Sujet de projet : Développement d'un moteur de recommandations respectueux de la vie privée

Contexte

Un moteur de recommandation est un système qui analyse des données pour suggérer des éléments pertinents (comme des produits, des films, de la musique ou des contenus) à un utilisateur, en se basant sur ses préférences, son historique, ou les comportements d'autres utilisateurs similaires. Les systèmes de recommandation sont omniprésents, utilisés dans les plateformes de streaming, les sites e-commerce ou encore les réseaux sociaux. Cependant, ces systèmes collectent et analysent d'importantes quantités de données utilisateur, soulevant des préoccupations quant à la confidentialité.

Dans ce projet, vous développerez un moteur de recommandations simple mais performant tout en essayant de garantir la protection des données utilisateur. Vous devrez choisir entre deux approches avancées de préservation de la vie privée :

1. **La Differential Privacy (DP)**, qui ajoute du bruit mathématiquement contrôlé aux données ou aux modèles.
2. **La cryptographie homomorphe**, permettant de réaliser des calculs sur des données chiffrées.

Le dataset Movie Lens https://grouplens.org/datasets/movielens/ disponible en différentes tailles propose des évaluations de films. Un moteur de recommandation basé sur le dataset MovieLens peut suggérer des films à un utilisateur en fonction de ses notes précédentes et de celles d'autres utilisateurs ayant des goûts similaires. On parle alors de filtrage collaboratif. L'autre approche **qui n'est pas demandée** ici est le filtrage par le contenu, on suggère du contenu proche de celui qui a été vu et apprécié (des approches hybrides existent).

En parallèle, ce projet inclura une dimension complète de **modélisation UML**, un développement en **architectures microservices**, et une gestion agile du projet.

Objectifs

1. **Créer un moteur de recommandations** s'appuyant sur le filtrage collaboratif en utilisant des techniques modernes de machine learning.
2. **Étudier voire mettre en œuvre une approche de protection de la vie privée**, au choix :
    - Differential Privacy, ou
    - Cryptographie homomorphe.

3. **Concevoir une architecture microservices**, incluant un serveur Java JPA/REST/WebSocket et une composante IA en Python.
4. **Réaliser une analyse, une conception et une modélisation UML**, incluant une gestion agile des tâches.
5. **Comparer les résultats de votre implémentation** (performance, précision, protection de la vie privée) avec des solutions standards et discuter des compromis réalisés.

Adaptations possibles

En cas de motivation particulière d'un groupe des adaptations au sujet sont possibles par exemple en utilisant d'autres techniques de protection de la vie privée (federated learning par exemple) ou en l'appliquant à d'autres types de données (images par exemple).

Travail demandé

1. Recherche théorique

1.1. État de l'art sur les systèmes de recommandation

- Étudier les approches classiques (filtrage collaboratif, factorisation matricielle, deep learning, etc.).
- Analyser l'impact de la confidentialité sur les recommandations.
- La wikipedia et cet article d'introduction sur medium sont de bons points de départ
    - https://en.wikipedia.org/wiki/Collaborative_filtering
    - https://medium.com/@eliasah/delving-deeper-into-recommender-systems-from-basics-to-state-of-the-art-d92ee8e277f2
    - https://medium.com/@eliasah/deep-dive-into-matrix-factorization-for-recommender-systems-from-basics-to-implementation-79e4f1ea1660

1.2. État de l'art sur les techniques de protection de la vie privée

- Faites des recherche et présenter les principes de la **Differential Privacy** :
    - Applications courantes.
    - Méthodes clés (ajout de bruit, gradients bruités, Local Differential Privacy, etc.).
- Présenter les principes de la **cryptographie homomorphe** :
    - Cryptographie homomorphe partielle et totale.
    - Bibliothèques courantes.
    - Limites (temps de calcul, taille des données chiffrées).

1.3. Analyse comparative des deux approches

- Identifier les avantages et inconvénients de chaque méthode dans le contexte des systèmes de recommandation.
- Fournir une grille de critères pour guider le choix (précision, protection des données, complexité technique, coût computationnel).

2. Conception et architecture

2.1. Analyse UML et gestion agile

- Utiliser UML pour modéliser les différentes composantes du système, incluant :
    - **Diagrammes de cas d'utilisation** : Identifier les fonctionnalités principales (recommandation, gestion des utilisateurs, etc.).
    - **Diagrammes de séquence** : Illustrer les échanges entre services (API, bases de données, moteur IA, gestion des clés).
    - **Diagrammes de classes** : Modéliser les entités principales (utilisateurs, produits, interactions, clés de chiffrement, etc.).
    - **Diagrammes de déploiement** : Représenter l'architecture microservices.
- Utiliser une méthodologie agile (Scrum ou Kanban) avec des livraisons incrémentales.

2.2. Architecture technique

- Implémenter une architecture microservices, incluant :
    - **Backend en Java** (JPA/REST/WebSocket) pour gérer les utilisateurs, les produits et les clés de chiffrement.
    - **Service IA en Python** pour le moteur de recommandation.
    - **Base de données relationnelle** pour stocker les données chiffrées ou bruitées.
    - **API Gateway** pour orchestrer les appels entre services.

3. Implémentation

3.1. Implémentation du moteur de recommandations

- Utiliser une technique moderne (filtrage collaboratif, factorisation matricielle, apprentissage supervisé ou non supervisé).
- Ajouter une composante de préservation de la vie privée, au choix :
    1. **Differential Privacy :**
        - Ajouter du bruit aux données utilisateur avant entraînement.
        - Utiliser des bibliothèques comme **TensorFlow Privacy** ou **Opacus**.
    2. **Cryptographie homomorphe :**
        - Chiffrer les données utilisateur.
        - Réaliser des calculs directement sur ces données chiffrées avec des outils comme **Microsoft SEAL** ou **PySEAL**.

3.2. Comparaison expérimentale

- Comparer les performances de votre solution (temps de calcul, précision des recommandations) avec :
    - Une implémentation sans préservation de la vie privée.
    - Des solutions standards utilisant des bibliothèques existantes.

4. Livrables et rendu final

Livrables

1. **Code source** :
    - Géré avec Git (utilisation de branches, pull requests, historique des commits).
    - Organisation claire des microservices.
2. **Modélisation UML** :
    - Diagrammes d'analyse et d'architecture adaptés à la méthodologie agile.
3. **Rapport technique** :
    - Résumé des recherches théoriques sur les recommandations et la confidentialité.
    - Démarche choisie (Differential Privacy ou cryptographie homomorphe) et justification.
    - Résultats expérimentaux (performances, précision, confidentialité).
4. **Présentation orale** (15 minutes) :
    - Inclure la démarche, les choix techniques, une introspection et les résultats obtenus.

Critères d'évaluation

1. **Rigueur théorique** (20 %) :
    - Qualité de l'état de l'art et de l'analyse comparative (DP vs cryptographie homomorphe).
2. **Conception UML et méthodologie agile** (20 %) :
    - Pertinence et clarté des diagrammes UML.
    - Gestion efficace du projet.
3. **Qualité technique de l'implémentation** (40 %) :
    - Fonctionnalité et robustesse du moteur de recommandations.
    - Bonne intégration des techniques de préservation de la vie privée.
    - Respect des principes microservices et qualité du code.
4. **Présentation et introspection** (20 %) :
    - Clarté de la présentation et capacité à analyser les forces/faiblesses de la solution.