Rapport n°2

Par *JAST*:
Sarah Santiago
Jimmy Fabre
Thuy-Trang Nguyen
Alban Richard

Sommaire :

1/Introduction

Dans le cadre de notre projet de S4, nous avons choisi d'étudier les algorithmes de recommandation et leur fonctionnement. Pour le début du projet nous n'avions pas vraiment une idée précise du contenu que nous voulions proposer aux utilisateurs mais nous avons trouvé une idée originale et intéressante.

1.1 Watch If ?

Cette application a pour but principal de proposer des films aux utilisateurs en fonction de leur goût. Nous souhaitons garder l'aspect "social" du projet en permettant aux utilisateurs de laisser des commentaires pour donner leur avis sur les différents films ou bien même d'en rajouter des nouveaux pas encore présents sur le site.

2/<u>Répartition des tâches</u> :

Voici un rappel de la répartition des tâches de notre groupe:

Répartition des tâches	Sarah Santiago	Jimmy Fabre	Thuy-Trang Nguyen	Alban Richard
Algorithme		❖	☸	
Communication		☸	❖	
Données	☸			❖
Interface	❖			☸
Réseaux	❖	☸		
Site Web	☸			❖
Installateur/ Lanceur			❖	☸

Legende : ☸ = Responsable ; ❖ = Suppleant

3/Avancement du projet :

Lors de ces derniers mois, le projet a avancé lentement mais nous avons réussi à fournir les résultats que nous attendions et nous sommes encore plus motivés pour finaliser ce projet.

3.1 Algorithme:

Pour mettre en œuvre ce projet, nous utilisons l'algorithme de filtrage collaboratif. Le filtrage collaboratif est la technique la plus couramment utilisée lorsqu'il s'agit de créer des systèmes de recommandation intelligents qui peuvent apprendre à donner de meilleures recommandations à mesure que davantage d'informations sur les utilisateurs sont collectées.

La plupart des sites Web comme Amazon, YouTube et Netflix utilisent le filtrage collaboratif dans le cadre de leurs systèmes de recommandation sophistiqués. Vous pouvez utiliser cette technique pour créer des recommandations qui donnent des suggestions à un utilisateur sur la base des goûts et des aversions d'utilisateurs similaires.

I) Qu'est-ce que le filtrage collaboratif ?

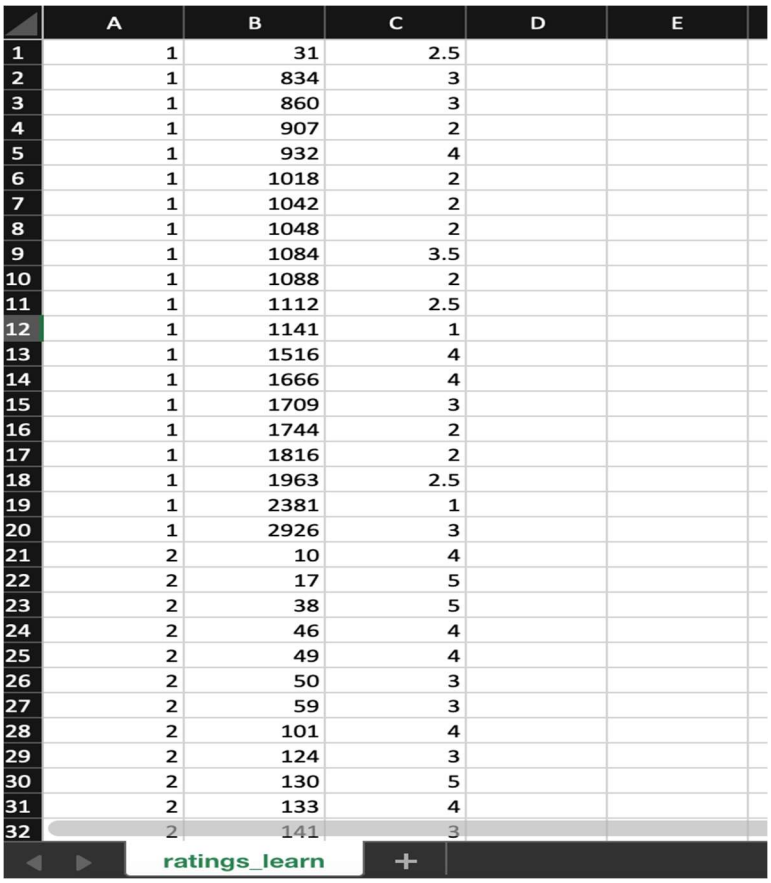
Le filtrage collaboratif est une technique qui peut filtrer les éléments qu'un utilisateur pourrait aimer sur la base des réactions d'utilisateurs similaires. Cela fonctionne en recherchant un grand groupe de personnes et en trouvant un plus petit ensemble d'utilisateurs ayant des goûts similaires à un utilisateur particulier. Il examine les éléments qu'ils aiment et les combine pour créer une liste classée de suggestions.

