avoir confiance dans la mesure de la similarité. En effet, si la valeur est haute alors cela veut dire que toutes les représentations sont très similaires entre les deux objets. Ainsi, on peut être confiant de considérer que les objets sont similaires. Dans le cas où la similarité est faible, cela veut dire que au moins une représentation des deux objets est faible. On considère ainsi que les objets sont différents. Cette vision peut conduire à une valeur moyenne des similarités très faibles. Aussi, cette méthode est très dépendante des représentations et ne propose aucun contrôle sur la similarité finale obtenue.

FIG. 11 – *Minimum des similarités.*

7.2.3 Maximum

Enfin notre dernière approche est de prendre la similarité maximale (voir figure 12). Contrairement à la similarité minimale, la similarité maximale fait confiance à la représentation donnant deux objets similaires. Ainsi, la seule solution pour avoir une similarité faible est que toutes les représentations sont très distantes pour deux objets. En effet, si une représentation donne une grande similarité, alors elle sera sélectionnée malgré que les autres représentations n'obtiennent pas de bonnes similarités. Contrairement à la similarité minimale, on risque dans ce cas d'obtenir une similarité moyenne haute. Enfin, tout comme le minimum, le maximum n'offre aucun contrôle sur le calcul de la similarité et est très dépendant des représentations.

FIG. 12 – *Maximum des similarités.*

8 Évaluation

8.1 Stratégies d'évaluation

Comme tout problème d'apprentissage machine, l'évaluation du système est crucial. Dans le cas d'un classifieur, l'évaluation est directe. En effet, le système est entraîné avec un jeu de données d'entraînement puis évalué sur un jeu de données spécifique. Dans le cas d'un système de recommandation, l'évaluation est plus complexe. L'objectif d'un système de recommandation étant de proposer à un utilisateur des objets qui lui sont pertinents et qui répondent à ses goûts. Néanmoins, la question de la pertinence est problématique. Comment peut-on mesurer la pertinence d'une recommandation ou d'un ensemble de recommandations ? Pour répondre à cette question, il existe différents moyens d'évaluer un système de recommandation.

La première méthode est l'évaluation hors ligne. Le principe de cette méthode est de considérer une première partie de l'historique d'utilisation pour entraîner le système puis utiliser l'autre partie pour évaluer les recommandations. Ainsi, si une recommandation est présente dans la partie cachée de l'historique, alors elle est considérée comme pertinente.

La seconde méthode consiste à faire passer des tests utilisateurs. Le principe est donc de faire utiliser le système par un panel de cobayes et leur demander à posteriori leur avis sur les recommandations.

Enfin, la troisième stratégie est l'AB testing. L'objectif est de diviser tout ou partie des utilisateurs d'un système en plusieurs groupes et de proposer à chaque groupe un système de recommandation différent. Ensuite, on laisse les groupes utiliser le système pendant un certain temps pour ensuite évaluer les systèmes en comparant différentes statistiques agrégées durant l'utilisation.

8.1.1 Hors ligne

Comme dit précédemment, l'évaluation hors ligne repose sur le principe de cacher une partie de la vérité terrain pour évaluer le résultat de l'algorithme. Ainsi, notre algorithme de recommandation va prendre en entrée un historique d'utilisation. Cet historique contient l'ensemble des interactions entre utilisateurs et objets. L'objectif est donc de ne prendre qu'une partie de ses interactions et construire un modèle avec celui-ci. Puis avec le reste des interactions, on va évaluer les recommandations obtenues après l'entraînement.

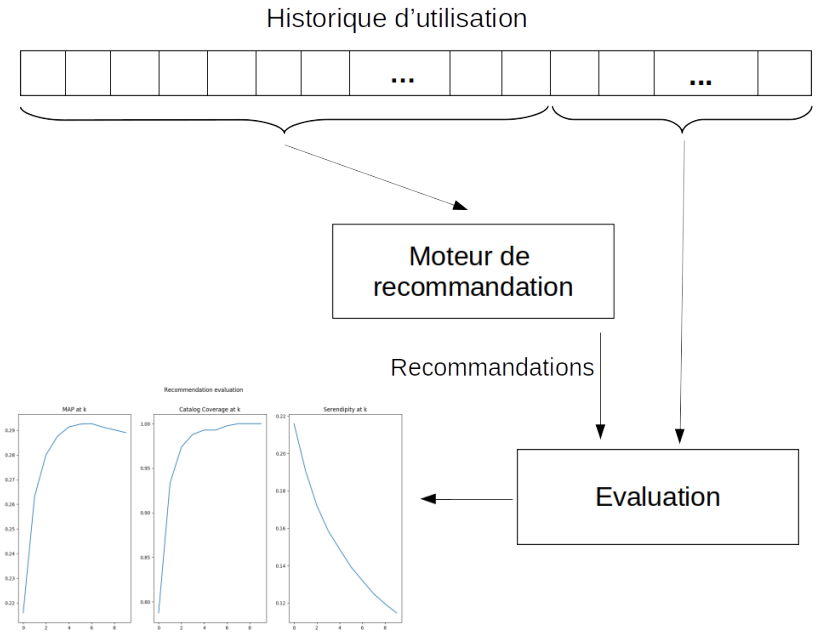


FIG. 13 – *Évaluation hors ligne.*

Pour obtenir de meilleurs résultats, les recommandations devront être au plus proche du comportement des utilisateurs. En effet, lors d'une évaluation hors ligne, on considère qu'une recommandation est pertinente pour un utilisateur si une interaction entre cet utilisateur et l'objet apparaît dans la partie cachée de l'historique. Cette définition de la pertinence pose des questions. En effet, l'historique ne contient aucune information sur la pertinence d'une interaction. Par exemple pour la musique, on sait qu'un utilisateur a écouté un morceau, mais on ne sait pas s'il l'a aimé. Ainsi, il est possible d'effectuer des recommandations que l'on évalue pertinentes alors qu'elles ne le sont pas (faux positif). Le problème contraire se pose aussi. On peut intuitivement penser que ce n'est pas parce qu'un utilisateur n'a jamais écouté un morceau qu'il ne l'aime pas et donc que celui-ci n'est pas pertinent. Ainsi, le système risque de recommander des objets pertinents qui seront évalués comme non-pertinents (faux négatifs). De plus, cette méthode d'évaluation est dépendante des données utilisées.

Néanmoins, cette méthode est très simple à mettre en œuvre et permet facilement de comparer divers algorithmes. En effet, cette méthode n'a besoin que d'un historique d'utilisation et des recommandations pour obtenir une évaluation et est rapide. Ainsi, il apparaît que c'est une méthode très utilisée lors de la conception d'un algorithme de recommandation.

8.1.2 Tests utilisateurs

Étant donné les limites de l'évaluation hors ligne, il semble plus pertinent de demander aux utilisateurs eux-même si les recommandations sont pertinentes pour eux. Ainsi, afin d'évaluer un système de recommandation, on peut directement le confronter à des testeurs et recueillir leurs avis sur la pertinence des recommandations. Cette approche consiste à faire passer des tests utilisateurs classiques. Il faut faire utiliser un système par un panel d'utilisateurs puis demander à ces utilisateurs de répondre à un questionnaire.

L'avantage de cette approche est que les réponses des utilisateurs ne sont pas discutables. En effet, dans le cas d'une recommandation, seul l'utilisateur peut affirmer sans se tromper

si une recommandation lui est pertinente ou non. Ainsi, il est aisé de savoir à quel point un système de recommandation est précis ou non. Aussi, il est possible de récupérer d'autres informations concernant le système depuis les utilisateurs. Par exemple, le moyen de proposer les recommandations à travers l'interface et l'organisation des recommandations peuvent être évalués et testés.

