Compte Rendu de Projet

SAE821 – Gérer un projet

Rapport de présenation de la modélisation d'un système de recommandation respectant la vie privée

---

Pegliasco Matteo
Berge Enzo
Audouard Florian
Hermelin Lois
Master Informatique et Mathématiques
Université de Toulon, La Garde, Var, France

Table des matières

Introduction

Ce rapport technique est un comtpe rendu dans le cadre de la SAE821 dont l'objectif est la conception et la modélisation d'un système de recommandation respectueux de la vie privée des utilisateurs. Les systèmes de recommandation sont aujourd'hui omniprésents dans de nombreux domaines, notamment sur les plateformes de commerce en ligne, les réseaux sociaux ou encore les services de streaming. Ils permettent d'orienter les utilisateurs vers des contenus pertinents en se basant sur l'analyse de leurs comportements et préférences. Cependant, pour la majorité, ce traitement massif de données personnelles soulèvent des problématiques importantes liées à la protection de la vie privée, l'intégritée et la sécurité des données. Il est donc essentiel de réfléchir à des solutions techniques permettant de préserver la confidentialité des données tout en maintenant l'efficacité des recommandations. L'objectif de ce projet est de modéliser un système de recommandation de films prenant en compte tout ces critères, en utilisant des méthodes préservant la protection des données utilisateurs. Ce rapport présentera les différentes étapes de la modélisation, les choix techniques réalisés, ainsi que les méthodes mises en œuvre pour concilier personnalisation et respect de la vie privée.

1 Recherches théoriques

1.1 Systèmes de recommandation

1.1.1 Comprendre les Systèmes de Recommandation

Les systèmes de recommandation sont devenus des outils incontournables au fils du temps. Ils permettent aux utilisateurs d'être dirigé vers des contenus pertinents et personnalisés parmi une mutilitude de choix, en s'appuyant sur des données comportementales, des préférences et des connaissances du contexte. Cette recherche retrace l'évolution des approches de recommandation, de leurs fondements les plus simples à des méthodes d'intelligence artificielle avancée, en s'appuyant sur l'article : Delving Deeper into Recommender Systems: From Basics to State-of-the-Art.

Evolution Chronologique

— Années 1990 - Premiers Modèles :

Le Filtrage Collaboratif (CF) repose sur l'idée que des utilisateurs ayant patagé des préférences similaires dans le passé auront probablement des préférences semblables à l'avenir. Il existe deux variantes :

— **User-based CF :** recherche des utilisateurs similaires à l'utilisateur cible pour recommander les objets qu'ils ont aimés.

— **Item-based CF :** recherche des objets similaires à ceux déjà aimés par l'utilisateur pour proposer des recommandations

Filtrage basé sur le Contenue (CB) s'appuie uniquement sur les comportements passés de l'utilisateur, en particulier ses interactions avec certains objets. Contrairement au CF, il ne tient pas compte de la communauté mais des caractéristiques des éléments.

— Années 2000 - Hybridation et Deep Learning :

Modèles hybrides Ces modèles combinent CF et CB pour bénéficier de leurs avantages respectifs et une amélioration du démarrage à froid.

Début de l'apprentissage profond L'arrivée du deep learning a introduit une nouvelle manière de représenter utilisateurs et objets à l'aide de réseaux de neurones capables d'extraire des caractéristiques complexes, non linéaires et profondes.

— Années 2010 - Contexte et Connaissances :

Les systèmes de **context-aware** prennent maintenant en compte des informations telles que le temps, la localisation, le dispositif utilisé ou encore l'état émotionnel.

Les systèmes **knowledge-bases**, eux, utilisent des bases de connaissance et des bases de règles explicites pour mieux cerner les besoins utilisateurs, particulièrement utiles dans des cas de niches où les données sont rares.

— Années 2020 - Intelligence Renforcée et Explicabilité :

La Recommandation par Apprentissage par Renforcement (RL) permet une approche sequentille et interactive. Le système apprend à adapter ses recommandations en fonction des réactions de l'utilisateur. L'algorithme explore l'espace des solutions pour améliorer ses retours.

La Recommandation Explicable fournit une justification compréhensible à l'utilisateur, améliorant ainsi la confiance et la transparence

Types de Systèmes de recommandation :
— **Basé sur la mémoire** : Filtrage collaboratif simple
— **Basé sur un modèle** : Factorisation de matrice, réseaux de neurones, SVD
— **Basé sur le contenu** : utilisation des attributs des objets
— **Modèles hybrides** : Combine le filtrage collaboratif et filtrage basé sur le contenue
— **Apprentissage profond** : Réseaux de neurones
— **Système Contextuels** : Tiennent compte du contexte d'utilisation
— **Basé sur les connaissances** : Utilisent des logiques de règles, des bases de fait et ontologies
— **Apprentissage par renforcement** : Tiennent compte d'une interaction avec le client
— **Explicabilité** : Fournit une approche plus transparantes
— **Mutimodalité** : Combines plusieurs types de données
— **Protection de la vie Privée** : Utilise des techniques pour définir une forme de protection des résultats de requêtes

Méthodes Avancées :

1. **Filtrage Collaboratif avec ALS** (Alternating Least Squares) : Repose sur le fait que deux utilisateurs ayant eu des opinions similaires auront des opinions similaires ans d'autres situations. Utilise la factorisation de matrice avec une méthode d'optimisation par moindrers carrés alternés.
2. **Singular Value Decomposition** (SVD) :

   Utilisation d'une matrice de notes $R = (utilisateur * items)$, le SVD la décompose comme : $R \approx U.\sum.V^t$
   — U = matrice des facteurs latents utilisateur
   — $\sum$ = matrice diagonal des valeurs singulières
   — V^t = transposé de la matrice des facteurs latents des items

   SVD version recommandation :

   Factorisation de la Matrice $\hat{R}_{ui} = P_u^T Q_i$
   R_{ui} = note prédite par l'utilisateur u pour l'item i
   P_u = vecteur latent de l'utilisateur u (préférences)
   Q_i = note vecteur latent de l'item$i$ (caractéristiques)
   Le produit scalaire donne la note préditque de u donnerait à i.