II). L'ensemble de données

Pour tester les algorithmes de recommandation, nous aurons besoin de données contenant un ensemble d'éléments et un ensemble d'utilisateurs ayant réagi à certains des éléments. La réaction peut être explicite (notation sur une échelle de 1 à 5, aime ou n'aime pas) ou implicite (visionner un article, l'ajouter à une liste de souhaits, le temps passé sur un article).
Nous collectons ici un ensemble de données des avis de 548 clients sur 9125 articles.

III). Étapes impliquées dans le filtrage collaboratif

a. Création d'une matrice utilitaire de notes entre les utilisateurs et les articles à partir des données du fichier csv



	A	B	C	D	E
1	1	31	2.5		
2	1	834	3		
3	1	860	3		
4	1	907	2		
5	1	932	4		
6	1	1018	2		
7	1	1042	2		
8	1	1048	2		
9	1	1084	3.5		
10	1	1088	2		
11	1	1112	2.5		
12	1	1141	1		
13	1	1516	4		
14	1	1666	4		
15	1	1709	3		
16	1	1744	2		
17	1	1816	2		
18	1	1963	2.5		
19	1	2381	1		
20	1	2926	3		
21	2	10	4		
22	2	17	5		
23	2	38	5		
24	2	46	4		
25	2	49	4		
26	2	50	3		
27	2	59	3		
28	2	101	4		
29	2	124	3		
30	2	130	5		
31	2	133	4		
32	2	141	3		

ratings_learn

Lorsque nous travaillons avec de telles données, nous les voyons principalement sous la forme d'une matrice composée des réactions données par un ensemble d'utilisateurs à certains articles d'un ensemble des articles. Chaque ligne contiendrait les évaluations données par un utilisateur, et chaque colonne contiendrait les évaluations reçues par un article.

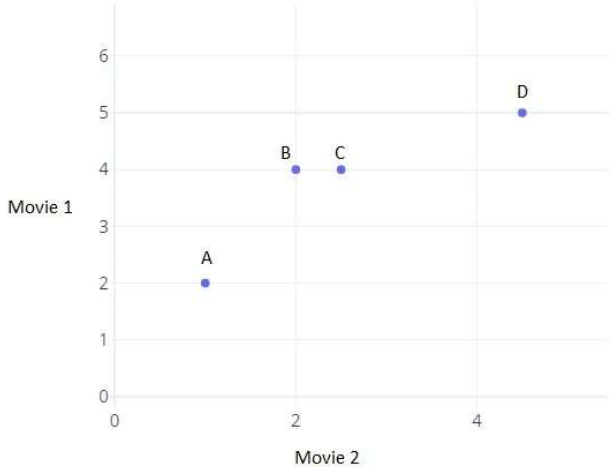
Dans la plupart des cas, les cellules de la matrice sont vides, car les utilisateurs n'évaluent que quelques éléments. Il est très peu probable que chaque utilisateur évalue ou réagisse à chaque article disponible. Une matrice avec des cellules principalement vides est appelée clairsemée, et l'opposé de cela (une matrice principalement remplie) est appelée dense.

b. Trouver la similarité d'un nouvel utilisateur (ou d'un utilisateur concerné) avec d'autres utilisateurs à l'aide de la similarité en cosinus centré

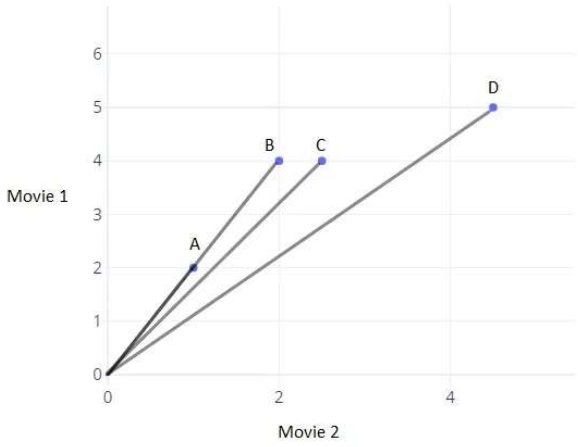
La similarité cosinus est une mesure de similarité entre deux vecteurs non nuls définis dans un espace de produit interne. La similarité cosinus est le cosinus de l'angle entre les vecteurs; c'est-à-dire qu'il s'agit du produit scalaire des vecteurs divisé par le produit de leurs longueurs. Il s'ensuit que la similarité cosinus ne dépend pas des grandeurs des vecteurs, mais seulement de leur angle.

