ReadsProfiler

Vlad corjuc

January 2019

1 Introducere

Lucrarea de față prezintă aspecte ale proiectului ReadsProfiler propus că temă la disciplină Rețele de Calculatoare. Printre aspectele prezentate în lucrarea de față se numără: descrierea tehnologiei Web utilizate, detalii de implementare vizate în aplicație precum și Arhitectură aplicației

Proiectul presupune implementarea unei aplicații server-client prin care se oferă acces la o librărie online. Cărțile vor putea fi căutate după criterii diverse (gen, autor, titlu, anul apariției, ISBN, rating) și descărcate de către client. Pe baza tuturor acestor informații (genurile în care se încadrează o carte, genurile abordate de un anumit autor, rating-ul cărții, căutările și descărcările unui utilizator), serverul va fi capabil să ofere recomandări de cărți clientului. Pe măsură ce activitatea clienților crește, sistemul de recomandări se va putea îmbunătăți în acuratețe, luând în considerare factori precum gusturile unui utilizator, preferințele altor utilizatori cu gusturi similare, rating-ul pe care l-au primit cărțile de la clienții cărora le-au fost recomandate.

Am ales proiectul prezentat deoarece am fost din totdeauna pasionat de cărți și în calitate de posibil user am considerat că această aplicație poate fi una foarte utilă pentru a găsi cu ușurință cărți pe gustul meu și pentru a putea ajută și alte persoane cu gusturi similare.

2 Tehnologii utilizate

"Transmission Control Protocol (sau TCP, în traducere liberă din engleză Protocolul de Control al Transmisiei) este un protocol folosit de obicei de aplicații care au nevoie de confirmare de primire a datelor. Efectuează o conectare virtuală full duplex între două puncte terminale, fiecare punct fiind definit de către o adresă IP și de către un port TCP.

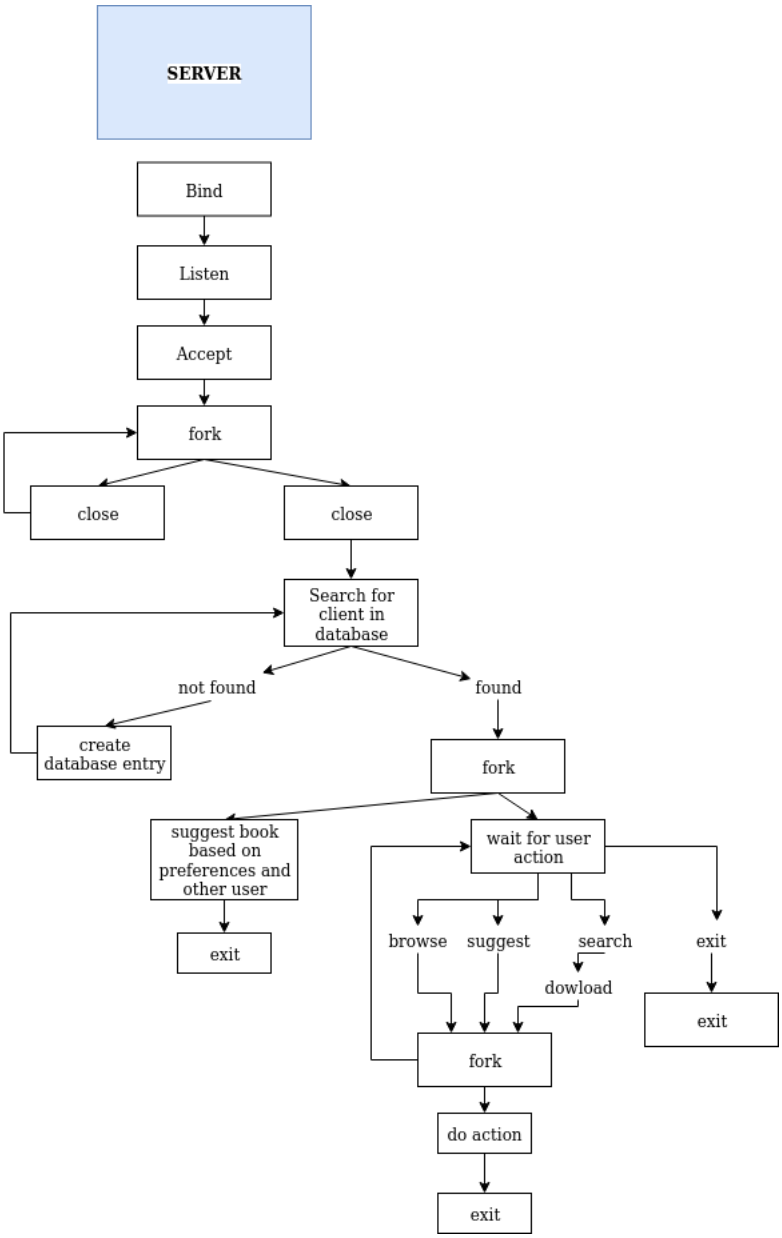
Transmission Control Protocol (TCP) este unul dintre protocoalele de bază ale suitei de protocoale Internet. TCP este unul dintre cele două componente originale ale suitei (celalalt fiind Protocolul Internet, sau IP), astfel încât întreaga suita este frecvent menționată ca stiva TCP/IP. În special, TCP oferă încredere, asigura livrarea ordonata a unui flux de octeți de la un program de pe un computer la alt program de pe un alt computer aflat în rețea. Pe lângă sarcinile sale de gestionare a traficului, TCP controlează mărimea segmentului de date, debitul de informație, rata la care se face schimbul de date, precum și evitarea congestionării traficului de rețea. Printre aplicațiile cele mai uzuale ce utilizează TCP putem enumera World Wide Web (WWW), posta electronica și transferul de fișiere (FTP).