Néanmoins, effectuer des tests utilisateurs demande d'avoir un nombre d'utilisateurs suffisamment grand pour que le panel soit représentatif de la cible et donc cela coûte cher. De plus, cela demande beaucoup de temps pour construire les profils utilisateurs et confronter les utilisateurs au système. Enfin, il faut pouvoir établir un questionnaire permettant de récupérer au mieux les retours des utilisateurs.

8.1.3 AB Testing

La dernière approche est appelée AB testing ou évaluation en ligne. Son principe est de diviser les utilisateurs finaux en plusieurs catégories et de proposer à chacune de ces parties un moteur de recommandation différent. Ensuite, on laisse les utilisateurs être naturellement confronté aux recommandations et au bout d'un certain temps, on évalue les performances des différents moteurs de recommandation (voir figure 14).

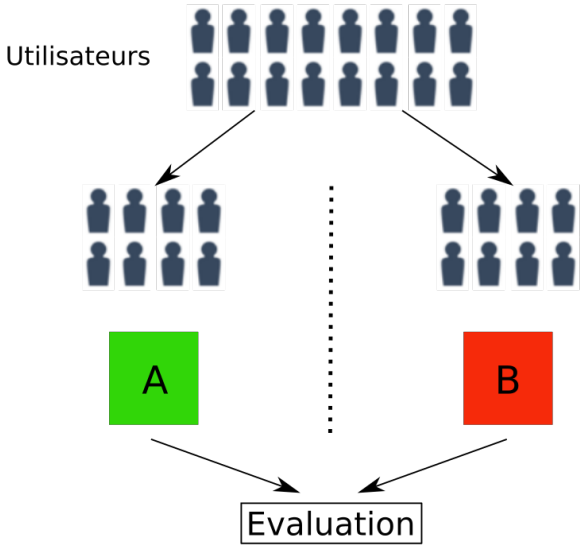


FIG. 14 – *Évaluation en ligne (AB testing).*

L'AB testing demande de pouvoir déployer les systèmes de recommandation en `live`. Un de ses avantages est que l'on peut comparer plusieurs moteurs de recommandation dans des conditions réelles d'utilisation. La seule limite à cela est qu'il faut garantir que le public confronté à chaque moteur de recommandation soit représentatif. Ainsi, selon la grandeur de la population d'utilisateurs, on ne peut comparer que quelques moteurs. De plus, l'AB testing demande un temps d'utilisation de plusieurs mois pour que les statistiques d'utilisation aient du sens.

8.2 Métriques d'évaluation

8.2.1 MAP

La première métrique à prendre en compte est la précision du système de recommandation. Cette métrique mesure la capacité du système à effectuer des recommandations pertinentes. Pour ce faire, il faut d'abord définir la mesure de la précision telle que le nombre d'objet pertinents recommandés sur le nombre d'objets recommandés :

$$P = \frac{\#\ recommandations\ pertinentes}{\#\ objets\ recommandés}$$

Cette métrique permet de mesurer la pertinence d'un ensemble de recommandations. Ainsi sur k recommandations, on mesure le nombre m tel que $m \leq k$ de recommandations pertinentes quel que soit leur position. En effet, cela ne prend pas en compte la position des recommandations dans la liste. Afin de prendre en compte l'ordre des recommandations et de favoriser le bon classement des recommandations pertinentes, on peut utiliser l'`average precision` (`AP`). De plus, on peut ne considérer qu'un nombre N de recommandations pour évaluer la précision. Ceci répond à la question : pour N recommandations, quel est mon `average precision` ?

Dans le cas où l'on recommande N objets, on définit l'`AP@N` telle que :

$$AP@N = \frac{1}{N}\sum_{k=1}^{N} P(k) * rel(k)$$

Avec :

$$rel(k) = \begin{cases} 1, & \text{si k est une recommandation pertinente} \\ 0, & \text{sinon} \end{cases} \tag{1}$$

Enfin, on peut obtenir le `mean average precision` à N (`MAP@N`) en moyennant l'ensemble des `AP@N`. Il s'agit d'une mesure de précision de l'ensemble du système de recommandation.

$$MAP@N = \frac{1}{N}\sum_{u=1}^{|U|}(AP@N)_u$$

8.2.2 Coverage

Le coverage correspond à la capacité du système de recommandation de proposer aux utilisateurs une plus ou moins grande proportion du catalogue. Le catalogue I correspond à l'ensemble des objets que le système peut recommander. Ainsi pour mesurer le coverage du catalogue, l'article [7] propose de calculer le ratio entre le nombre d'objets recommandés et le nombre total d'objets disponibles. Pour construire l'ensemble des objets recommandés $U_{j=1...N} I_L^j$, il faut prendre en compte un ensemble de recommandations faites à tous les utilisateurs sur une certaine période de temps. Ainsi le coverage du catalogue est défini tel que :

$$\text{Catalog coverage} = \frac{|U_{j=1...N} I_L^j|}{|I|}$$

Cette mesure de coverage est globale. Elle mesure quelle portion du catalogue est recommandée après un certain nombre de recommandations pour tous les utilisateurs. Il peut être

aussi intéressant de ne calculer le coverage que pour chaque utilisateur indépendamment. Ainsi, on pourrait étudier les distributions des coverages pour les utilisateurs. En effet, à l'aide de la moyenne de ces coverages et leurs écarts types, on peut capturer certaines caractéristiques du système de recommandation. Une moyenne faible signifie que le système recommande une faible portion du catalogue en moyenne à ces utilisateurs. Alors que si la moyenne est forte, c'est le contraire : il construit pour chaque utilisateur des recommandations diverses. L'écart type quant à lui nous renseigne sur la stabilité du système. En effet, une faible valeur indique que tous les utilisateurs ont le même niveau de diversité alors qu'une grande valeur signifie que certains utilisateurs ont des recommandations diverses et d'autres non.

8.2.3 Serendipité

La sérendipité est la mesure de la surprise du système de recommandation. Cela mesure la capacité de l'algorithme de proposer aux utilisateurs des objets surprenants et pertinents. Comme mentionnée précédemment, la pertinence d'une recommandation est difficile à mesurer objectivement. La surprise, elle aussi est difficile à évaluer. Dans l'article [7], la surprise est définit en utilisant un algorithme de recommandation naïf. En effet, pour savoir si une recommandation est inattendue, il faut qu'elle soit absente des recommandations de l'algorithme naïf. Ainsi, on peut définir PM comme l'ensemble des objets recommandés par l'algorithme naïf et RS l'ensemble recommandé par l'algorithme que l'on évalue.

$$UNEXP = \frac{RS}{PM}$$

$$SRDP = \frac{\sum_{i=1}^{N} rel(UNEXP_i)}{N}$$

Néanmoins, cette méthode repose grandement sur un second système de recommandation. Selon ce système, la mesure de la sérendipité peut beaucoup changer. C'est pourquoi l'article [10] propose d'autres moyens pour mesurer la surprise d'une recommandation et propose notamment d'utiliser une fonction de distances entre objets. Ainsi, les auteurs définissent la surprise comme la distance minimale entre l'objet recommandé et l'ensemble des objets P utilisés par l'utilisateur. Ainsi, si cette distance est grande, cela veut dire que l'objet recommandé est distant de tous les objets de l'utilisateur. Alors que si elle est faible, cela veut dire que l'objet recommandé est semblable à au moins un objet de l'ensemble de l'utilisateur et donc il n'est pas ou peu surprenant. Ainsi, la surprise est définie telle que :

$$S(i) = min_{j \in P} dist(i,j)$$

Avec $dist(i,j)$ la fonction de distance entre deux objets i et j.

Cette méthode semble néanmoins très dépendante de la fonction de distance choisie. Il faut pouvoir garantir que la distance mesurée ait un sens vis-à-vis de la définition de deux objets. Il faut que cette distance garantisse que deux objets semblables pour les utilisateurs ait une distance grande alors que deux objets différents non. Cela implique que les caractéristiques utilisées pour calculer cette distance représentent bien les objets considérés et soient discriminantes.