Pour comprendre le concept de similarité, créons d'abord un jeu de données simple. Les données incluent quatre utilisateurs A, B, C et D, qui ont noté deux films. Les classements sont stockés dans des listes, et chaque liste contient deux nombres indiquant le classement de chaque film :

Les notes par A sont [1.0, 2.0]
Les notes par B sont [2,0, 4,0]
Les notes par C sont [2,5, 4,0]
Les notes par D sont [4,5, 5,0]



Dans le graphique ci-dessus, chaque point représente un utilisateur et est tracé par rapport aux notes qu'il a attribuées à deux films.



Le graphique montre quatre lignes joignant chaque point à l'origine. Les lignes pour A et B coïncident, rendant l'angle entre elles égal à zéro. Vous pouvez considérer que si l'angle entre les lignes augmente, la similitude diminue et si l'angle est nul, les utilisateurs sont très similaires. Pour calculer la similarité à l'aide de l'angle, vous avez besoin d'une fonction qui renvoie une similarité plus élevée ou une distance plus petite pour un angle inférieur et une similarité inférieure ou une distance plus grande pour un angle supérieur. Le cosinus d'un angle est une fonction qui décroît de 1 à -1 lorsque l'angle augmente de 0 à 180.

Vous pouvez utiliser le cosinus de l'angle pour trouver la similitude entre deux utilisateurs. Plus l'angle est élevé, plus le cosinus sera bas et donc, plus la similarité des utilisateurs sera faible. Vous pouvez également inverser la valeur du cosinus de l'angle pour obtenir la distance cosinus entre les utilisateurs en la soustrayant de 1.

Notez que les utilisateurs A et B sont considérés comme absolument similaires dans la métrique de similarité cosinus malgré des notes

différentes. C'est en fait un phénomène courant dans le monde réel, et les utilisateurs comme l'utilisateur A sont ce que vous pouvez appeler des évaluateurs difficiles. Un exemple serait un critique de cinéma qui attribue toujours des notes inférieures à la moyenne, mais le classement des éléments de sa liste serait similaire à celui des évaluateurs moyens comme B.

Pour prendre en compte ces préférences individuelles des utilisateurs, vous devrez amener tous les utilisateurs au même niveau en supprimant leurs préjugés. Vous pouvez le faire en soustrayant la note moyenne donnée par cet utilisateur à tous les éléments de chaque élément évalué par cet utilisateur. Voici à quoi cela ressemblerait :

- Pour l'utilisateur A, le vecteur de notation [1, 2] a la moyenne de 1,5. Soustraire 1,5 de chaque note vous donnerait le vecteur [-0,5, 0,5].
- Pour l'utilisateur B, le vecteur d'évaluation [2, 4] a la moyenne de 3.

En soustrayant 3 de chaque évaluation, vous obtiendrez le vecteur [-1, 1].En faisant cela, vous avez changé la valeur de la note moyenne donnée par chaque utilisateur à 0. Essayez de faire la même chose pour les utilisateurs C et D, et vous verrez que les notes sont maintenant ajustées pour donner une moyenne de 0 pour tous les utilisateurs , ce qui les ramène tous au même niveau et supprime leurs préjugés. Le cosinus de l'angle entre les vecteurs ajustés est appelé cosinus centré. Cette approche est normalement utilisée lorsqu'il y a beaucoup de valeurs manquantes dans les vecteurs et que vous devez placer une valeur commune pour remplir les valeurs manquantes.

Remplir les valeurs manquantes dans la matrice des notations avec une valeur aléatoire peut entraîner des inexactitudes. Un bon choix pour remplir les valeurs manquantes pourrait être la note moyenne de chaque

utilisateur, mais les moyennes originales des utilisateurs A et B sont respectivement de 1,5 et 3, et remplir toutes les valeurs vides de A avec 1,5 et celles de B avec 3 serait en faire des utilisateurs différents. Mais après avoir ajusté les valeurs, la moyenne centrée des deux utilisateurs est 0, ce qui vous permet de saisir plus précisément l'idée que l'élément est supérieur ou inférieur à la moyenne pour les deux utilisateurs, toutes les valeurs manquantes dans les vecteurs des deux utilisateurs ayant la même valeur 0.

c. Prédire les notes des éléments qui ne sont pas encore notés par un utilisateur

Après avoir déterminé une liste d'utilisateurs similaires à un utilisateur U, vous devez calculer la note R que U donnerait à un certain élément I. Encore une fois, tout comme la similarité, vous pouvez le faire de plusieurs façons. Vous pouvez prédire que la note R d'un utilisateur pour un élément I sera proche de la moyenne des notes attribuées à I par les 5 ou 10 meilleurs utilisateurs les plus similaires à U. La formule mathématique de la note moyenne donnée par n utilisateurs ressemblerait à comme ça:

