Café sans fil :
Système de recommandation

Rapport de projet dans le cadre du cours :
IFT 3150 - Projet d'informatique

Auteur : Bio Samir Gbian (20250793)

Superviseur : Louis-Edouard LAFONTANT

Department d'Informatique et de Recherche Opérationnelle (DIRO)
Université de montréal
Date

Table des matières

1 Introduction **3**

2 Exigences et Analyse **4**

3 Conception **5**

4 Implémentation **6**

5 Tests et résultats **7**

6 Conclusion **8**

7 Références **9**

8 Annexe **10**

8.1 Pseudo codes des algorithmes de recommandation 10

8.1.1 Méthodes utilitaires 10

8.1.2 Algorithmes principales 11

Table des figures

1 Introduction

Contexte :
Afin d'améliorer le service des cafés étudiants de l'Université de Montréal (UdeM), le projet Café sans-fil a été initié au trimestre d'automne 2023 durant lequel furent construits l'infrastructure backend et un premier prototype de l'application. Le projet a été poursuivi au trimestre d'hiver durant lequel l'application fut évaluée et enrichi par l'ajout d'éléments à caractères sociaux tel que les évènements, reflétant mieux la nature sociale des cafés étudiants. Le lancement de la plateforme est prévue pour la fin de cet été, intégrant le travail réalisé à l'hiver et des améliorations mineures additionnelles.

Problématique :
Cependant, faute d'exploitation des données fournis entre autre par les utilisateurs, l'expérience utilisateur actuelle manque grandement de personnalisation, impactant la satisfaction client. En effet, si un étudiant possède certaines restrictions ou préférences alimentaires ou des allergies, la plateforme ne l'aide nullement à satisfaire ses exigences (critères), résultant sur un plus lourd travail de recherche, pouvant décourager certains. Cette lacune a aussi des conséquences sur les cafés, impactant les ventes et la fidélisation des clients ainsi que la prise de décision en vue d'une quelconque amélioration de leurs services et du menu. En effet, sans analyse de données, il est difficile pour les gérants d'optimiser le menu ou les services pour éviter des pertes et mieux répondre aux clients du café.

Proposition :
En réponse aux problèmes énoncés, ce projet vise à mettre en place un système de recommandation utilisant les données fournies par les utilisateurs. Pour créer un moyen efficace de collecte et d'analyse de données nécessaire au système de recommandation, nous envisageons enrichir la plateforme avec de nouvelles fonctionnalités permettant aux utilisateurs de communiquer leurs préférences et réagir avec plus de choix aux propositions des cafés.

2 Exigences et Analyse

3 Conception

4 Implémentation

5 Tests et résultats

6 Conclusion

7 Références

8 Annexe

8.1 Pseudo codes des algorithmes de recommandation

8.1.1 Méthodes utilitaires

Algorithm 1 Repas pas encore consommés

Entrée : Menu du restaurant (M), utilisateur actuel (u)
Sortie : Liste de repas

Algorithme(M[1...n], u)
    $L \leftarrow u.historique\_achat$
    Pour chaque repas $r \in L$
        Si $r \in M$:
            $M.retirer(r)$
    retourner M

Algorithm 2 Mesure similarité : Jaccard

Entrée : Deux listes ou ensembles (L_1, L_2)
Sortie : Nombre décimal représentant la similarité

Jaccard($L_1[1...n]$***,***$L_2[1...m]$***) :***
    $union \leftarrow L_1 \cup L_2$
    $inter \leftarrow L_1 \cap L_2$
    retourner $inter.length/union.length$

Algorithm 3 Clusters en fontion des préférences de l'utilisateur

Entrée : Menu du restaurant (M)
Sortie : Liste des régimes contenant les catégories de repas

clusters(M[1...n])
    // Nous avons k régimes différents et p catégories de repas
    $R \leftarrow [[0...n]...k]$
    Pour chaque repas $r \in M$:
        mettre le repas dans un ensemble en fonction de son régime
        rajouter les différents régimes de repas à R
    Pour chaque régime $T[1...\ell] \in R$
        mettre chaque repas du même régime dans une catégorie
    retourner $R[[[0...n]...p]...k]$

8.1.2 Algorithmes principales

Algorithm 4 Collaborative filtering

Description : Recommandations basé sur les similarités entre utilisateurs.
Entrée : Liste des utilisateurs de l'application (U), utilisateur (u)
Sortie : Liste L des recommandations