9 Éxperiences

Afin de valider nos hypothèses et d'évaluer l'impact de l'ajout du contenu dans les recommandations du système existant, on va mettre en place un environnement de tests et diverses expériences. Ces expériences et leurs mises en place seront décrites ici. Mais d'abord, on traitera rapidement des données utilisées de leurs formes et leurs contenus.

9.1 Les données

Les données utilisées pour l'ensemble de nos expériences sont :

- Un historique d'utilisation des morceaux joués. Cet historique contient des entrées renseignant chaque morceau joué lors de sessions de jeux, en prenant en compte que chaque session est liée à un utilisateur. Ainsi, on a plusieurs entrées (morceaux) par session et plusieurs sessions par utilisateurs.
- Les vidéos et musiques correspondant aux morceaux présents dans le jeu.

On peut compter un nombre de 419 morceaux de musique pour un ensemble de 4999 utilisateurs. Ces utilisateurs ont joué en tout 161810 sessions de jeu avec en moyenne 6.26 morceaux joués par session et en moyenne 3.24 session de jeux par utilisateurs.

On peut voir sur la figure 15 le nombre de parties par morceaux. On a donc les morceaux en abscisse et le nombre de parties en ordonnée. Les morceaux sont classés du plus joué au moins joué. On remarque que la courbe décroît rapidement pour tendre vers une valeur très faible. On a ainsi un petit nombre de morceaux beaucoup joués puis un grand nombre de morceaux peu joués.

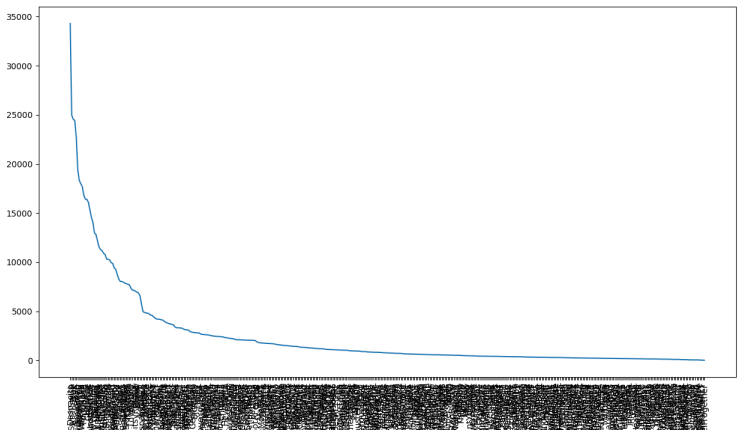


FIG. 15 – *Nombre de parties par morceaux.*

Ces données ne contiennent aucune information sur l'expérience du joueur lors de ses parties. En effet, aucune mention n'est faite quant à ses performances et son comportement lors du jeu. Il est donc impossible de savoir si une partie jouée est une partie plaisante ou non pour le joueur.

9.2 Méthodes experimentales

9.2.1 Sélection et réduction des caractérisques

Afin d'évaluer l'impact et le résultat de la sélection et/ou réductions des caractéristiques, nous avons déjà calculé ces sélections et ou réductions. Pour ce faire, on a utilisé l'ensemble des caractéristiques audio et vidéo pour construire en tout 8 représentations, 4 pour le contenu audio et 4 pour le contenu vidéo. Ces représentations sont les suivantes :

- La représentation complète contenant toutes les caractéristiques (`ALL`).
- La sélection des 100 caractéristiques les plus importantes obtenues avec l'arbre décisionnel (`SF` pour `selected features`).
- Les caractéristiques réduites à l'aide de la `PCA` (`PCA`). On a réduit le nombre de dimensions à 100 et donc on obtient en sortie 100 caractéristiques.
- Les caractéristiques encodées par l'autocodeur (`encoded`). En récupérant la représentation obtenue dans la couchecentrale du réseau, on obtient une réduction de dimension. Dans notre cas, on a cherché à réduire vers 100 caractéristiques.

La première chose est d'évaluer l'impact de la réduction de dimension sur le calcul des similarités. Ainsi, pour chaque représentation, on calcule l'histogramme des similarités entre tous les morceaux pour les similarités cosinus et cosinus centré entre tous les morceaux. À cet histogramme, on ajoute le calcul de la moyenne des similarités et leurs écart type. Le résultat attendu pour les représentations complètes est un écart type proche de 0. Ceci se manifesterait par un histogramme comprenant un pic et des zéros partout ailleurs. Plus concrètement, sur le fait que tous les morceaux sont équidistants les uns des autres. L'objectif des autres représentations est d'aplatir au maximum l'histogramme et donc obtenir un écart type le plus grand possible.

9.2.2 Recommandation

Afin de valider l'hypothèse principale, on a donc mesuré l'impact de l'ajout du contenu dans la recommandation. L'intérêt est donc de comparer la version déjà existante aux versions hybrides que l'on testera. Pour ce faire, on va évaluer toutes les versions (dont celle de base) puis on comparer simplement les résultats obtenus pour chaque métrique en soustrayant les résultats obtenus pour les versions hybrides par les résultats de la version de base. Ainsi, si l'on obtient une valeur positive, cela veut dire que la version hybride performe mieux que la version de base. L'objectif idéal étant de n'obtenir que des valeurs positives pour toutes les métriques.

Pour chaque version du système, on va évaluer les métriques `MAP`, `coverage` et la `sérendipité` à N pour $N \in [1,10]$. Ainsi, pour chaque métrique, on peut voir l'évolution des performances selon le nombre de recommandations proposées.

Concernant les méthodes hybrides en places, on cherche à exhaustivement combiner toutes les possibilité en notre possession. Ainsi, on a :

- 2 représentations pour l'usage. D'abord l'usage complet où l'on représente chaque morceau pour un vecteur représentant si le morceau a été joué lors d'une session ou non. Puis, l'usage réduit en utilisant un `SVD`. Dans le second cas, on calcule la décomposition de la matrice user-item avec un nombre de coefficients de 32. Puis, on utilise la matrice représentant les objets comme représentation de l'usage.

– 4 représentations du contenu vidéo et 4 représentations du contenu audio. Pour chacun, nous avons l'ensemble des caractéristiques, les caractéristiques sélectionnées, la réduction de la `PCA` et enfin l'encodage des caractéristiques.
– 2 fonctions de similarités : la similarité cosinus et cosinus centré.
– 4 méthodes d'hybridation : l'empilement, le mélange et garder la similarité minimale ou maximale.

Nous avons donc mis en place 256 possibilités. Il s'agit donc d'une évaluation exhaustive de toutes les possibilités que l'on propose. Cela permet de déterminer le mélange le plus performant.

De plus, on cherche à évaluer le choix de la similarité en comparant toutes les versions utilisant la similarité cosinus à celles utilisant la similarité cosinus centré. Pour ce faire, on compare chaque méthode correspondante pour chaque similarité et on moyenne les différences obtenues. Ainsi, on peut avoir un point de vue global sur le choix de la fonction de similarité.

Enfin les différences entre les différentes méthodes seront visualisées sur des courbes relatives (voir figure 16). Ces courbes montreront les performances relatives de la méthode utilisée par rapport à la méthode de base. Dans le cas où les performances sont positives, on pourra voir la courbe représentant la différence devenir positive et prendre une couleur verte. Dans le cas où la méthode, est moins performante que la méthode de base, alors la courbe sera rouge et négative.