   Apprentissage de P et Q :

$$\min_{P,Q} \sum_{(u,i)\in K} \left(R_{ui} - P_u^T Q_i\right)^2 + \lambda \left(\|P_u\|^2 + \|Q_i\|^2\right)$$

   Avec K l'ensemble des paires utilisateur-item connues (données d'entraînement), R_{ui} note réelle et λ le coefficient de régularisation
3. **Recurrent Neural Network** (RNN) :

   Le réseau de neurones permet de traiter des séquences (texte, audio, clics, utilisations ...) Il possède une mémoire interne (qui utilise l'info précédente à chaque étape) et gère les dépendances temporelles.

   À chaque instant (t), le RNN reçoit une entrée (x_t) et un état caché précédent (h_{t-1}), puis il calcule :
   — L'État caché : $h_t = \tanh(W_h h_{t-1} + W_x x_t + b)$
   — La Sortie : $y_t = \text{softmax}(W_y h_t + b_y)$
   Avec :
   - h_t = état caché (mémoire interne) à l'instant t
   - x_t = entrée à l'instant t
   - y_t = sortie prédite à l'instant t
   - W_h, W_x, W_y = matrices de poids
   - b, b_y = biais

- tanh = fonction d'activation
- softmax = utilisée pour une prédiction de classe

4. **ConvNet** (CNN) :
   Modèle de traitement spation de données (image, audio, vidéos) :
   — **Convolutions :** Extraction de caractéristiques locales
   — **Pooling :** Reduction dimensionnelle
   — **Fattening :** Prise de décision finale
5. **Multi-Armed Bandit :**
   Approche d'apprentissage par renforcement, visant à maximiser les gains et explorer l'espace d'action tout en exploitant les meilleures options (permet de recommander tout en testant de nouvelles options).

1.1.2 Filtrage collaboratif

Le filtrage collaboratif est une méthode de recommandation qui repose sur l'exploitation des interactions passées entre les utilisateurs et les objets. Contrairement au filtrage basé sur le contenu, il ne nécessite aucune information sur les caractéristiques des objets recommandés.

Fonctionnement

1. **Collecte de données** : Récupération des interactions entre utilisateurs et objets, sous forme explicite (notes, avis) ou implicite (clics, temps passé, etc.).
2. **Construction de la matrice utilisateur-objet** : Création d'une matrice où les lignes représentent les utilisateurs, les colonnes les objets, et les cellules contiennent les interactions.
3. **Mesure de similarité** : Calcul de la similarité entre utilisateurs (user-based) ou entre objets (item-based) à l'aide de fonctions telles que la similarité cosinus ou le coefficient de corrélation de Pearson.
4. **Prédiction** : Estimation de la préférence d'un utilisateur pour un objet non encore évalué, en se basant sur les préférences des utilisateurs ou objets similaires.
5. **Génération de recommandations** : Sélection des objets les plus susceptibles d'intéresser l'utilisateur cible.

Exemple illustratif Considérons la matrice suivante représentant les notes de films par trois utilisateurs :

	Film A	Film B	Film C	Film D
Utilisateur 1	5	?	4	2
Utilisateur 2	4	3	5	1
Utilisateur 3	2	4	1	5

Ici, le système peut prédire la note que l'utilisateur 1 donnerait au *Film B*, en se basant sur les préférences des utilisateurs 2 et 3, jugés similaires selon leurs comportements passés.

Avantages

— **Pas besoin de métadonnées** : Il peut fonctionner sans connaître les caractéristiques des objets à recommander.
— **Recommandations personnalisées** : Il permet de proposer des contenus pertinents et parfois inattendus grâce à l'analyse des préférences collectives.
— **Adaptabilité** : Le système évolue avec les comportements des utilisateurs, ce qui le rend dynamique.

Inconvénients

— **Problème du démarrage à froid (cold start)** : Difficile de faire des recommandations précises pour un nouvel utilisateur ou un nouvel objet sans historique d'interactions.
— **Sparsité des données** : Les matrices utilisateur-objet sont souvent très clairsemées, ce qui complique la détection de similarités fiables.
— **Dépendance au comportement collectif** : Il peut recommander des objets peu pertinents si les utilisateurs similaires ont des goûts trop différents ou incohérents.

Applications
— **Divertissement** : Netflix, Spotify pour recommander des films ou de la musique.
— **E-commerce** : Amazon pour proposer des produits selon les comportements d'achat d'utilisateurs similaires.
— **Éducation** : Recommandation de cours ou ressources pédagogiques.
— **Réseaux sociaux** : Suggestion de contacts, contenus ou groupes.

1.1.3 Factorisation de matrice

La factorisation de matrice [8] est une technique de recommandation user-based qui approxime les préférences des utilisateurs par le produit de deux matrices de plus faible dimension. Ces matrices représentent des caractéristiques latentes des utilisateurs et des items — ici, les films. Comme évoqué précédemment, cette approximation peut être obtenue à l'aide de l'algorithme des moindres carrés alternés ou de la descente de gradient stochastique . Cette méthode est particulièrement adaptée aux systèmes de recommandation car elle s'applique efficacement à de grands ensembles de données, gère bien la sparsité des notations, et peut être étendue pour intégrer des informations supplémentaires (comme des métadonnées ou du contenu).

Par exemple :

Considérons une matrice de notes R représentant les notes données par 2 utilisateurs à 4 films :

$$R = \begin{bmatrix} 5 & 3 & 0 & 1 \\ 4 & 0 & 0 & 1 \end{bmatrix}$$

où R_{ij} est la note donnée par l'utilisateur$i$ au film j, et représente une note manquante.
L'objectif de la factorisation est donc d'approcher R par le produit de deux matrices : $R \approx U \cdot V$. En partant de deux matrices de tailles arbittraire est en corrigeant les erreurs, on peut obtenir une approximation de la matrice de notes R comme :

$$R \approx \begin{bmatrix} 1.5 & 0.5 \\ 1.2 & 0.4 \end{bmatrix} \cdot \begin{bmatrix} 3.0 & 1.5 & 0.5 & 0.2 \\ 1.0 & 1.0 & 0.2 & 0.5 \end{bmatrix} = \begin{bmatrix} 5.00 & 2.75 & 0.85 & 0.55 \\ 4.00 & 2.20 & 0.68 & 0.44 \end{bmatrix}$$

Avec cette factorisation, la méthode prédit alors que l'utilisateur 1 donnerait une note proche de 0.85 au troisième film et l'utilisateur 2 donnerait une note provhe de 0,2.
Toutefois, cette approche présente certaines limites. Tout d'abord, elle necessite une étude profonde de ses hyperparamètres pour obtenir des résultats pertinents, notamment le nombre de facteurs latents à utiliser. De plus, elle est sensible au problème du démarrage à froid pour les nouveaux utilisateurs ou items -films dans le cas présent- et a un coût de calcul qui peut devenir important selon la taille du jeu de données.

