Documentatia proiectului ReadsProfiler

Mariuta Silviu-Andrei[1 [0000−1111−2222−3333]]

Facultatea de Informatica ,Universitatea "Alexandru Ioan Cuza" Iasi,Strada General Berthelot 16 700483 Iasi, Romania silviu.andrei53m@gmail.com

Abstract. Documentatia proiectului readsProfiler ce consta intr-o aplicatie client-server ce ofera acces la o librarie online si recomanda de carti userilor,atat pe baza cautarilor si descarcarilor acestora cat si pe baza ratingurilor tuturor userilor .Documentatie cuprinde detalii despre tehnologiile utilizate,arhitectura apicatiei,detalii de implementare ,use cases.

Keywords: readProfiler · Machine Learning · Computer Networks

1 Introducere

Documentatia proiectului cuprinde atat detalii de arhitectura cat si de implementare.

Pentru realizarea recomandarilor se foloseste un algoritm de invatare automata,mai exact algoritmul kmeans.Odata clusterizati userii,se vor face recomandari pe baza clusterului din care fac parte,fiind recomandate cartile cele mai indragite de userii din clusterul respectiv.

Se prezinta tehnologiile utilizate in realizarea acestui proiect,si motivatia din spatele alegerilor facute.

Se prezinta arhitectura bazei de date (folosita pentru stocarea informatiilor despre utilizatori)si a aplicatiei in sine.

Se prezinta algoritmul kmeans folosit pentru a recomanda userilor carti.Se ofera si secvente de cod relevant in C/C++.

De asemenea ,sunt oferite use case-uri pentru o mai buna intelegere a modului de operare al aplicatiei.

Posibile imbunatatiri ale aplicatiei sunt oferite.

2 Tehnologiile utilizate

Se foloseste tehnologia TCP intrucat aceasta tehnologie garanteaza transferul integral al datelor,fara pierderi si in ordine.Acest lucru este foarte important in cadrul acestei aplicatie client-server unde se transfera pachete de date a caror integritate este esentiala(transmiterea unor carti de la server la client:trebuie asigurata transmiterea corecta a datelor,altfel continutul cartii primite de client o sa difere de continutul adevarat pe care serverul il trimite iar cartea devine imposibil de citit).

Baza de date a aplicatiei este realizata in SQLite,fiind gratuit si usor de folosit. Se foloseste API-ul SQLite pentru C/C++ deoarece permite interogarea bazei de date cu o oarecare usurinta.

Se foloseste Qt pentru interfata grafica a clientului.

3 Arhitectura aplicatiei

Se foloseste o baza de date pentru a retine detalii despre cautarile,descarcarile si ratingurile tuturor utilizatorilor.

Arhitectura bazei de date:

author: name , genres

books: id_book , genre_id , author , title , publish_year , ISBN , average_rating

downloads: user_id , book_id

searches: user_id, searched_id, clicked_id

users: id_user, username, id_book, rating

genres: genre_id , genre_name , parent_id

Aplicatia respecta patternul Model View Controller.

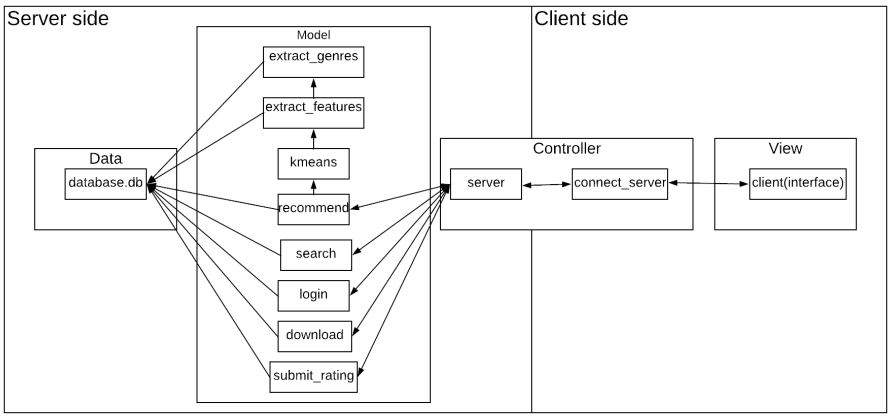

Fig. 1. Arhitectura UML

Model -extraxt_genres: Creeaza ontologia de genuri necesara pentru a creea vectorul de features .