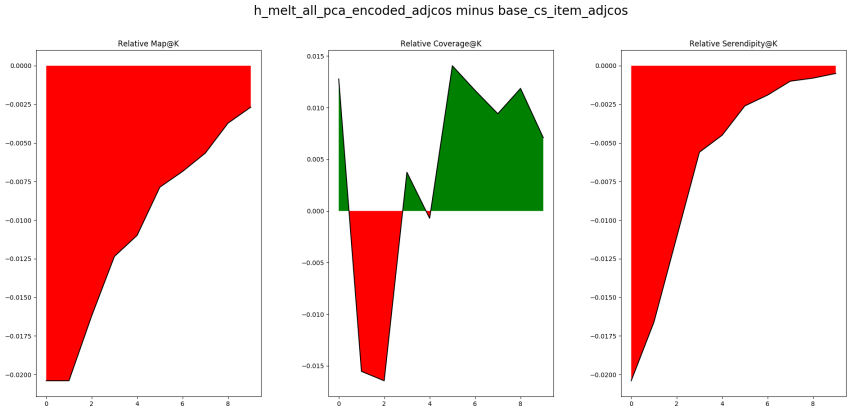


FIG. 16 – *Exemple de courbe de comparaison.*

La méthodologie d'évaluation est la suivante :

1. À partir de l'historique d'entraînement, construction de la matrice `user-item`. Récuperation et normalisation des caractéristiques audio et vidéo précalculées.
2. Calcul de la matrice de similarité selon la méthode évaluée.
3. À partir de l'historique de test, construction des prédictions et des recommandations.
4. Evaluation des recommandations.

9.2.3 Cold Start

Le cold start est une problématique courante dans le cas de la recommandation. Il existe deux types de cold start, le cold start utilisateur et objet.

Dans le cas du cold start utilisateur, cela correspond à la situation où un utilisateur arrive dans le système et où il n'a pas interagi avec le système. Le problème est que très souvent

les moteurs de recommandations ne savent pas gérer ces cas et ils ne peuvent construire des recommandations pour cet utilisateur.

Dans le cas du cold start objet, il s'agit de la situation où un objet est ajouté au système. Ce cas est moins problématique que le cold start utilisateur. En effet, certains moteurs de recommandation sont capables de recommander le nouvel objet malgré qu'aucun utilisateur n'ait interagi avec l'objet. Néanmoins, la méthode courante du filtrage collaboratif ne peut recommander cet objet avant qu'il ait pu être utilisé par des utilisateurs.

On cherche donc aussi à évaluer l'impact de l'ajout du contenu dans un contexte de cold start. Pour créer un tel contexte, on va filtrer l'usage pour créer une situation de cold start utilisateur et un cold start objet.

Pour le cold start utilisateur, on cherche à supprimer l'usage d'un certain nombre de sessions. Il faut néanmoins garder une partie de l'usage pour que l'utilisateur puisse exister.

Concernant le cold start objet, on va simuler l'ajout de nouveaux morceaux en supprimant l'usage de certains morceaux. Comme pour le cold start utilisateur, il faut garder un peu d'usage concernant l'objet pour que l'objet existe dans le système.

Les ensembles d'utilisateurs et d'objets filtrés sont choisis aléatoirement dans l'ensemble des utilisateurs et objets utilisés pour l'évaluation. De plus, pour chaque cas de cold start, on met en place la même méthodologie de test que pour la recommandation simple.

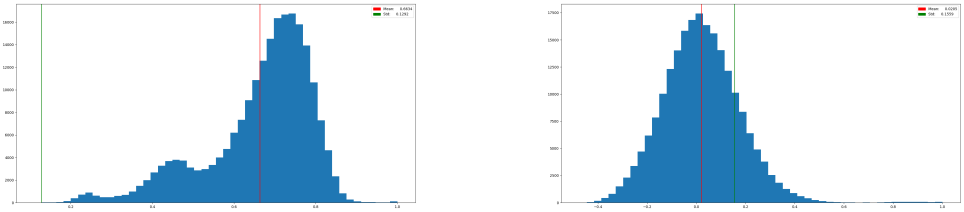
Normalement, ajouter le contenu devrait améliorer les résultats sur le cold start objet. En effet, dans le cas où l'on ajoute un nouvel objet, il est impossible d'établir de la similarité avec l'usage, car il n'a pas d'usage et donc on ne peut pas le recommander, il ne sera le plus proche voisin de personne. Or, il est possible de calculer une similarité avec son contenu et donc il devient possible de le recommander.

10 Résultats

10.1 Impact de la réduction et la sélection des caractéristiques

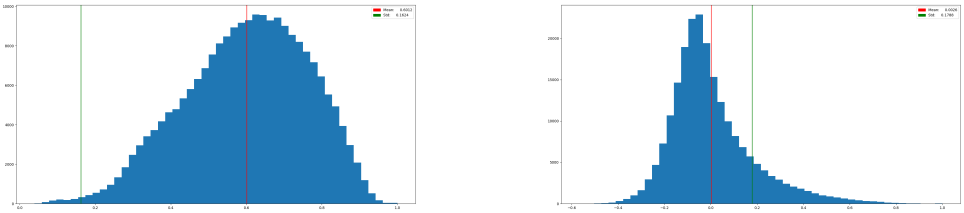
Comme précisé avant, afin de juger de l'impact de la réduction et de la sélection des caractéristiques, on utilise la distribution des similarités obtenues avec les différentes représentations. On compare ainsi si on règle en parti le problème de haute dimensionnalité. L'objectif étant d'obtenir des objets qui ne sont pas tous équidistants entre eux. L'ensemble des résultats présenté ici auront été calculés avec la similarité cosinus centré.

Ainsi, pour le contenu complet audio et vidéo, on obtient les distributions présentes dans les figures 17 18.



(a) *Similarité cosinus.* (b) *Similarité cosinus centré.*

FIG. 17 – *Différences selon la fonction de similarité pour le contenu audio.*



(a) *Similarité cosinus.* (b) *Similarité cosinus centré.*

FIG. 18 – *Différences selon la fonction de similarité pour le contenu vidéo.*

La figure 19 nous montre toutes les réductions pour chaque fonction de similarité pour le contenu audio.

On peut observer que la sélection des caractéristiques a peu d'influence sur la distribution compare aux distributions obtenues à partir du contenu complet. La sélection des caractéristiques sur le calcul des similarités a donc très peu d'influence pour le contenu audio.

Concernant l'utilisation de la `PCA` on peut observer que les distributions sont plus étroites que celles d'origines. Elles sont bien centrées en zéro, mais ont néanmoins un écart type très faible (0.103 chacune). Ceci implique que les différences de distances entre les objets les plus dissemblables sont faibles.

Pour ce qui est de l'utilisation de l'autoencodeur, on remarque un pic sur la distribution utilisant la fonction cosinus. La fonction cosinus centré quant à elle nous permet d'obtenir une distribution presque centrée en zéro avec un grand écart type (0.3572). Il apparaît donc que

l'utilisation de l'autoencodeur avec la fonction cosinus centré donne de bon résultats quant au calcul des similarités audio.