$$R_U = (\sum_{u=1}^{n} R_u)/n$$

Cette formule montre que la note moyenne attribuée par les n utilisateurs similaires est égale à la somme des notes qu'ils ont attribuées divisée par le nombre d'utilisateurs similaires, soit n. Il y aura des situations où les n utilisateurs similaires que vous avez trouvés ne sont pas également similaires à l'utilisateur cible U. Les 3 premiers d'entre eux peuvent être très similaires, et le reste peut ne pas être aussi similaire à U que les 3 premiers. Dans ce cas , vous pouvez envisager une approche dans laquelle la note de

l'utilisateur le plus similaire compte plus que celle du deuxième utilisateur le plus similaire, etc. La moyenne pondérée peut nous aider à y parvenir. Dans l'approche de la moyenne pondérée, vous multipliez chaque note par un facteur de similarité (qui indique à quel point les utilisateurs sont similaires). En multipliant par le facteur de similarité, vous ajoutez des pondérations aux notes. Plus le poids est lourd, plus la note compte. Le facteur de similarité, qui agirait comme des poids, devrait être l'inverse de la distance discutée ci-dessus car moins de distance implique une plus grande similarité. Par exemple, vous pouvez soustraire la distance cosinus de 1 pour obtenir la similarité cosinus. Avec le facteur de similarité S pour chaque utilisateur similaire à l'utilisateur cible U, vous pouvez calculer la moyenne pondérée à l'aide de cette formule :

$$R_U = (\sum_{u=1}^{n} R_u * S_u)/(\sum_{u=1}^{n} S_u)$$

Dans la formule ci-dessus, chaque note est multipliée par le facteur de similarité de l'utilisateur qui a donné la note. La note finale prédite par l'utilisateur U sera égale à la somme des notes pondérées divisée par la somme des pondérations. Avec une moyenne pondérée, vous accordez plus d'importance aux évaluations d'utilisateurs similaires par ordre de similarité.

LES TRAVAUX QUE NOUS AVONS RÉALISÉS:

1. Écrire une fonction pour convertir les données du fichier csv en matrice

```c
void get_utility_matrix(double *utility_matrix, char *s, int No_of_movies){
        char *line, *record;
        char tmp[1024];
        int i=0, j=0, k=0;
        FILE *fstream = fopen(s,"r");
        while((line=fgets(tmp,sizeof(tmp),fstream))!=NULL)
        {
                record = strtok(line,",");
                while(record!=NULL)
                {
                        if(k==0)
                        { //first string is user id, which will give our row
                                i = atoi(record)-1;
                        }
                        else if(k==1)
                        { //second string is movie id which will give our coloumn
                                j = atoi(record)-1;
                        }
                        else
                        { //third is the actual rating
                                utility_matrix[i*No_of_movies + j] = atof(record); //converting string to float/doubl
                        }
                        record = strtok(NULL,",");
                        k++;
                }
                k=0;
        }
        fclose(fstream);
        free(line);
```

2. Écrire une fonction qui trie les films recommandés et leurs notes de la note la plus élevée à la plus basse

```c
void sort(int *recommended_movies, double *predicted_ratings, int no_of_recommended_movies){
        int i=0,j=0;

        //sorting recommended movies and their ratings from highest rating to lowest
        for(i=0;i<no_of_recommended_movies;i++){
                for(j=i+1;j<no_of_recommended_movies;j++){
                        if(predicted_ratings[i]<predicted_ratings[j]){
                                double temp1; int temp2;
                                temp1 = predicted_ratings[i]; temp2 = recommended_movies[i];
                                predicted_ratings[i] = predicted_ratings[j]; recommended_movies[i] = recommended_movies[j];
                                predicted_ratings[j] = temp1; recommended_movies[j] = temp2;
                        }
                }
        }
}
```

3. Écrire une fonction qui calcule la note moyenne d'un utilisateur pour tous les films qu'il regarde

```c
double calc_average(double *utility_matrix,int No_of_movies)
{
        double average, sum=0;
        int i=0, count=0;

        //traverse through each rating
        for(i=0;i<No_of_movies;i++)
        {
                if(utility_matrix[i]!=0)
                { // if rated
                        count++; //increase count
                        sum += utility_matrix[i]; //add to total sum
                }
        }
        return average = sum/count;
}
```

4. Écrire une fonction pour créer une matrice normalisée qui soustrait la note moyenne de la note actuelle

```c
void normalize_matrix(double *utility_matrix, double *normalized_matrix, int No_of_users, int No_of_movies)
{ //inputs: utility matrix and new matrix to save normalized ratings
        int i=0,j=0;
        for(i=0;i<No_of_users;i++)
        {
                //calculate average for each user
                double average = calc_average(&utility_matrix[i*No_of_movies],No_of_movies);

                //traverse through each movie rating
                for(j=0;j<No_of_movies;j++)
                {
                        if(utility_matrix[i*No_of_movies + j] == 0)
                        {
                                normalized_matrix[i*No_of_movies + j] = 0;
                        }
                        else
                        {
                                normalized_matrix[i*No_of_movies + j] = utility_matrix[i*No_of_movies + j] - average; /
                        }
                }
        }
}
```