-extract_features: Creeaza vectorul de features folosit de catre algoritmul kmeans pentru a face recomandari.

-download: Cauta fisierul care contine cartea ce se doreste descarcata

-login: Cauta in baza de date un username.

-search: Cauta in baza de date,dupa titlu,gen,autor sau anul publicatiei o carte

-kmeans: Realizeaza clusterizarea utilizatorilor.

-recommend: Cauta in baza de date cartile ce urmeaza a fi recomandate utlizatorilor

-submit_rating: Adauga ratingul unui utilizator in baza de date.

Data

-database.db:baza de date

View

-client(user_interface): Interfata grafica cu care utilizatorul interactioneaza,realizata in Qt

Controller

-server: Primeste cererile de la client si le realizeaza,trimitand inapoi mesaje clientului.

-conenct_server: Trimite cereri catre server si primeste inapoi mesaje pe care le trimite ulterior interfetei grafice cu care interactioneaza utiizatorul

4 Detalii de implementare

4.1 Cod relevant

Pentru realizarea recomandarilor se foloseste o tehnica numita collaborative filtering folosind un algoritm de unsupervised learning,mai exact algoritmul K Means.Acesta reuseste sa grupeze userii cu gusturi asemanatoare(ce au ratinguri,cautari si descarcari asemanatoare) in "clusteri".Apoi fiecarui user dintr-un cluster i se va recomanda carti pe care ceilalti useri din cluster le-au descarcat,cautat sau le-au dat un rating foarte bun.

Pentru fiecare user vom construi un feature vector care va contine detalii privind activitatea lor.

Algoritmul K Means are trei etape:

1.Initializarea random a centroizilor

2.Asignarea clusterilor. Fiecare user va fi asignat clusterului a carui centroid este cel mai apropiat de acesta.

Notiunea de distanta este definita ca fiind distanta euclidiana.Astfel,aceasta etapa a algoritmului poate fi scrisa matematic,ca o minimizare.Pentru un user i :

$$c[i] = \min_k ||x[i] - centroid[k]||^2 \tag{1}$$

,unde $c[i]$ reprezinta id-ul clusterului de care userul i apartine.

3.Mutarea centroizilor In acest moment,fiecare user apartine unui cluster i,unde i reprezinta indicele centroidului de care userul este cel mai apropiat.

In etapa 3 se actualizeaza coordonatele centroizilor,pentru ca acestia sa se situeze in "centrul" clusterilor pe care ii desemneaza.Acest lucru se face prin actualizarea coordonatelor ca fiind media coordonatelor punctelor din cluster.

Implementarea C/C++ a algoritmului K Means:

```
void kmeans(int x[num_users][num_features])
{
        float u[num_centroids][num_features],min_distance=-1;
        int c[num_users],num_points[num_centroids];
        //u[0]...u[k]- centroids' coordinates
        //c[0].c[i]- to which cluster user i belongs
        //random initialization
        for(int k=0;k<num_centroids;k++)
        {
                for(int i=0;i<num_features;i++)
                u[k][i]=rand()%5;
        }
        //cluster assigment
        for(int i=0;i<num_users;i++)
        {
                min_distance=-1;
                for(int k=0;k<num_centroids;k++)
                {
                        float distance=0;
                        //compute distance from centroid k to user i
                        for(int j=0;j<num_features;i++)
                        {
                                distance=distance+pow(x[i][j]-u[k][j],2);
                        }
                        if(distance<min_distance ||min_distance==-1)
                        {
                                min_distance=distance;
                                c[i]=k;
                        }
                }

        }
        //moving centroid
        //num_points[i]-number of users belonging to centroid i
        for(int k=0;k<num_centroids;k++)
        {
                for(int i=0;i<num_features;i++)
                u[k][i]=0;

        }
        for(int i=0;i<num_users;i++)
        {
                for(int j=0;j<num_features;j++)
                {
                        u[c[i]][j]=u[c[i]][j]+x[i][j];
```

```
                        num_points[c[i]]+=1;
                }
        }
        for(int i=0;i<num_centroids;i++)
        {
                for(int j=0;j<num_features;j++)
                {
                        u[i][j]=u[i][j]/num_points[i];
                }
        }
}
```