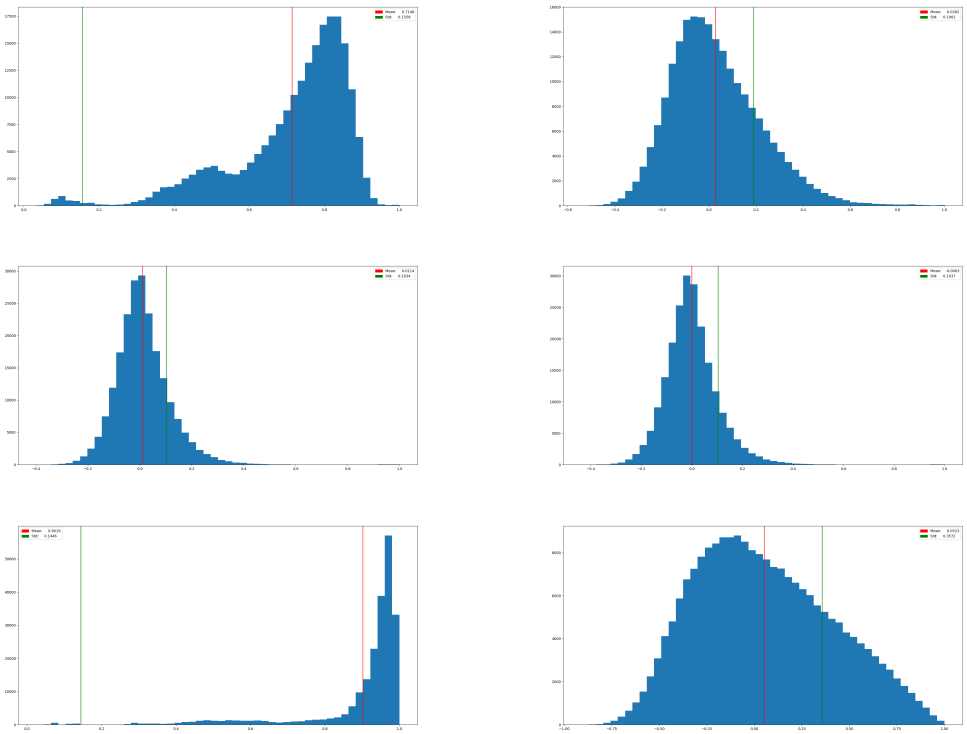


FIG. 19 – *Distribution des similarités pour les contenus audio réduits obtenus avec les deux fonctions de similarité. La colonne de droite correspond à la similarité cosinus et celle de gauche à la similarité cosinus centré. La première ligne contient les résultats pour les caractéristiques sélectionnées, la seconde ligne pour celles réduites avec la* `PCA` *et la dernière avec celles encodées.*

La figure 20 nous montre toutes les réductions pour chaque fonction de similarité pour le contenu vidéo.

Tout d'abord, on peut rapidement observer que comme pour le contenu audio, l'utilisation de la `PCA` n'est pas performante. En effet, on obtient presque des pics centrés en 0 avec de faibles écarts types (0.109). Le même constat en pire peut être fait pour l'autoencoder. En effet, dans ce cas la, on obtient des pics en 1. Les distributions obtenues avec le `PCA` et l'autoencodeur ne sont pas bonnes du tout.

Concernant la sélection de caractéristiques, on peut observer des distributions un peu semblables aux distributions d'origines. Elles sont respectivement un tout petit peu meilleures si l'on compare leur écarts types. Malgré la forme de dôme de la distribution obtenue avec la fonction cosinus, celle obtenue avec la fonction cosinus centré est plus intéressante. Premièrement, elle obtient un écart type supérieur, mais aussi n'est pas comprise seulement dans l'interval [0,1]. La variation des similarités est donc plus grande. Dans le cas du contenu vidéo, on préférera donc l'utilisation de la sélection de caractéristiques avec la fonction de cosinus centré.

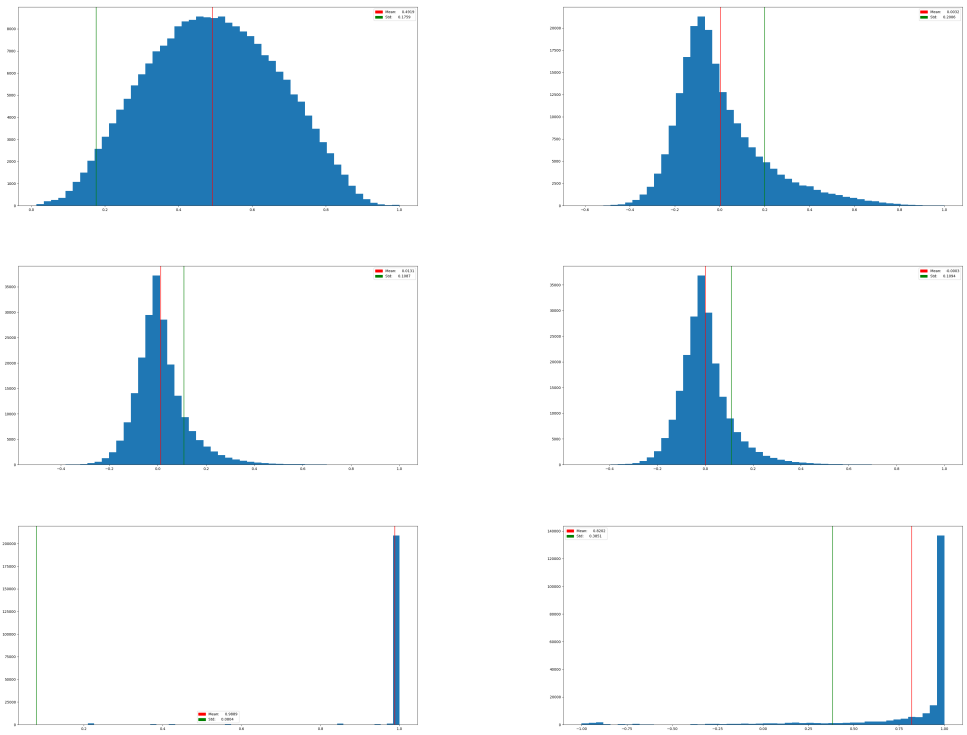


FIG. 20 – *Distribution des similarités pour les contenus vidéos réduits obtenus avec les deux fonctions de similarité. La colonne de droite correspond à la similarité cosinus et celle de gauche à la similarité cosinus centré. La première ligne contient les résultats pour les caractéristiques sélectionnées, la seconde ligne pour celles réduites avec la* `PCA` *et la dernière avec celles encodées.*

10.2 Choix de la fonction de similarité

Afin de comparer les deux métriques de similarités, on cherche à comparer l'ensemble des méthodes utilisant une métrique contre l'ensemble des méthodes utilisant l'autre métrique. On va donc évaluer chaque méthode avec chaque similarité et faire la différence des performances pour chaque méthode. En moyennant l'ensemble des différences, on obtient une différence globale entre les deux métriques. Si l'on applique ceci, on obtient le résultat de la figure 21.

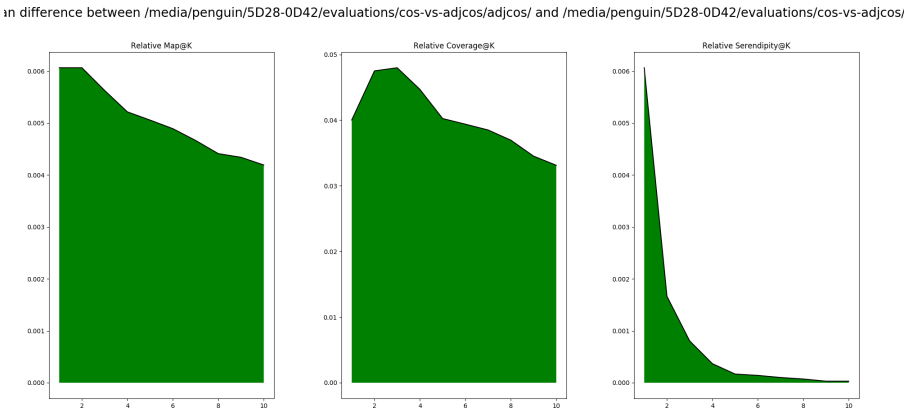


FIG. 21 – *Différence globale entre la similarité cosinus centré et la cosinus.*

On observe donc la différence moyenne des résultats obtenus avec la similarité cosinus centré moins ceux obtenus avec la similarité cosinus. On peut voir des valeurs positives et des courbes vertes pour l'ensemble des métriques d'évaluation. On peut noter que la sérendipité converge vers zéro très vite et que la différence de `MAP` est très faible. On peut donc en conclure que le choix de la fonction de similarité a peu d'impact sur la sérendipité et le `MAP`. Néanmoins, on voit que le coverage est globalement meilleur avec la similarité cosinus centré. À 10 recommandations, on obtient en moyenne une valeur d'environ 3.4% de plus. Cela semble faible, mais c'est une moyenne de toutes les différences.