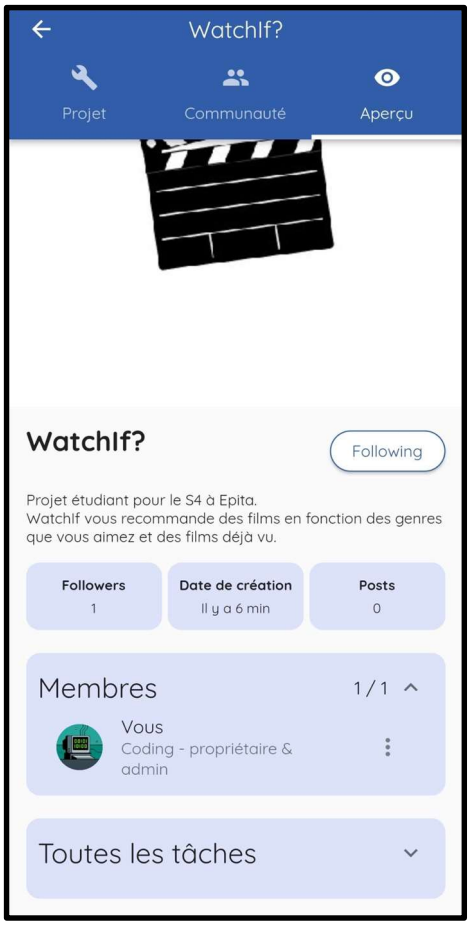
5. Écrire une fonction pour calculer la similarité

```c
double pearson_correlation(double *A, double *B, unsigned int size)
{
    double dot_p=0.0;
    double mag_a=0.0;
    double mag_b=0.0;
    int i;
    for(i=0; i<size; i++)
    {
        dot_p += A[i]*B[i];
        mag_a += A[i]*A[i];
        mag_b += B[i]*B[i];
    }
    return dot_p/(sqrt(mag_a)*sqrt(mag_b));
}

void calc_similarity(double *normalizeduser, double *normalized_matrix, double *similarity, int No_of_users, int No_of_movies)
{
        int i=0,j=0;
        for(i=0;i<No_of_users;i++)
        { //traverse through each user
                double *A;
                A = (double *)malloc(sizeof(double) * No_of_movies);

                //get rating vector for that user
                for(j=0; j<No_of_movies; j++)
                {
                        A[j] = normalized_matrix[i*No_of_movies + j];
                }

                //find similarity between new user and ith user
                similarity[i] = pearson_correlation(normalizeduser,A,No_of_movies);
                free(A);
        }
}
```

6. Écrire une fonction pour prédire les films qu'ils regarderont et les points qu'ils donneront pour ces films

```c
int make_prediction(double *user, int *similar_users, int no_of_susers, double *similarity, double *utility_matrix, int *recommended_movies, double *predic
       int i=0,k=0;
       int no_of_recommended_movies = 0;
       for(i=0;i<No_of_movies;i++)
       { //traverse through each movie
               double sum1=0, sum2=0;
               int count=0;
               if(user[i]==0){ //if not rated by the user
                       for(k=0;k<no_of_susers;k++)
                       { //traverse through similar users
                               if(utility_matrix[similar_users[k]*No_of_movies + i]==0) continue; //all similar users who have also rated movie i
                               sum1 += similarity[similar_users[k]]*utility_matrix[similar_users[k]*No_of_movies + i];
                               sum2 += similarity[similar_users[k]];
                               count++;
                       }
                       if(count>1)
                       { //the movie is common between atleast two users otherwise if it only has one user then we will get that same rating
                               recommended_movies[no_of_recommended_movies] = i; //add movie index to recommended movies
                               predicted_ratings[no_of_recommended_movies] = sum1/sum2; //make prediction
                               no_of_recommended_movies++;
                       }
               }
       }
       return no_of_recommended_movies;
}
```

3.2 Communication:



Nous commençons à parler du projet autour de nous et nous avons créé une page sur l'application CoMeet, réalisée par un étudiant d'Epita, qui permet de partager nos projets.

Les utilisateurs de l'application pourront suivre l'avancée du développement à travers des posts.

Par la suite une communication sera réalisée sur les réseaux sociaux et au seins des élèves de l'école pour ramener le plus d'utilisateur sur notre application.