1.2 Système de recommandation à froid

Au fil du temps, plusieurs approches ont été développées pour répondre aux problèmes de recommandation à froid, c'est-à-dire lorsqu'il s'agit de recommander de nouveaux films ou de faire des suggestions à de nouveaux utilisateurs. Parmi elles, on peut distinguer les principales suivantes :
— **Les méthodes basées sur le contenu (content-based)**, comme présenté par Schein, A. I., Popescul, A., Ungar, L. H., et Pennock, D. M. (2002) [2], utilisent les caractéristiques des items — ici, des films — comme la description, le genre ou les acteurs. En calculant la similarité entre les caractéristiques de films (par exemple avec la similarité cosinus), on peut recommander des films proches de ceux déjà appréciés par un utilisateur.
— **Les approches hybrides** combinent le filtrage collaboratif et le filtrage basé sur le contenu afin de bénéficier des avantages des deux. Par exemple et comme présenté par Xiaoyuan Su et Taghi M. Khoshgoftaar (2010) [4], plusieurs implémentation sont possible. Il est possible d'implémenter séparemment une méthode collaborative et une méthode basée sur le contenu, puis de combiner les résultats pour générer des recommandations plus pertinentes ou encore d'implémenter un modèle unifié, intégrant à la fois des caractéristiques du contenu et du filtrage collaboratif. L'ajout progressif d'interactions utilisateur permet ainsi d'améliorer la précision des recommandations.

— **Les méthodes basées sur le clustering** regroupent les utilisateurs ou les items selon leurs caractéristiques communes, comme présenté par Nguyen et al. (2024)[14] ou encore Snider (2019) [7]. L'objectif de cette approche est de générer des recommandations en s'appuyant sur la structure des groupes formés en recommendant, par exemple , un unique cluster de données pour limiter la diversité ou inversement en recommendant des films de plusieurs clusters pour la favoriser.

1.3 Confidentialité

1.3.1 Chiffrement homomorphe

Le chiffrement homomorphe est une technique de cryptographie qui permet d'effectuer des calculs sur des données chiffrées sans avoir besoin de les déchiffrer au préalable mais dont le résultat déchiffré conserve les opérations entre les données claires. Pour cela, plusieurs catégories de chiffrements existent :

— **Chiffrement partiellement homomorphe** : Il permet de réaliser des opérations sur des données chiffrées pour une loi spécifique, par exemple l'addition ou la multiplication entre des données chiffrées.
— **Chiffrement presque entièrement homomorphe** : Il permet de réaliser un nombres limité d'opérations sur des données chiffrées pour plusieurs lois spécifiques, par exemple l'addition et la multiplication entre des données chiffrées, à cause d'une accumulation de bruit.
— **Chiffrement entièrement homomorphe** : Il permet de réaliser un nombre illimité d'opérations sur des données chiffrées pour toutes les lois possibles, mais il est limité par la taille du bruit accumulé lors des opérations.

Dans le cadre de ce projet nous nous sommes intéressés particulièrement au chiffrement entièrement homomorphe pour sa flexibilité d'utilisation pour différents système de recommandations.
Ainsi, plusieurs approches existent pour la mise en place du chiffreemnt homomorphe :

— **Chiffrement DGHV avec bootstrapping [11]** : Le chiffrement DGHV [9] est un système de chiffrement homomorphe, opérant sur des entiers, basé sur le problème de la somme de sous-ensembles clairsemée et le problème du PGCD approché.
— **Chiffrement GSW avec bootstrapping[11]** : Le chiffrement GSW [10] est un système de chiffrement homomorphe, opérant sur des bits, basé sur le problème de la factorisation des entiers et le problème du logarithme discret.
— **Chiffrement TFHE** : Le chiffrement TFHE [13] est un système de chiffrement homomorphe, opérant sur des bits, basé sur la version torique du problème Learning With Errors.

Toutefois, même si si le chiffrement homomorphe permet de préserver la confidentialité des données, leurs utilisations présentent plusieurs contraintes :
— Les données chiffrées sont généralement plus volumineuses que les données claires, ce qui peut poser des problèmes de stockage et de transmission de données. Par exemple, dans l'article de I. Chillotti, N. Gama, M. Georgieva, et M. Izabachène. (2016) [6] où on apprend en conclusion de l'article que la taille du texte chiffré est 400 000 fois plus grand que celui d'origine.
— L'utilisation de "bootstrapping" [11] pour limiter l'accumulation de bruit peut augmententer le temps de calcul avec un nombre d'opérations significatifs réalisées. En effet, servant de contrôle de la taille
— du bruit, les opérations de "bootstrapping" augmentent les opérations réalisées sur le texte chiffré, pouvant ainsi influer sur le temps de calcul. augmentent aussi le temps de calcul.
— La mise en place sans ressouces externes est complexe et les ressources externes - sans prendre en compte leurs fiabilités - sont principalement disponible avec des bibliothèques en C++ ou RUST, comme Microsoft SEAL, HElib ou PALISADE, ce qui complique l'interaction avec d'autres langages de programmation comme JAVA.

1.3.2 Differential privacy

La confidentialité différentielle est une approche qui vise à garantir la protection de la vie privée des individus dans les ensembles de données en restreignant la relation entre les données de sortie et les données

de l'utilisateur. Pour cela, plusieurs approches existent :

- **Mécanisme exponentiel** : Le mécanisme exponentiel [12] permet de sélectionner aléatoirement un élément parmi un ensemble de données, en favorisant les éléments ayant le plus de poids selon une fonction d'évaluation donnée.
- **Réponse aléatoire** : Le mécanisme de réponse aléatoire telle qu'utilisé dans le bulletin de psychologie écrit par de BEGIN G. et SAVARD F. [1] vise à brouiller les réponses données en remplaçant aléatoirement certaines réponses par des valeurs aléatoires. En connaissant la probabilité d'obtenir une valeur aléatoire, il est alors possible d'estimer la proportion de données réelles ainsi que la variance au sein des données.
- ***K-anonymisation* et *L-diversité*** : La *K-anonymisation*[3] et la *L-diversité* consistent à masquer les éléments de quasi-identification d'un individu en le regroupant avec $K-1$ autres individus dans une représentations plus large, tout en conservant la visibilité des L informations spécifiques à chaque utilisateurs et destinées à être partagées.
- **Mécanisme de Laplace ou Gaussien** : Le mécanisme de Laplace ou Gaussien [5] consiste à altérer les données des utilisateurs pour masquer les données réelles tout en conservant la répartition moyenne des données en ajoutant un bruit aléatoire.