Algorithme(U[1...n], u) :
    $L \leftarrow [vide]$
    $\tau \leftarrow$ seuil de similarité
    $S \leftarrow U.retirer(u)$
    $L_u \leftarrow$ [[u.likes], [u.repas_consommés], [u.cafés_visités]]
    Pour chaque utilisateur $x \in S_{n\times 1}$
        $L_x \leftarrow$ [[x.likes], [x.repas_consommés], [x.cafés_visités]]
        $J \leftarrow [vide]$
        Pour $i \leftarrow 0$ à $L_x.length$
            $j \leftarrow Jaccard(L_u[i], L_x[i])$
            $J.ajouter(j)$
        $score \leftarrow sum(J)$
        Si $score \geq \tau$:
            $L \leftarrow (L_x[0] \cup L_x[1]) \backslash (L_u[0] \cup L_u[1])$
        $S.retirer(x)$
    retourner L

Algorithm 5 Content based filtering

Description : Recommandations basé sur les habitudes de consommation de l'utilisateur.
Entrée : Menu du cafe (M), utilisateur (u)
Sortie : Liste L des recommandations

Algorithme(M[1...n], u) :
    $L \leftarrow [vide]$
    $P \leftarrow$ Items pas encore achetés (***Algorithme 1***)
    $clusters \leftarrow regrouper\_par\_cluster(M)$
    $cf \leftarrow [vide]$
    Tant que cf.length < clusters.length :
        $c \leftarrow cluster\_favoris(C, u)$
        $cf.inserer(c)$
        $C \leftarrow C.retirer(c)$
    Pour chaque cluster $c \in cf$:
        $L.ajouter(P \cap c)$
    retourner L

---

cluster_favoris(C[[1...n],[1...m],...k], u) :
    $L \leftarrow [empty]$
    Pour chaque cluster $c[1...n] \in C$:
        $tmp \leftarrow [empty]$
        Pour chaque repas $r \in c[1...n]$:
            SI $u \in r.likes$:
                $tmp.ajouter(r)$
        $L.ajouter(tmp.length)$
    $i \leftarrow L.index\_du\_max$
    retourner $C[i]$

---

regrouper_par_cluster(M) :
    $groupes \leftarrow \{vide\}$
    Pour chaque item $\in M$:
        Si $item.cluster \notin groupes$:
            $groupes[item.cluster] = [vide]$
        $groupes[item.cluster].inserer(item)$
    retourner groupes

Algorithm 6 Knowledge based filtering

Description : Recommandations basé sur les spécifications de l'utilisateur.
Entrée : Utilisateur actuel (u), Liste des repas du menu (M)
Sortie : Liste des recommandations.

Algorithme(M[1...n], u) :
    $A \leftarrow u.liste\_allergens$
    $allergenes \leftarrow repas\_allergens(A)$
    $P \leftarrow u.preferences$
    $R \leftarrow clusters(M)$ // ***Algorithme 3***
    $E[0...n] \leftarrow$ récupérer les bons repas en fonction du régime (P[0]) et des
    catégories (P[1]) spécifiées par l'utilisateur
    retourner E

---

repas_allergenes(A[1...n], M[1...m]) :
    $L \leftarrow [vide]$
    Pour chaque repas $r \in M$:
        Si $r.allergens \cap A \neq \emptyset$:
            L.inserer(r)
    retourner L

Algorithm 7 Recommandation globale

Description : Recommandations à tout les utilisateurs.
Entrée : Liste des repas du menu (M)
Sortie : Liste des recommandations

Algorithme(M[1...n]) :
    $k \leftarrow$ nombre de repas à recommander
    $I \leftarrow$ items les plus achetés
    retourner $items\_plus\_aimes(I, k)$

---

items_plus_aimes(I[1...n],k)) :
    $likes \leftarrow [vide]$
    $res \leftarrow [vide]$
    Pour chaque $item \in I$:
        likes.insérer(item.likes.length)
    Pour $i \leftarrow 0$ à k
        $max\_like \leftarrow max(likes)$
        $index \leftarrow likes.index(max\_likes)$
        $res.inserer(I[index])$
        $likes[index] \leftarrow -1$
    retourner res