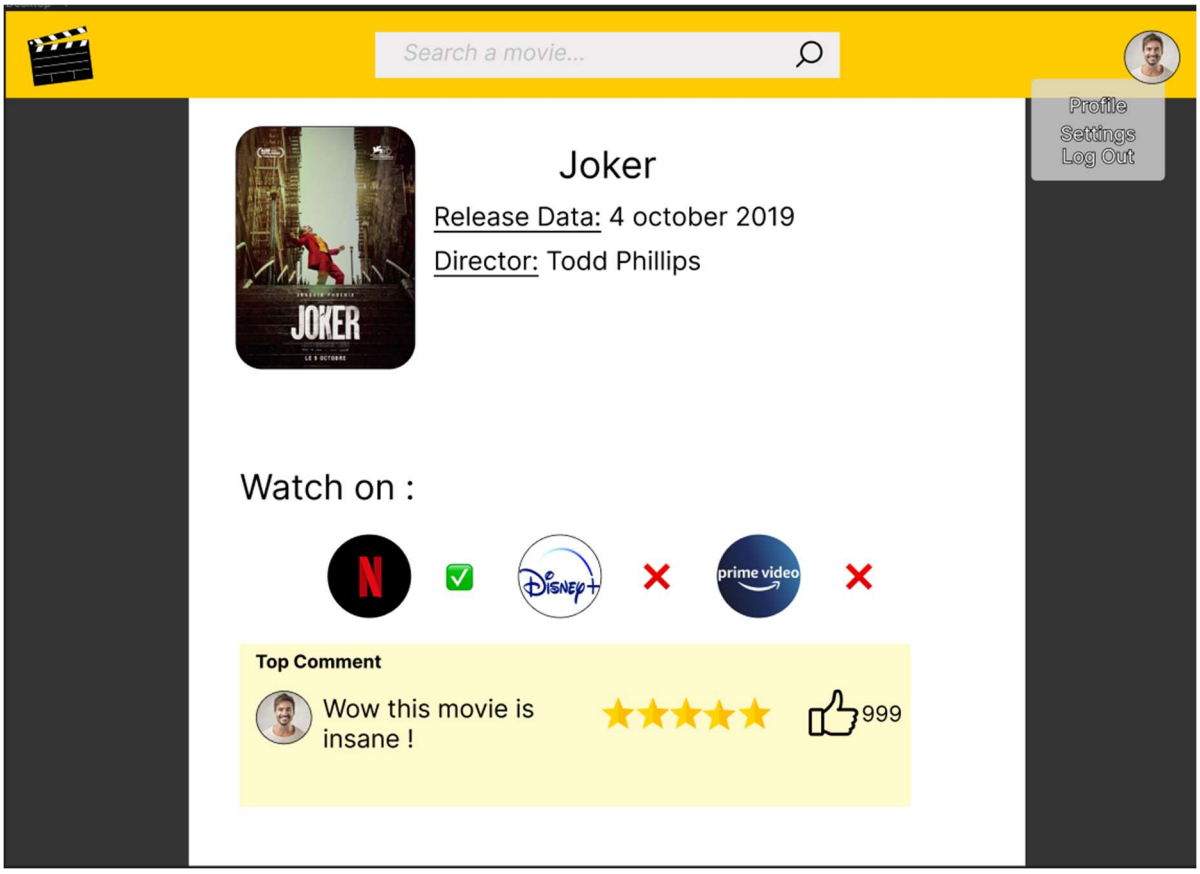
3.3 Données:

Nous possédons un dataset de plus de 9000 films avec leurs genres et des notes sur 5 donner par des utilisateurs.
Ce dataset va permettre d'entraîner l'algorithme de recommandation.
Il est pour le moment sous format csv mais nous allons le convertir en une base de données qui peut être parcourue grâce à Sqlite3.

3.4 Interface:

L'interface nous a posé quelques soucis, la bibliothèque gtk étant très complète et en même temps compliqué à utiliser nous nous sommes perfectionner dans son utilisation mais pour le moment nous obtenons pas le résultat attendu.

Nous avons peaufiné notre maquette pour avoir une idée précise du résultat attendu.