2 Démarche choisie

2.1 Motivations du choix

2.1.1 Confidentialité des données

Le cadre de ce projet est de mettre en place un système de recommandation de films respectant la vie privée des utilisateurs. Nous avons considéré que la base de données des utilisateurs ainsi que les films proposés pouvaient évoluer au fil du temps. Pour la protection de la vie privée des utilisateurs, nous avions considéré l'utilisation du chiffrement homomorphe TFHE, mais la complexité de son implémentation et de son utilisation, avec notamment l'interaction entre différents langages de programmation (JAVA, C++, RUST), ainsi que la complexité de l'utilisation de l'apprentissage machine avec une bibliothèque spécialisée, nous a poussés à nous intéresser davantage à la confidentialité différentielle.
Plus précisément, nous nous sommes particulièrement intéressés à la mise en place du mécanisme de bruitage et du mécanisme exponentiel pour la mise en place de la confidentialité différentielle pour, respectivement, les recommandations pour un nouvel utilisateur et les recommandations personnalisées d'un utilisateur ayant déjà évalué une certaine quantité de films. En ce qui concerne le mécanisme de génération du bruit, nous avons choisi d'utiliser le mécanisme de Laplace pour masquer les données des utilisateurs. Ce choix est motivé par le contexte de ce projet. En effet, les utilisateurs peuvent modifier leur evaluation d'un film. Dans cet optique, et contrairement au mécanisme gaussien, nous avons favorisé le mécanisme de Laplace car c'est un mécanisme ϵ-indistinguable. C'est-à-dire qu'il garantit qu'une modification d'une entrée dans une base de données crée une modification bornée de la distribution de probabilité. De plus, étant donné que la base de données initale possède relativement peu d'évaluation utilisateurs pour chaque film, le mécanisme de Laplace est privilégié puisque la génération de bruit à forte valeur est moins probable que pour le mécanisme gaussien, dû à une distribution plus centré en 0 pour la courbe Laplacienne, ce qui permet de garantir une meilleure précision dans l'étude de la moyenne des évaluations des films.
Enfin, dans le cadre d'une protection des données privée des utilisateurs, nous avons eu la volonté de limité l'utilisation de ressources externes pour la génération des données bruitées afin de reposer au minimum sur la fiabilité de ces ressources et avoir un maximum de contrôle sur la génération de bruit (c.f 2.1.2 Analyse du bruit Laplacien). Ce faisant, l'implémentation du mécanisme Gaussien impliquerait l'implémentation complexe, avec le temps disponible, de l'approximation de l'inverse de la fonction :

$$F(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt \tag{1}$$

afin de générer du bruit. Ainsi, nous avons privilégié l'implémentation plus simple du mécanisme de Laplace qui s'adapte mieux aux traitements ponctuels sur des bases de données de taille variable, ccorrespondant aux besoins de notre système d'entrainement pour nos recommandations. En complément, et comme présenté précédemment, nous avons considéré l'utilisation du mécanisme exponentiel pour une

sélection aléatoire biaisée d'un film recommandé. Cette approche permet à la fois de favoriser certains films - ici ceux qui ont une note moyenne la plus éloigné de 2.5 pour recommender des films qui n'ont jamais été noté ou des films avec une bonne notation - grâce à une fonction d'utilité définie dans le système, tout en limitant le risque que l'utilisateur puisse extraire des informations individuelles sur les autres utilisateurs ayant permis de générer les recommandations avec une bonne calibration.

2.1.2 Analyse du bruit Laplacien

Comme présenté plus tôt, la mise en place manuel du mécanisme de Laplace pour la génération de bruit permet un meilleure contrôle sur la génération du bruit. Cette section vise à présenter les implications du choix des paramètres du mécanisme de Laplace pour la génération de bruit.
La génération du bruit de Laplace est basée sur la distribution de Laplace. Plus précisement, et comme mise en place dans le système en posant b le budget de confiancialité, la génération du bruit de Laplace est définie par :

$$Laplace(b) = b \cdot \text{sign}(x) \cdot \ln(1 - 2|x|) \tag{2}$$

, où x est un nombre aléatoire tiré de la distribution uniforme sur l'intervalle $]-0.5, 0.5[$.
A partir de cette écriture, il est alors possible de définir la variance du bruit généré par le mécanisme de Laplace ainsi que de choisir le budget de confidentialité b pour la génération du bruit avec que la probabilité de génération de bruit supérieur à une certaine limite L soit inférieur à un certains seuil p.

Propriété 1 :
Soit $b > 0$.
La variance fonctionnelle du bruit généré par le mécanisme de Laplace est donnée par : $2b^2$.

Démonstration :
Plaçons nous dans les conditions de l'énoncé.
La variance fontionnelle de la fonction $\phi : x \mapsto Laplace(b)$, noté $V(\phi)$, est donnée par : $\int_{-0.5}^{0.5}(Laplace(b) - \overline{Laplace})^2 dx$, où $\overline{Laplace}$ est la moyenne de la fonction ϕ sur l'intervalle $]-0.5, 0.5[$.
La fonction ϕ étant antisymétrique, on a alors que $V(\phi) = \int_{-0.5}^{0.5} Laplace(b)^2 dx = \int_{-0.5}^{0.5} b^2 \ln(1 - 2|x|)^2$.

Par symétrie de la fonction $x \mapsto \ln(1 - 2|x|)$, on a que : $V(\phi) = 2b^2 \int_{-0.5}^{0} \ln(1 + 2x)^2 dx$.
En posant $X = 1 + 2x$, on a alors que : $V(\phi) = 2b^2 \int_0^1 \frac{\ln(X)^2}{2} dX$.
En utilisant l'intégration par parties, on a alors que : $V(\phi) = b^2([X \ln(X)]_0^1 - 2[X(\ln(X) - 1)]_0^1)$.
Par croissance comparée , on a alors que : $V(\phi) = 2b^2$.