Si l'on s'intéresse à la différence entre les deux fonctions concernant l'utilisation seule de l'usage, on peut observer la différence présentée à la figure 22. Cette figure nous montre des différences minimes mais positives pour la sérendipité et le `MAP`. Elle nous montre aussi une grande différence concernant le coverage. Cette différence est de hauteur d'un peu moins de 5% si l'on considère 10 recommandations, mais est supérieure pour un nombre de recommandations inférieur. Cela veut dire que l'utilisation de la fonction cosinus centré entraîne une augmentation plus rapide du coverage que la fonction cosinus.

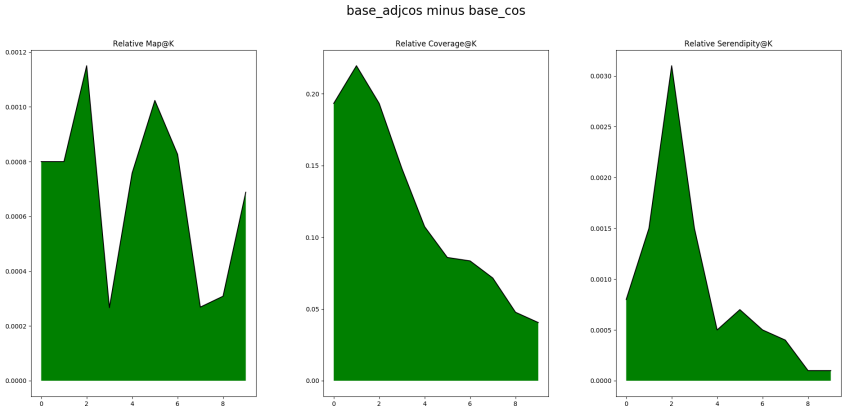


FIG. 22 – *Différence entre la similarité cosinus centré et la similarité cosinus pour du filtrage collaboratif pur.*

10.3 Impact de l'ajout du contenu

10.3.1 Sur la recommandation

Premièrement, on peut observer les résultats obtenus avec la méthode existante de référence sur la figure 23. Cette méthode a été légèrement modifiée pour utiliser la similarité cosinus centré. En effet, avec cette métrique, les performances augmentent et afin d'isoler l'impact seul de l'ajout du contenu, on préfère comparer avec la meilleure version entre la fonction cosinus et cosinus centré.

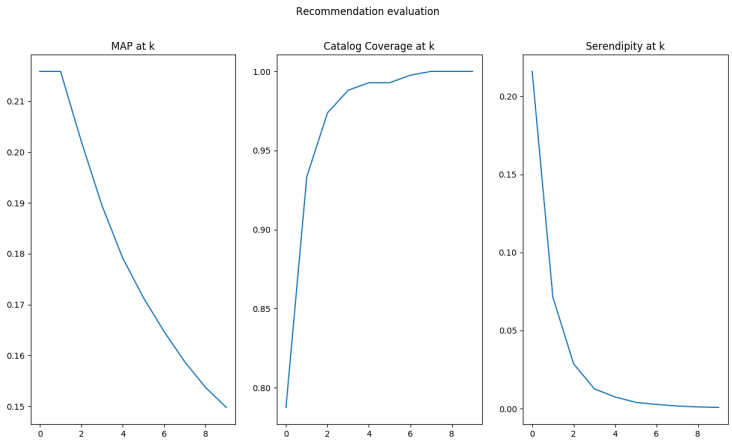


FIG. 23 – *Résultats de référence.*

Sur ce résultat, on peut observer la précision pour une recommandation par utilisateur atteindre 21% puis descendre à 15% pour 10 recommandations considérées. On voit donc la précision qui diminue si l'on considère de plus en plus de recommandations. Concernant le `coverage`, on peut voir que les valeurs tendent rapidement vers la valeur de 1 (valeur maximale). Ceci indique que pour un certain nombre de recommandations par utilisateurs, on tend à recommander l'ensemble du catalogue de niveaux. Ce point peut facilement être expliqué par la taille du catalogue vis-à-vis au nombre de joueurs. En effet, pour à peu près 400 morceaux, on a plusieurs milliers d'utilisateurs considérés. Il semble donc logique que si l'on construit plusieurs recommandations par utilisateurs, on va couvrir tout le catalogue quelque soit la méthode utilisée. Néanmoins, il peut être intéressant de comparer l'évolution de cette métrique selon le nombre de recommandations considérées. Dans ce cas, la seule possibilité d'amélioration du `coverage` est de converger vers 1 plus rapidement que la méthode de base. La sérendipité, quant à elle, donne de mauvais résultats. Ceci est entièrement normal. En effet, comme on le voit avec la précision, plus on augmente le nombre de recommandations par utilisateur, plus la précision diminue et donc moins la proportion de recommandations pertinentes est grande. Or, cette pertinence est prise en compte par la sérendipité et donc la sérendipité va automatiquement diminuer. De plus, on peut voir qu'elle tend vers 0. Ceci vient du fait que le peu de recommandation pertinente est noyé dans l'ensemble de toutes les recommandations effectuées et donc tend vers 0. Il faut noter que concernant la sérendipité, un biais est introduit par la méthode d'évaluation. En choisissant d'utiliser l'utilisation cachée d'un utilisateur pour établir la pertinence d'un objet, tout objet pertinent peut difficilement être considéré comme surprenant étant donné que l'utilisateur a choisi cet objet. Comme pour le `coverage`, on comparera l'évolution de la sérendipité mais il ne s'agit pas d'une métrique centrale dans notre évaluation.

Dans cette partie ne seront pas présentés tous les résultats, seulement les plus intéressants et pertinents.

Tout d'abord, il est bien de noter que les résultats varient énormément selon la méthode d'hybridation choisie. En effet, le choix de comment mêler les différentes représentations a plus d'impact que le choix des représentations prises en compte.

On peut tout d'abord constater que l'empilement des représentations donne de bons résultats qui restent globalement équivalents aux résultats de la méthode de base. Comme on peut voir sur la figure 24, on peut voir quelques résultats variables. Néanmoins, on remarque que les résultats sont soit équivalent soit moins bons. Sur les meilleurs résultats, on peut voir des courbes vertes,

mais les valeurs sont minimes voir nulles. On peut donc voir trois exemples de résultats. Le premier nous montre un résultat équivalent aux résultats de références. On peut voir beaucoup de verts, mais les valeurs sont très faibles et peuvent donc être considérées comme nulles. Concernant les autres résultats, nous sommes soit en dessous de la référence, soit équivalent à celle-ci. L'empilement nous permet donc dans les meilleurs cas de reproduire la méthode de référence. Ceci est expliqué par le fait que la part de l'usage dans la nouvelle représentation est beaucoup plus grande que la part du contenu. Ainsi, lors du calcul des similarités, l'usage a plus d'importance dans le calcul. Ceci peut se confirmer par les résultats semblables au second exemple. En effet, on obtient ce type de résultat si l'on utilise une réduction de l'usage obtenu avec un `SVD`. Dans ce cas, l'usage prend une part plus faible et on obtient de moins bons résultats, semblables à des résultats que l'on pourrait obtenir si l'on considérerait seulement le contenu. On peut en conclure que l'empilement des représentations est une approche dépendante de la taille et des différences de tailles des différentes représentations.