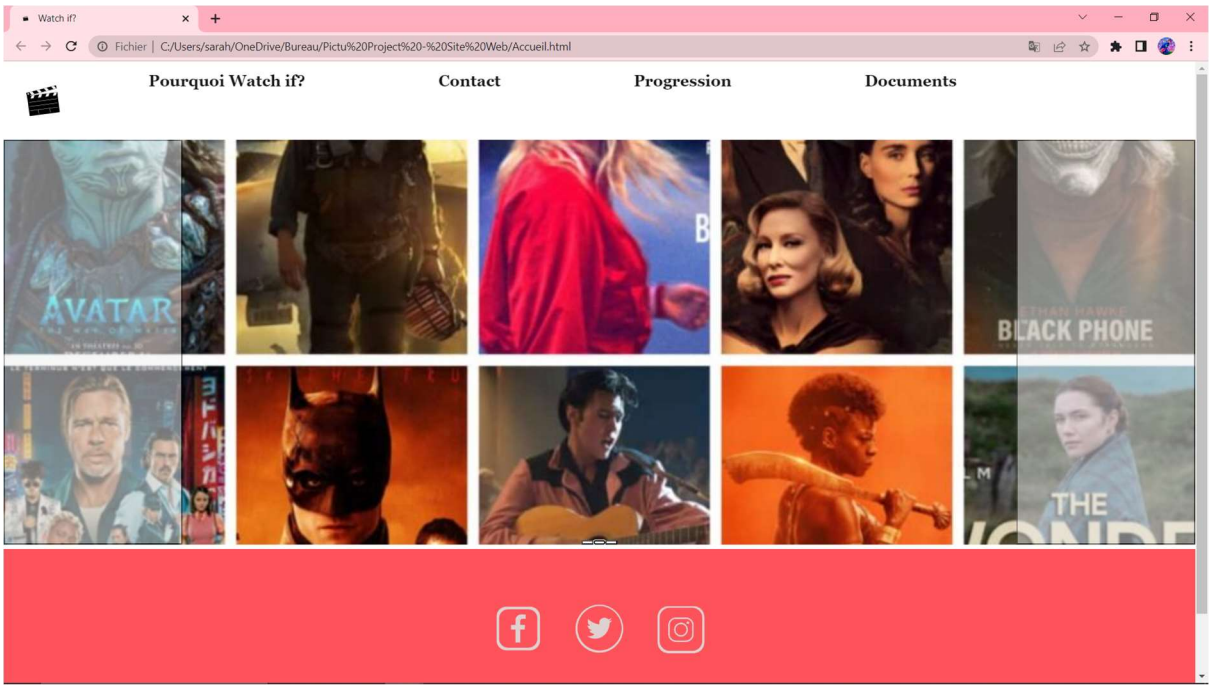


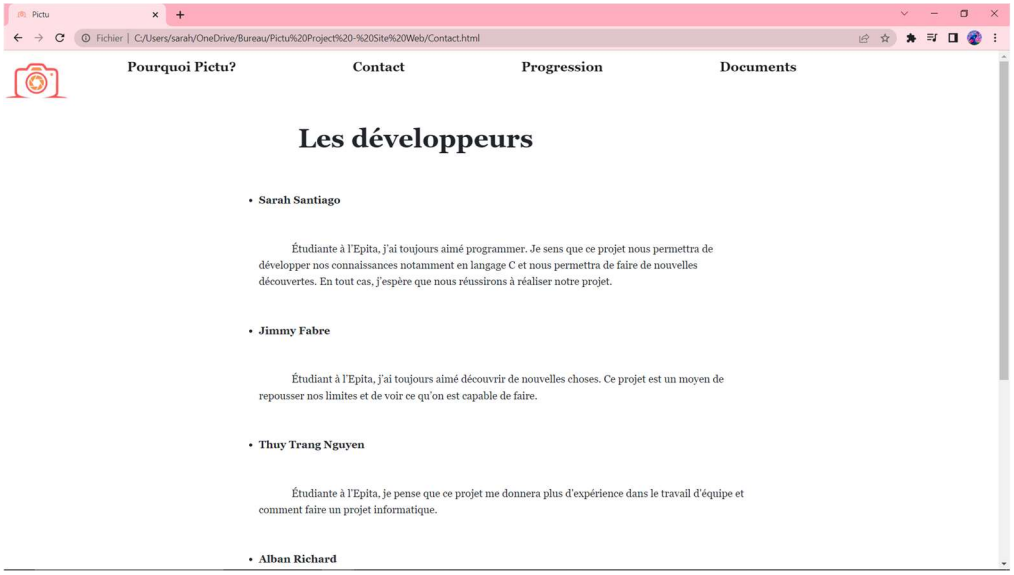
3.5 Réseau:

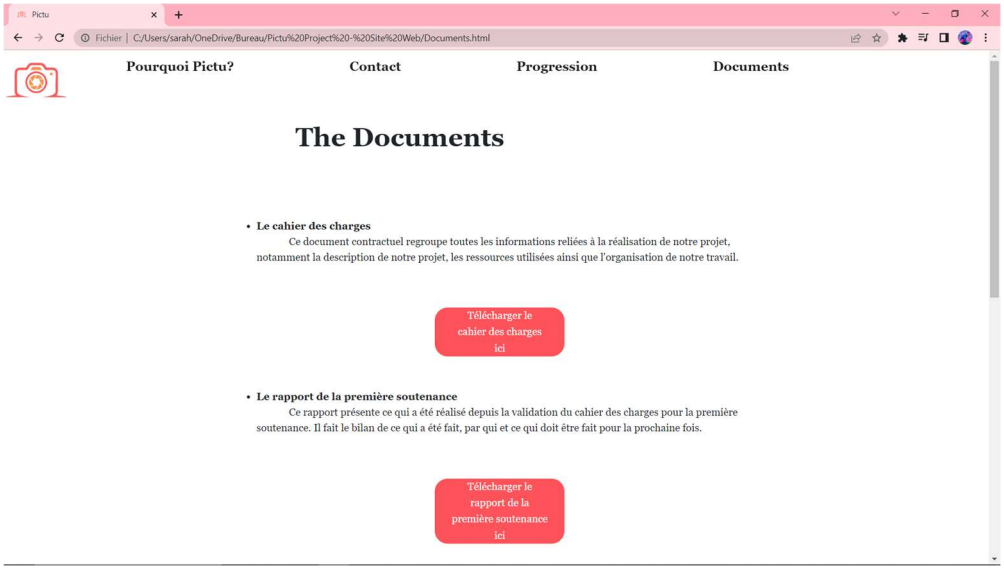
Le serveur peut dorénavant gérer plusieurs connexions simultanées grâce à l'utilisation de Threads.
Nous nous sommes familiarisé avec l'utilisation de sqlite3 en C et nous pouvons maintenant réaliser une connexion à l'aide d'un identifiant et un mot de passe qui est stocké dans une base de données.