Propriété 2 :
Soient $p \in]0, 1[$, $L > 0$ et $b > 0$.
$P(|Laplace(b)| > L) \leq p$ si et seulement si $b \leq \frac{-L}{\ln(p)}$.

Démonstration :
Plaçons nous dans les condition de l'énoncé.
Sachant que l'évènement $(|Laplace(b)| > L)$ est équivalent à $(Laplace(b) < -L) \cup (Laplace(b) > L)$, on a alors que : $P(|Laplace(b)| > L) = P(Laplace(b) < -L) + P(Laplace(b) > L)$ car les évènements sont disjoints.

Comme $b > 0$ par hypothèse et que $x \mapsto \ln(1 - 2|x|)$ est négative sur l'intervalle $]-0.5, 0.5[$, on a alors que : $(Laplace(b) > L) \Leftrightarrow (-b \ln(1 + 2x) > L)$ et $(Laplace(b) < -L) \Leftrightarrow (b \ln(1 - 2x) < -L)$.
Ainsi : $(Laplace(b) > L) \Leftrightarrow (-b \ln(1 + 2x) > L) \Leftrightarrow (x < \frac{e^{-L/b} - 1}{2})$ avec $x \in]-0.5, 0[$
Et : $(Laplace(b) < -L) \Leftrightarrow (b \ln(1 - 2x) < -L) \Leftrightarrow (x > \frac{1 - e^{-L/b}}{2})$ avec $x \in]0, 0.5[$.

Par hypothèse, comme x suit une une loi uniforme sur l'intervalle $]-0.5, 0.5[$, on a alors que :

$$P(Laplace(b) > L) = \frac{e^{-L/b} - 1}{2} - (-0.5) = \frac{e^{-L/b}}{2}$$
$$P(Laplace(b) < -L) = 0.5 - \frac{1 - e^{-L/b}}{2} = \frac{e^{-L/b}}{2}$$

Et donc : $P(|Laplace(b)| > L) = e^{\frac{-L}{b}}$.
On cherche b tel que $P(|Laplace(b)| > L) \leq p$ et on a alors que : $e^{\frac{-L}{b}} \leq p \Leftrightarrow b \leq \frac{-L}{\ln(p)}$.

Ainsi, avec une implementation manuelle du mécanisme de Laplace, il est alors possible de manipuler le budget confidentiel b pour contrôler la génèration de bruit tout en maximisant la variance du bruit généré pour augmenter la confidentialité des données. Pour cela, et en utilisant les deux propriétés précédentes, le choix optimale du budget de confidentialité b pour la génération du bruit est : $b = \frac{-L}{\ln(p)}$ avec L la limite de bruit maximal souhaitée et p le seuil de probabilité de génération de bruit supérieur à L.

2.1.3 Models de recommandations

Pour le système de recommendation, nous savons choisi de mettre en place deux approches pour la recommandation de films : un systeme se basant sur le SVD pour un individus connu par le système - individu ayant déjà noté un certain nombre de films - et un autre système de recommendation basé sur la clustering non supervisé pour la recommendation à froid pour un nouvel utilisateur.

— **Pour le SVD :**

Le système de recommandation à chaud mis en place repose sur une approche collaborative fondée sur la décomposition en valeurs singulières (SVD). Inspirée des travaux classiques en filtrage collaboratif (notamment ceux de Sarwar et al., 2000), cette méthode permet de prédire les préférences d'un utilisateur connu par le système, c'est-à-dire un utilisateur ayant déjà noté un certain nombre de films.

1. Construction de la matrice d'interactions : une matrice utilisateur–film est construite à partir des notes données par les utilisateurs. Chaque ligne représente un utilisateur et chaque colonne un film, les entrées correspondant aux notes attribuées.
2. Factorisation de la matrice : cette matrice est ensuite factorisée en trois sous-matrices :

$$R \approx U \Sigma V^T$$

   — U représente les facteurs latents des utilisateurs,
   — V les facteurs latents des films,
   — Σ une matrice diagonale contenant les valeurs singulières.
   Cette décomposition permet de capturer des facteurs implicites (préférences, styles, affinités thématiques) non directement observables dans les données brutes.
3. Prédiction des notes : une fois les vecteurs latents appris, le système peut estimer la note qu'un utilisateur donnerait à un film non encore vu, en effectuant le produit scalaire entre les vecteurs correspondants. Les films obtenant les meilleures prédictions peuvent alors être recommandés.

L'objectif de ce système de recommandation est donc, à la fois, de proposer des suggestions personnalisées aux utilisateurs connus du système, en exploitant les préférences implicites issues des notations passées, de maximiser la pertinence des recommandations tout en réduisant la redondance, et de tirer parti des corrélations latentes entre utilisateurs et films pour améliorer la qualité globale des prédictions.

— Pour le système de clustering :

Le système de recommandation à froid mis en place se base sur une recommendation item based via un système de clustering, à l'image de Snider 2019 [7] et de la sélection daléatoire de films. Plus précisement, Le système de clustering Pour ce faire, l'algotirthme de recommendation se compose en trois étapes principales :

1. Sélection des films pouvant être recommander selon certains critières. Dans notre implémentation, les films recommendables sont les films ayant une note moyenne supérieure ou égale à un seuil (ici, 3) ou ayant un nombre d'évaluations inférieur ou égal à un seuil (ici, 0). Cette approche permet alors de recommander des films étant globalement appréciés par les utilisateurs l'ayant

noté et donc de recommander des films qui pourront plaire à l'utilisateur mais aussi de recommander des films peu notés pour obtenir plus d'informations statistiques sur ces films tout en favorisant la diversité des films recommandés.