4.2 Use cases

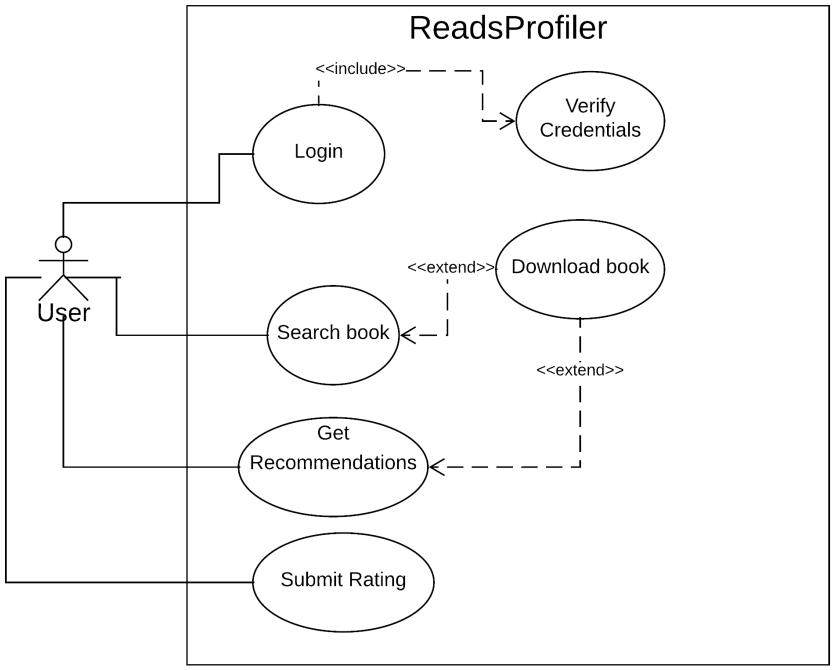

Fig. 2. Diagrama UML a Use Case-urilor

Preconditii:Search book,Get Recommendations se pot realiza abia dupa ce s-a realizat logarea cu succes.

Ex de scenarii de utilizare:

Utilizatorul incearca sa se logheze prin username→ serverul verifica existenta utilizatorului in baza de date →nu se gaste username-ul respectiv in baza de date asa ca logarea nu are succes.Utilizatorul incearca sa caute o carte→serverul remarca faptul ca logarea nu a avut loc cu succes→serverul trimite un mesaj clientului cerandu-i datele de logare.→clientul trimite mesajul interfetei grafice cu care utilizatorul interactioneaza

Utilizatorul incearca sa se logheze prin username→ serverul verifica existenta utilizatorului in baza de date →se gaseste in baza de date usernameul si are loc logarea→ userul inchide aplicatia fara sa faca altceva.

Utilizatorul logat primeste lista de recomandari → descarca o carte din lista de recomandari

Utilizatorul logat cauta o carte dupa titlu,gen,autor,an de publicatie,ISBN sau rating→se cauta in baza de date cartea respectiva iar rezultatele cautarii sunt afisate utilizatorului→Utilizatorul descarca una din cartile primite ca rezultat→Utilizatorul primeste o lista de recomandari de alte carti ce s-ar putea sa ii placa→Utilizatorul nu descarca niciuna din cartile recomandate.

Utilizatorul logat cauta o carte→ serverul nu gaseste nicio carte care sa se potriveasca → se trimite o lista de recomandari din care userul alege sa descarce o carte

5 Concluzii

Se poate imbunatatii algoritmul de realizare a recomandarilor prin implementarea metodei Elbow pentru a alege un numar optim de clusteri.

6 Bibliografie

References

1. Tutorialspoint, https://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm
2. Coursera, https://www.coursera.org/learn/machine-learning
3. Pagina cursului de retele de calculatoare, https://profs.info.uaic.ro/ computernetworks/index.php
4. Wikipedia,K-means clustering, https://en.wikipedia.org/wiki/K-means_clustering
5. Wikipedia,Model-View-Controller,https://en.wikipedia.org/wiki/Model-view-controller
6. Wikipedia,Use Case,https://en.wikipedia.org/wiki/Use_case
7. Forum Qt, despre qmessagebox-uri ,https://forum.qt.io/topic/58214/solved-qmessagebox-buttons
8. QtCentre,Convertire QString-char*,https://www.qtcentre.org/threads/30516-How-to-convert-QString-to-const-char*
9. Documentatia qt, qmake,https://doc.qt.io/archives/3.3/qmake-manual-3.html
10. Tutorial qt,https://www.youtube.com/playlist?list=PLS1QulWo1RIZiBcTr5urECberTITj7gjA