3.6 Site Web:

Après la première soutenance, nous avons apporté quelques modifications à notre suite. Tout d'abord, puisque nous avons fait quelques modifications sur le projet en lui-même, nous devions changer le design, ne serait-ce que le logo, etc... Aussi, nous avons pu le remplir un peu plus . Voici des photos du site:







3.7 Installateur / Lanceur:

Cette partie ne pourra être commencée et finie qu'à la fin du projet quand l'application sera prête à sortir.

4/**Nos impressions** :

Sarah Santiago:

Malgré plusieurs difficultés rencontrées, je me réjouis tout autant de travailler sur ce projet qui va nous permettre d'enrichir nos connaissances.

Jimmy Fabre:

Le projet est vraiment passionnant et je pense que ce groupe peut mener ce projet à bien et fournir un produit de qualité à la fin.

Thuy Trang Nguyen:

Ce projet est incroyable, il m'a aidé à utiliser le langage C plus couramment. Bien qu'il ait été très difficile d'écrire le code car dans le processus d'écriture, nous avons rencontré beaucoup d'erreurs lors de la compilation et nous avons dû passer beaucoup de temps à les corriger.

Alban Richard:

Par obligation médical Alban n'a pas pus rédiger son impressions mais depuis la 1er soutenance il a su s'impliquer dans le projet et s'intéresser aux autres parties.

5/Prévisions pour la soutenance finale :

Pour la prochaine soutenance, nous comptons maintenir le rythme et atteindre les objectifs que nous avions fixés dans notre cahier des charges.

Pour rappel, nous avions fixés ce tableau:

Répartition des tâches	1re soutenance	2e soutenance	3e soutenance
Algorithme	☆ ☆	☆ ☆	☆ ☆ ☆
Communication		☆	☆ ☆ ☆
Données		☆	☆ ☆ ☆
Interface	☆	☆ ☆	☆ ☆ ☆
Réseaux	☆	☆	☆ ☆ ☆
Site Web	☆	☆	☆ ☆ ☆
Installateur / Lanceur			☆ ☆ ☆

Legende : ☆ = Tâche commencée ; ☆ ☆ = Tâche avancée ; ☆ ☆ ☆ = Tâche terminée

Cependant, des petites étoiles ne représentent rien de concret. Il serait plus judicieux de donner des objectifs réels afin de ne pas rester trop vague. Ainsi, voici une liste d'objectifs que nous nous fixons pour la prochaine soutenance:

Algorithme:

Nous avons réussi à écrire des fonctions discrètes qui correspondent aux étapes nécessaires à la construction d'un système de recommandation. Dans la soutenance suivante, nous trouverons un moyen de lier les fonctions ensemble et de finir le produit.

Communication:

Pour la prochaine soutenance, nous voulons réaliser une petite campagne de communication pour faire connaître notre application.

Données:

Pour la prochaine soutenance, nous allons essayer de récupérer encore plus de données pour agrandir notre catalogue de film.

Interface:

Bien évidemment pour la dernière soutenance l'interface devra être finie et complète en espérant que nous arriverons à implémenter toutes les fonctionnalités que nous désirons.

Réseau:

Comme l'interface le serveur devra être à 100% fonctionnel en espérant que les test en ligne ne pose pas de soucis supplémentaire.

Site Web:

Nous comptons le compléter au niveau du contenu ainsi que le perfectionner pour le rendre plus agréable à utiliser.

Installateur / Lanceur:

L'application aura son installateur qui pourra être téléchargé depuis notre site web.

6/Conclusion :

En définitive, nous pensons toujours mener ce projet à bien malgré les difficultés que nous pouvons rencontrer. En effet, ce projet nous tient tout autant à cœur qu'au début et nous rapporte une autre vision de la programmation. Aussi, nous avançons dans les temps sans avoir de retard conséquent qui nous serait fatal dans le cas contraire. Nous allons alors nous concentrer sur ce travail afin de rendre un projet de qualité à la fin de l'année.