2. Clustering des films : Une fois nos films sélectionnés, des vecteurs de caractéristiques sont générés à partir de leurs informations descriptives. Ici, nous avons utiliser les genres et le synopsis des films ainsi que leurs dates de sortie (année, jour et mois) pour générer un vecteur de caractéristiques. Pour générer le vecteur de caractéristiques des genres, respectivement du synopsis, dans le serveur python, nous avons utiliser la fonction "MultiLabelBinarizer" de la bibliothèque "sklearn", respectivement la fonction "SentenceTransformer" de la bibliothèque python "sentence_transformers".
   Après la normalisation de ces vecteurs, une réduction de dimension est appliquée (via l'algorithme UMAP) suivie d'un clustering non supervisé (DBSCAN) comme présenté dans ..... . L'utilisation de l'algorithme DNBSCAN permet ainsi de regrouper les films selon leurs similarité - ici, la similarité cosinus - tout en offrant une meilleure adapation à l'évolution des films dans la base de données que des clusters supervisé.
3. Sélection aléatoire de films : Afin de favoriser la diversité des films recommandés, le système sélectionne aléatoirement un cluster, puis, de manière biaisée pour favoriser les films avec la moyenne la plus faible (représentant les films sans évaluation), un film au sein de celui-ci. Pour ce faire, nous avons utilisé une fonction d'utilité définie dans le système pour favoriser les films ayant une note moyenne la plus proche de 0, qui après le tri de l'étape 1, représente les films les moins notés.

L'objectif de ce système de recommendation vise donc, à la fois, de représenter la diversité des films dans la base de données, limiter la redondance des recommendations avec les choix aléatoires de clusters et de films, mais aussi de recommander des films ayant un certains intêret, ici les films appréciés en moyenne par les utilisateurs ou les films peu noté (car possiblement récent ou peu vu).

2.2 Implémentation dans le système

Ainsi pour implémenter ces différents élèments, nous avons structuré notre projet de la manière suivante :

— Un serveur de base de données, implémenté en JAVA, dédié au stockage des informations des films ainsi que les informations relatives au utilisateurs. Dans un optique de eprformance dans le cadre d'une utilisation en production, ce serveur est relié à une couche de recherche rapide, elle aussi implémentée en JAVA, qui permet d'augmenter la rapidité de réponse du système lors des recherches.
— Un serveur de recommandation, implémenté en python, dédié aux recommendations à chaud et à froid. Il est relié au serveur de base de données pour récupérer les informations nécessaires à la génération des recommandations via un transfert de fichiers CSV.

Plus précisement,

front communique avec une base de données base de données avec une couche de recherche rapide pour augmenter la rapidité de réponse du système. connexion de la base de donnée avec le serveur de recommendation lors de l'envoie des données les notes utilisateurs sont chiffrées avec le mécanisme de Laplace en fonction du nombre de notation utilisateur, on utilise un systeme spécifique lors du renvoie,

3 Résultats expérimentaux

3.1 Précision des recommandations et performance computationnelles

3.1.1 Système de recommandation à chaud

3.1.2 Système de recommandation à froid

Pour les systèmes de recommandation à froid, les notions de précision et de rappel ne sont pas adaptées, car le système ne dispose pas d'informations sur les préférences des utilisateurs. Il ne peut donc pas prédire quels films seraient pertinents pour eux. Notre objectif avec ce type de système se centre principalement autour de la représentation la diversité des films présents dans la base de données, tout en limitant la redondance des recommandations et en mettant en avant des films peu notés ou bien évalués.
Ainsi, nous avons choisi d'évaluer la qualité des recommandations à l'aide des métriques suivantes :

— Le nombre de clusters différents présents dans les recommandations pour un utilisateur.
— Le nombre de films distincts recommandés par utilisateur.
— Le pourcentage de films sans évaluation (i.e., ayant 0 note) qui sont recommandés.
— La diversité des genres des films recommandés.

Par ailleurs, afin de comparer les performances des systèmes de recommandation à froid selon la méthode de réduction de dimension utilisée, nous avons également analysé les temps d'initialisation, d'entraînement et de réponse du système. Pour effectuer ces mesures — visibles dans les tableaux « Résultats expérimentaux du système de recommandation à froid sans bruit » (Tableau 1) et « Résultats expérimentaux du système de recommandation à froid avec bruit » (Tableau 2) — nous avons utilisé une même graine aléatoire pour la génération des clusters et effectué 609 tests pour chaque configuration de réduction de dimension avec une demande de 20 films à recommender. Ces deux tableaux présentent les métriques suivantes :

— La dimension des vecteurs après réduction de dimension : **k**
— Le nombre de clusters différents présents dans les recommandations pour un utilisateur : **Clusters**
— Le ratio des genres recommandés par rapport à ceux présents dans la base de données : **Genre Div.**
— Le ratio des genres recommandés par rapport à ceux effectivement recommandables : **Inner Div.**
— Le ratio de films peu notés (ayant 0 note) parmi les films recommandés : **0 Cov.**
— Le taux de couverture des recommandations sur les films éligibles : **Cov.**
— Le nombre moyen de films distincts recommandés par utilisateur : **Mov./User**
— Le nombre moyen de clusters distincts recommandés par utilisateur : **Clust/User**
— Le nombre moyen de films recommandés ayant peu de notes : **0 Mov./Reco**
— Le temps de réponse moyen du système de recommandation : **Resp. Time**
— Le temps moyen d'initialisation du système : **Init.**
— Le temps moyen d'entraînement du système : **Train.**

À partir des deux tableaux de résultats, nous constatons que les performances du système de recommandation à froid sont relativement stables entre les données bruitées, avec un budget de confidentialité fixé à $b = 0.5 \approx \frac{-1}{\ln(0.1)}$, et non bruitées. Dans la suite, nous nous concentrons sur les résultats obtenus avec bruit, car ce sont les données qui sont utilisées dans le système pour les recommandations.
Le tableau « Résultats expérimentaux du système de recommandation à froid avec bruit » (Tableau 2) confirme la stabilité des performances du système, comme en témoignent les faibles variances des métriques analysées. Globalement, le système montre une certaine redondance dans ses recommandations, avec un nombre moyen de films distincts recommandés d'environ 2 par utilisateur pour 20 films. En revanche, il parvient à une diversité efficace de genres, avec une couverture moyenne d'environ 60% des genres présents dans la base de données. La représentation des clusters est également satisfaisante, avec en moyenne 9 à 11 clusters différents représentés dans les recommandations, soit une couverture proche des 10 clusters extraits lors du clustering.
Par ailleurs, environ 23% des films recommandés n'ont reçu aucune évaluation, ce qui équivaut à environ 4 films peu notés par recommandation, et montre donc la mise en avant de contenus peu explorés.
Enfin, en termes de performance système, le temps de réponse et le temps d'initialisation restent très raisonnables. Le temps d'entraînement, bien que plus variable, s'élève en moyenne à 2 minutes 30, ce qui

reste acceptable dans un cadre expérimental. Néanmoins, ce temps dépend fortement de la taille de la base de données et du nombre de notations, et peut devenir plus coûteux à l'échelle.