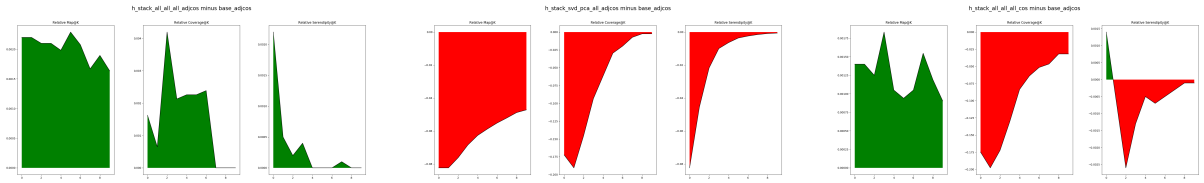


FIG. 24 – *Exemple de résultats obtenus avec l'empilement des representations.*

Les deux exemples présentés dans la figure 25 ont été obtenus avec l'approche de mélange par pondération des similarités. Le premier exemple est un des nombreux mauvais résultats obtenus avec cette approche. Cette méthode donne de mauvais résultats dans la plupart des cas. Néanmoins, comme on peut le voir avec le second exemple, on peut obtenir des résultats intéressants. On voit que la courbe de `MAP` et de sérendipité sont négatives, mais convergent vers 0 et donc donnent au-delà de 10 recommandation des résultats équivalents mais le `coverage` lui converge plus vite. Ceci semble prometteur. À noter que les coefficients ont été choisis pour que chaque représentation ait le même poids dans le calcul de la similarité finale. Il pourrait être intéressant d'explorer différentes valeurs de ces coefficients pour voir leurs impacts sur les résultats. Ces résultats restent néanmoins négatifs. Il faut pour améliorer un peu une métrique, sacrifier les deux autres.

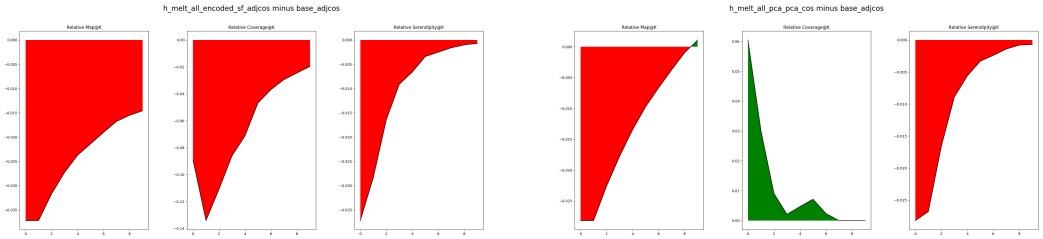


FIG. 25 – *Exemple de résultats obtenus avec le mélange des similarités.*

Les deux exemples présentés dans la figure 26 ont été obtenus avec l'approche de combinaison des similarités (mélange avec les fonction ou `max`). Les résultats de ces deux approches sont très similaires. Soit les résultats sont très négatifs (exemple un et deux) soit on améliore une métrique en sacrifiant les autres. En effet, on peut voir sur le troisième exemple que le `coverage`

converge plus rapidement vers la valeur maximale. Néanmoins, ceci est obtenu en sacrifiant grandement la précision. Dans ce cas, le moteur est beaucoup moins performant car il propose beaucoup moins de recommandations pertinentes.

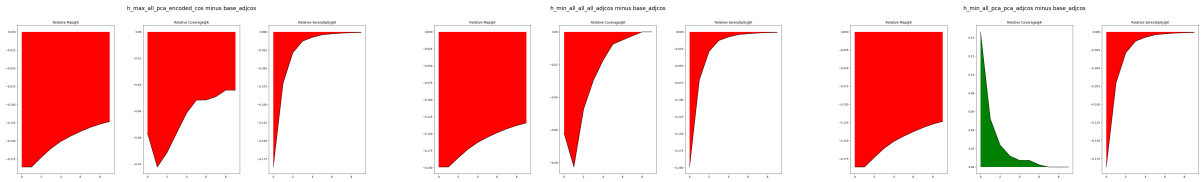


FIG. 26 – *Exemple de résultats obtenus avec la combinaison des similarités (fonction* `min` *ou* `max`*).*

On peut en conclure que l'ajout du contenu dans ce système n'améliore pas ou très peu les recommandations obtenues. Ceci peut s'expliquer par un catalogue faible. D'un coté avec peu d'objets on finit rapidement à recommander l'ensemble des morceaux quelle que soit la méthode utilisée. Aussi, chaque objet est du coup beaucoup utilisé et donc sa représentation d'utilisation est assez complète et donc est une bonne représentation. Ainsi, l'usage prend beaucoup d'importance et donne de très bons résultats. Ainsi, en ajoutant le contenu, on ne peut pas beaucoup améliorer la couverture du catalogue et on va automatiquement pénaliser la précision.

10.3.2 Sur le Cold Start

Cold start utilisateur: Si l'on observe les résultats obtenus dans un contexte de `cold start` utilisateur, on peut noter que nos méthodes donnent des résultats moins bons ou équivalents que ceux de la méthode de référence. Ceci est attendu. Effectivement, l'ajout du contenu n'apporte rien dans le cas d'un `cold start` utilisateur. On reste donc dans un cas équivalent au contexte d'évaluation normal. Et donc l'ajout du contenu ne permet pas d'avoir de meilleurs résultats. On obtient une couverture du catalogue qui converge plus rapidement vers l'ensemble des objets couverts et la différence de précisionest soit négative soit elle converge vers 0.

Cold start objet: Dans le cas du `cold start` objet, on peut s'attendre à avoir de bons résultats. En effet, l'ajout du contenu devrait permettre de compenser l'absence d'utilisation d'un nouvel objet et donc de calculer la similarité de cet objet avec tous les autres. Ceci se confirme par quelques résultats. La figure 27 nous montre deux exemples de bons résultats. On peut voir la différence de précision converger vers 0 et la couverture du catalogue être positive et terminer sur une valeur positive. Ceci indique que notre méthode arrive dans ces deux cas à recommander des objets que la méthode de référence ne recommande pas pour un niveau de précision équivalent. Ces résultats sont obtenus en mélangeant les similarités et en utilisant la fonction de similarité cosinus centré.

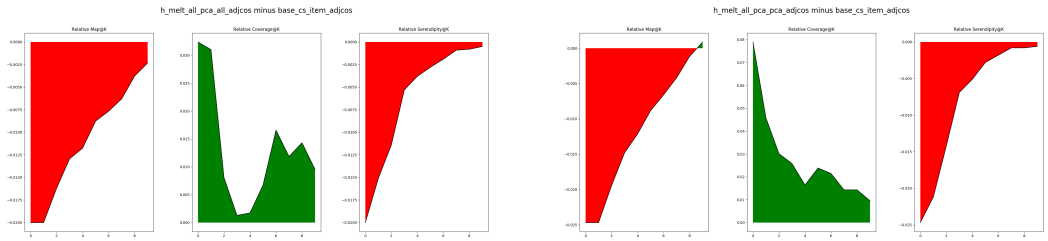


FIG. 27 – *Exemple de bons résultats obtenus dans un contexte de* `cold start` *objet.*

Les autres approches d'hybridation donnent des résultats mitigés. Soit la couverture du catalogue est meilleure et la précision est bien en dessous, soit les résultats sont équivalents, soit ils sont totalement mauvais.

Néanmoins, les meilleurs résultats sont encourageants et confirment l'hypothèse selon laquelle l'utilisation du contenus aide dans le cas d'un `cold start` objet.