3.2 Limites et perspectives

3.2.1 Système de recommendation à chaud

3.2.2 Système de recommendation à froid

Le système de recommandation à froid mis en place dans ce projet présente plusieurs limites. Tout d'abord, il repose sur la sélection aléatoire de films au sein des clusters. Ainsi, lorsque le nombre de films à recommander augmente, la probabilité que certains clusters ne soient pas sélectionnés diminue.
Pour répondre aux demandes de recommandations en petit nombre tout en assurant une certaine diversité, il serait pertinent de sélectionner un film par cluster lors des premières suggestions, en suivant des règles spécifiques. Par exemple, on pourrait privilégier les clusters les plus importants. Cela permettrait de garantir une représentation minimale de l'ensemble de la base de données tout en assurant la diversité des recommandations.

Ensuite, comme mentionné précédemment, le système de recommandation à froid repose sur une représentation vectorielle des caractéristiques des films. Toutefois, afin de mieux adapter cette représentation à une base de données spécifique d'utilisateurs, il serait intéressant d'y intégrer des statistiques globales telles que le nombre d'évaluations, une approximation de la note moyenne et la variance des notes pour chaque film. Cette approche pourrait enrichir la représentation des films en tenant compte de leur réception par les utilisateurs, et ainsi permettre une recommandation plus pertinente.

Enfin, d'approfondir l'analyse des performances du système de recommandation à froid en augmentant le budget de confidentialité pourrait permettre de mieux évaluer l'impact du bruit sur la diversité des recommandations ainsi que sur la précision des recommandations.

3.2.3 Mécanisme de Laplace

Comme nous l'avons vu dans la section 2.1.2, il est possible de contrôler la génération du bruit dans le mécanisme de Laplace. Toutefois, celle-ci est indépendante du nombre de données à protéger pour un film donné. Un bruit important tend à avantager les films ayant un grand nombre d'évaluations, tout en défavorisant ceux qui en ont peu, rendant alors difficile l'extraction d'informations statistiques pertinentes. Par exemple, en observant l'écart moyen du bruit généré par l'algorithme 1 (présenté en annexe), avec un budget de confidentialité $b = \frac{-1}{\ln(0.9)}$, on peut obtenir environ 6,5 points d'écart par rapport à zéro pour 10 évaluations, contre seulement -0,1 pour 10 000 évaluations. Au contraire, en prenant $b = \frac{-0.5}{\ln(0.2)}$, on peut obtenir environ 0.5 points d'écart par rapport à zéro pour 10 évaluations, contre seulement -0.0005 pour 10 000 évaluations
Dans ce projet, notre approche repose sur l'utilisation d'un budget de confidentialité faible, afin de garantir une étude statistique fidèle aux données réelles, quel que soit le nombre d'évaluations d'un film. Cette stratégie est adaptée à une base de données avec peu d'évaluations, comme MovieLens, où le nombre maximal de notes pour un film est de 329. En revanche, dans une base comportant un volume beaucoup plus important, un bruit trop faible pourrait permettre une extraction probabiliste des données sensibles.

Dans cette optique, il serait intéressant d'envisager la mise en place d'un budget de confidentialité va-

riable, ajusté en fonction du nombre d'évaluations par film et de regarder la compatibilité d'une telle approche avec les systèmes de recommandation.

3.2.4 Mécanisme exponentiel

Dans ce projet, le mécanisme exponentiel a été mis en place afin d'ajouter un niveau de confidentialité côté interface utilisateur, limitant ainsi le risque que les habitudes de consommation d'autres utilisateurs puissent être déduites. Toutefois, l'implémentation de ce mécanisme augmente le temps de génération des recommandations, car il est nécessaire de recalculer les probabilités de sélection pour chaque film à chaque itération.

Ainsi, il serait intéressant d'étudier des optimisations du mécanisme exponentiel permettant de réduire ce temps de calcul tout en conservant les garanties de confidentialité. Une autre piste serait l'implémentation d'un système de préchargement des recommandations, afin de limiter le temps de réponse côté interface utilisateur.

3.3 Résumé

Références

[1] J.-P. Rolland, « La technique de la réponse aléatoire : un moyen de contrôler la désirabilité sociale dans la mesure de l'estime de soi », *Bulletin de Psychologie*, vol. 33, no. 343, 1979. [En ligne]. Disponible : https://www.persee.fr/doc/bupsy_0007-4403_1979_num_33_343_1128 [Consulté le 28 mai 2025].

[2] A. I. Schein, A. Popescul, L. H. Ungar, et D. M. Pennock, « Methods and Metrics for Cold-Start Recommendations », in *Proc. 25th Annu. Int. ACM SIGIR Conf. Res. Dev. Inf. Retr. (SIGIR '02)*, 2002, pp. 253–260. [En ligne]. Disponible : https://www.researchgate.net/publication/221300490_Methods_and_Metrics_for_Cold-Start_Recommendations [Consulté le 28 mai 2025].

[3] Wikipedia, « K-anonymisation », 2006. [En ligne]. Disponible : https://fr.wikipedia.org/wiki/K-anonymisation [Consulté le 28 mai 2025].

[4] X. Su et T. M. Khoshgoftaar, « A Survey of Collaborative Filtering Techniques », *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 1, pp. 107–117, 2010. [En ligne]. Disponible : https://pages.stern.nyu.edu/~atuzhili/pdf/TKDE-Paper-as-Printed.pdf [Consulté le 28 mai 2025].

[5] C. Dwork et A. Roth, *The Algorithmic Foundations of Differential Privacy*, Found. Trends Theor. Comput. Sci., vol. 9, no. 3–4, pp. 211–407, 2014. [En ligne]. Disponible : https://www.cis.upenn.edu/~aaroth/Papers/privacybook.pdf [Consulté le 28 mai 2025].

[6] I. Chillotti, N. Gama, M. Georgieva, et M. Izabachène, « Faster Fully Homomorphic Encryption : Bootstrapping in Less Than 0.1 Seconds », *Cryptology ePrint Archive*, Rapport 2016/870, 2016. [En ligne]. Disponible : https://eprint.iacr.org/2016/870 [Consulté le 28 mai 2025].

[7] G. Snider, « Unsupervised Product Clustering : Exploring the Cold Start Problem », *SSENSE Tech Blog*, Medium, 2019. [En ligne]. Disponible : https://medium.com/ssense-tech/unsupervised-product-clustering-exploring-the-cold-start-problem-8053ef04bac9 [Consulté le 28 mai 2025].

[8] E. Azar, « Deep Dive into Matrix Factorization for Recommender Systems : From Basics to Implementation », Medium, 2020. [En ligne]. Disponible : https://medium.com/@eliasah/deep-dive-into-matrix-factorization-for-recommender-systems-from-basics-to-implementation-79e4f [Consulté le 28 mai 2025].

[9] E. Goblé et R. Gatelier, « Le chiffrement DGHV », Université de Technologie de Compiègne. [En ligne]. Disponible : https://www.utc.fr/~wschon/sr06/tx_chiffrement_homomorphe/pages/dghv.html [Consulté le 28 mai 2025].

[10] E. Goblé et R. Gatelier, « GSW », Université de Technologie de Compiègne. [En ligne]. Disponible : https://www.utc.fr/~wschon/sr06/tx_chiffrement_homomorphe/pages/vecteurs_propres_approx.html [Consulté le 28 mai 2025].

[11] E. Goblé et R. Gatelier, « Le Bootstrap », Université de Technologie de Compiègne. [En ligne]. Disponible : https://www.utc.fr/~wschon/sr06/tx_chiffrement_homomorphe/pages/bootstrapping.html [Consulté le 28 mai 2025].

[12] *Programming Differential Privacy*, « Chapter 9 — Fully Homomorphic Encryption (FHE) ». [En ligne]. Disponible : https://programming-dp.com/ch9.html [Consulté le 28 mai 2025].

[13] B. Chandran, A. Ghoshal, S. Halevi, S. Ranellucci, et N. P. Smart, « Faster Homomorphic Comparison Operations for BGV and BFV », *Cryptology ePrint Archive*, Rapport 2023/958, 2023. [En ligne]. Disponible : https://eprint.iacr.org/2023/958 [Consulté le 28 mai 2025].

[14] Q. Nguyen, M. Naghiaei, et Y. Zhang, « Cold-start Recommendation by Personalized Embedding Region Elicitation », *arXiv preprint* arXiv :2406.00973, 2024. [En ligne]. Disponible : https://arxiv.org/abs/2406.00973 [Consulté le 28 mai 2025].

Annexes

Listing 1 – Fonction de génération de bruit Laplacien

```
def sign(x):
    if x >= 0:
        return 1
    elif x < 0:
        return -1

def laplace(b):
    x = random.uniform(-0.5, 0.5)
    return b * sign(x) * math.log(1 - 2 * abs(x))

def average_noise(b, n_test):
    total_noise = 0
    for _ in range(n_test):
        total_noise += laplace(b)
    return total_noise / n_test
```

TABLE 1 – Résultats expérimentaux du système de recommandation à froid **sans bruit** ($b = 0.5$, 20 films recommandés, 609 tests par configuration)

k	Clusters	Genre Div.	Inner Div.	0 Cov.	Cov.	Mov./User	Clust/User	0 Mov./Reco	Resp. Time (s)	Init (s)	Train (s)
sans reduction	2	0.456	0.456	0.0025	0.703	8.057	2.0	0.044	0.0267	0.032	94.74
50	11	0.602	0.602	0.0104	0.206	2.363	9.422	0.187	0.0065	0.030	157.94
100	10	0.602	0.602	0.0117	0.215	2.471	8.754	0.210	0.0071	0.016	157.40
150	10	0.609	0.609	0.0130	0.230	2.634	8.874	0.235	0.0069	0.027	180.02
200	12	0.575	0.575	0.0115	0.213	2.445	9.903	0.207	0.0075	0.018	189.72
250	11	0.607	0.607	0.0120	0.230	2.634	9.368	0.217	0.0056	0.016	187.13
300	11	0.565	0.565	0.0109	0.221	2.534	9.386	0.197	0.0054	0.015	201.89
350	11	0.589	0.589	0.0074	0.189	2.163	9.455	0.133	0.0053	0.016	204.99
400	10	0.578	0.578	0.0133	0.209	2.402	8.824	0.240	0.0071	0.018	223.61
Moyenne	10.75	0.591	0.591	0.0113	0.214	2.456	9.248	0.190	0.0064	0.019	187.83
Variance	0.73	0.002	0.002	0.00004	0.0003	0.036	0.38	0.0011	0.0000007	0.00003	569.77

TABLE 2 – Résultats expérimentaux du système de recommandation à froid **avec bruit** ($b = 0.5$, 20 films recommandés, 609 tests par configuration)

k	Clusters	Genre Div.	Inner Div.	0 Cov.	Cov.	Mov./User	Clust/User	0 Mov./Reco	Resp. Time (s)	Init (s)	Train (s)
sans reduction	2	0.466	0.466	0.0018	0.762	7.936	2.0	0.033	0.0325	0.031	100.69
50	10	0.603	0.603	0.0140	0.245	2.550	8.811	0.251	0.0042	0.014	111.17
100	10	0.613	0.613	0.0119	0.248	2.583	8.824	0.213	0.0051	0.019	124.10
150	10	0.604	0.604	0.0146	0.247	2.570	8.810	0.263	0.0050	0.014	125.55
200	9	0.585	0.585	0.0136	0.256	2.667	8.209	0.245	0.0052	0.014	132.46
250	10	0.598	0.598	0.0109	0.237	2.468	8.796	0.195	0.0055	0.013	170.78
300	9	0.585	0.585	0.0135	0.245	2.552	8.141	0.243	0.0056	0.014	160.96
350	12	0.607	0.607	0.0100	0.222	2.314	9.888	0.181	0.0042	0.013	166.62
400	11	0.593	0.593	0.0121	0.218	2.271	9.450	0.218	0.0049	0.013	179.57
Moyenne	10.13	0.598	0.598	0.0128	0.240	2.497	8.854	0.226	0.0050	0.014	146.90
Variance	0.99	0.001	0.001	0.000002	0.0002	0.021	0.38	0.0008	0.0000002	0.000004	682.13