11 Perspectives

11.1 Données

Dans le cas d'un jeu vidéo, il pourrait être intéressant de prendre en compte des informations liées au jeu et au `gameplay` pour améliorer les recommandations. En effet, dans notre cas, aucune information implicite n'est utilisée car elles ne sont pas en notre possession. Néanmoins, des informations sur l'expérience utilisateur lors des niveaux pourraient être intéressantes pour améliorer les recommandations. Ces informations pourraient être les performances du joueur telles que calculées par le jeu, ou bien son comportement dans le jeu (actions entreprises, choix effectués, échecs et réussites dans les niveaux), ou bien même une évaluation du niveau des joueurs et du niveau de difficulté des niveaux. Ces informations pourraient être utilisées pour personnaliser le challenge que le jeu propose au joueur et ainsi pour l'encourager à jouer et à améliorer ses performances. En effet, en proposant un challenge adapté au joueur, on peut créer une expérience optimale où le joueur sera pleinement immergé dans le jeux [4]. De plus, la recommandation de contenu en jeux est peu étudiée actuellement, mais reste prometteuse. Ceci pourrait améliorer l'expérience utilisateur mais fait néanmoins perdre aux concepteurs du jeu leur contrôle sur l'expérience finale. La thèse de master [3] de Monsieur Henrique Moises Fernandes propose une approche de recommandation de contenu en jeu. Il apprend du comportement du joueur son profil de joueur pour ensuite lui proposer du contenu dédié.

11.2 Contenu audio et video

Un travail supplémentaire sur le choix des caractéristiques audio et vidéo pourrait être intéressant. En effet, il serait intéressant de trouver un lien entre le contenu audio et vidéo et l'utilisation que les joueurs ont des morceaux. Une approche intéressante serait d'extraire un maximum d'informations et d'utiliser un réseau de neurones profond pour, à partir de ces caractéristiques, converger vers une représentation de l'usage des morceaux. Le principe serait, à partir du contenu audio et vidéo, de converger vers l'usage et extraire une représentation intermédiaire apprise par le réseau. Afin de représenter l'usage des morceaux, un `SVM` peut être utilisé à partir de la matrice `user-item`. Ensuite, cette représentation mêlant contenu et usage pourrait être utilisée par un moteur de recommandation basé contenu.

De plus, effectuer un travail plus approfondi sur l'extraction de caractéristiques adaptées au problème de la recommandation pourrait être envisagé. Durant ce stage, nous avons choisi d'aller au plus simple et d'extraire un maximum d'informations (notamment pour l'audio) et de filtrer et/ou réduire ensuite. Il s'agit d'une approche simple, mais pas de l'approche la plus efficace ni rigoureuse.

11.3 Stratégies d'hybridation

Durant ce court stage, nous avons choisi de faire au plus simple pour mélanger le filtrage collaboratif et approche basée contenu. Mais un plus grand travail sur les méthodes d'hybridation aurait pu être effectué. En effet, il existe beaucoup de manières de combiner ces deux approches de la recommandation. Aussi, il pourrait être intéressant d'ajouter aussi le contexte dans la construction des recommandations. Dans le domaine de l'écoute musicale, la prise en compte du contexte a montré des résultats encourageant en améliorant les recommandations. Il pourrait

3. https://www.nexusmods.com/skyrim/mods/90626

être intéressant d'utiliser le moment où le joueur joue et l'environnement social du joueur pour lui proposer des niveaux. En effet, si le joueur joue avec ses amis, il peut être intéressant de leur proposer des niveaux adaptés au jeu à plusieurs.

11.4 Évaluation

Au vu des limites de la techniques d'évaluation, la mise en place d'`AB testing` nous apporterait un regard plus objectif sur les performances des méthodes utilisées. En effet, l'évaluation hors ligne est connue pour favoriser le filtrage collaboratif. Ceci vient du fait que cette technique d'évaluation récompense la capacité du système à reproduire le comportement d'un utilisateur et c'est ce à quoi le filtrage collaboratif est fort. L'approche basée contenu est forte pour recommander des objets dont le contenu est semblable à ce qu'aime l'utilisateur, or ceci n'est pas mesuré par la technique d'évaluation.

De plus, la mise en place d'autres métriques d'évaluation est à étudier. Tout d'abord, car la précision est un élément important pour un moteur de recommandation, mais recommander des objets pertinents n'est pas le seul objectif d'un moteur de recommandation. On peut en effet attendre d'un moteur de recommandation qu'il nous surprenne et nous fasse découvrir de nouveaux objets, qualité que la précision ne mesure pas. Dans ce cas, il est intéressant de mesurer la sérendipité du moteur. Mais cette métrique comporte plusieurs défauts. En effet, il est difficile de juger de la surprise d'une recommandation. Étudier la question plus en profondeur pourrait permettre de mieux évaluer la derendipité.

Enfin, étant donné le faible nombre de morceaux à recommander on constate que le coverage global converge rapidement vers une valeur de 100% quelle que soit la méthode. Il pourrait donc être intéressant de calculer les distributions de coverage par utilisateurs comme indiqué dans la partie concernant le coverage. Cela pourrait donner une métrique un peu plus précise et renseigner sur les différentes performances entre toutes les méthodes.

Conclusion

Dans un cas ou l'usage est très complet, on peut voir qu'ajouter le contenu au moteur de recommandation n'améliore pas nécessairement les performances, sauf dans le cas d'un `cold start` objet. Effectivement, dans le cas où on simule l'ajout de nouveaux morceaux dans le système, l'ajout du contenu permet dans de recommander plus de morceaux différents pour une précision équivalente. Les performances contrastées quand on ajoute le contenu restent néanmoins à contraster. En effet, la méthode d'évaluation utilisée nous permet de mesurer la reproduction de l'utilisation, or dans ce cas précis, le filtrage collaboratif est tout indiqué. Donc ces résultats contrastés peuvent être expliqués par l'historique d'utilisation complet pour le faible nombre d'objets présents ainsi que la méthodologie d'évaluation qui peut biaiser les résultats.

Références

[1] Simon funk's blog - netflix update: Try this at home. `http://sifter.org/simon/journal/20061211.html`. Accessed: 2018-09-02.

[2] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? In Catriel Beeri and Peter Buneman, editors, *Database Theory — ICDT'99*, pages 217–235, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[3] Leo Breiman. *Classification and regression trees*. Routledge, 2017.

[4] Jenova Chen. Flow in games (and everything else). *Commun. ACM*, 50(4):31–34, April 2007.

[5] Yashar Deldjoo, Mehdi Elahi, Paolo Cremonesi, Franca Garzotto, Pietro Piazzolla, and Massimo Quadrana. Content-based video recommendation system based on stylistic visual features. *Journal on Data Semantics*, 5(2):99–113, Jun 2016.

[6] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In Josef Bigun and Tomas Gustavsson, editors, *Image Analysis*, pages 363–370, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[7] Mouzhi Ge, Carla Delgado, and Dietmar Jannach. Beyond accuracy: Evaluating recommender systems by coverage and serendipity. pages 257–260, 01 2010.

[8] Emilia Gómez. Tonal description of polyphonic audio for music content processing. *INFORMS Journal on Computing*, 18(3):294–304, 2006.

[9] G.E. Hinton and R.R. Salakhutdinov. Reducing the dimensionality of data with neural networks. 313:504–7, 08 2006.

[10] Marius Kaminskas. Measuring surprise in recommender systems. 2014.

[11] Lior Rokach and Oded Maimon. *Data Mining With Decision Trees: Theory and Applications*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2nd edition, 2014.

[12] S. Streich and Perfecto Herrera. Detrended fluctuation analysis of music signals danceability estimation and further semantic characterization. In *AES 118th Convention*, 2005.

[13] Min Xu, Ling-Yu Duan, Jianfei Cai, Liang-Tien Chia, Changsheng Xu, and Qi Tian. Hmm-based audio keyword generation. In Kiyoharu Aizawa, Yuichi Nakamura, and Shin'ichi Satoh, editors, *Advances in Multimedia Information Processing - PCM 2004*, pages 566–574, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.