TCP este optimizat, mai degrabă, pentru livrarea exacta decât livrarea la timp a datelor, și prin urmare, TCP înregistrează uneori, întârzieri relativ mari de timp (de ordinul secundelor), în timpul de așteptare pentru unele mesaje ce sosesc în alta ordine sau pentru retransmisia de mesaje pierdute. Acesta nu este deosebit de potrivit pentru aplicații în timp real, cum ar fi Voice over IP."[1]

Pentru acest proiect am preferat să utilizez protocolul TCP pentru nivelul transport al aplicației. Motivația acestei alegeri fiind necesitatea receptionării pachetelor fără pierderi și reconstituirea acestora în ordinea trimiterii(în caz contrar, conținutul unei cărți poate să fie alterat în mod grav). Aplicația necesitând nu de o viteză de transmisie a pachetelor mai mare ci mai degrabă de control asupra acestora.

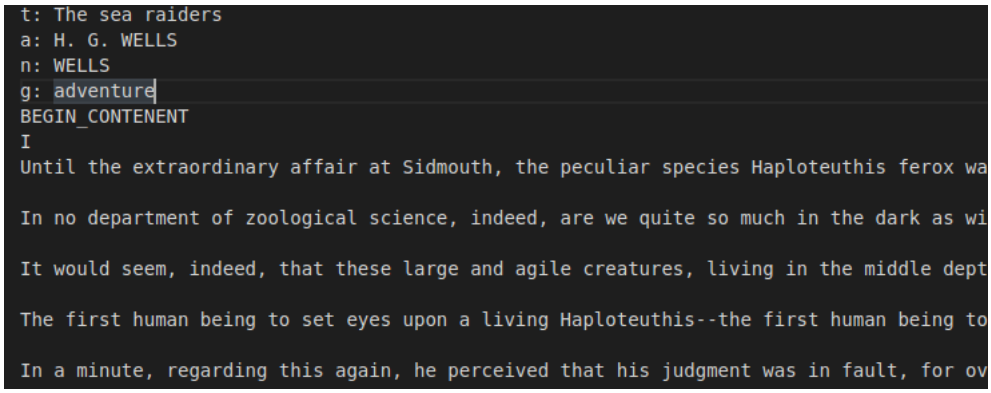
O altă tehnologie utilizată în proiect este baza de date MySQL, utitlizata pentru a stoca informații despre fiecare client(numele de utilizator,numărul și cartile descărcate) pe baza cărora se va realiza sugerarea altor cărți.

3 Arhitectura aplicatiei



4 Detalii de implementare

Fiecare carte va fi precedată de un antet precum în imagine:




```
Unde tagurile au urmatoarele semnificatii: "t:" -title
                                           "a:" -author
                                           "n:" -nickname
                                           "g:" -genres
                                           "s:" -subgenres
                                           "i:" -id
                                           "d:" -date
```

Main function for the program:

```
int main{
thData* ThreadData;
int ThreadIndex=0;

MYSQL* dbConnect;
CONNECT_TO_DATABASE_CHECK(dbConnect);\\CONNECTING TO THE DATABASE


    struct sockaddr_in server;
    struct sockaddr_in from;
int serverDescriptor;

/*create server socket*/
SOCKET_TCP_CHECK(serverDescriptor); \\CREATING TCP SOCKET
perror("[SERVER]: CREATE SOCKET");

    bzero (&server, sizeof (server));
    bzero (&from, sizeof (from));
```

```
    server.sin_family = AF_INET;     \\constructing SOCKADDR structure
    server.sin_addr.s_addr = htonl (INADDR_ANY);
    server.sin_port = htons (PORT);
BIND_CHECK(serverDescriptor,server);  \\BIND socket with socketaddr
perror("[SERVER]: BIND");

LISTEN_CHECK(serverDescriptor\\Listen
perror("[SERVER]: LISTEN");
    while (1)
    {
     int client;
     socklen_t length = sizeof (from);

ACCEPT_CHECK(client,serverDescriptor,from,length);\\accept sockaddr from client
perror("[SERVER]: ACCEPT");


ThreadData=(struct thData*)malloc(sizeof(struct thData)); \\construct thread for the client
ThreadData->idThread=ThreadIndex++;
ThreadData->client=client;
ThreadData->dbConnect=dbConnect;



pthread_t CLIENT_COMMANDS_THREAD;
pthread_create(&CLIENT_COMMANDS_THREAD, NULL, CLIENT_CMD, ThreadData); \\ create and execute
                                                        \\thread that resolve user commands


    } /* while */
mysql_close(dbConnect);  \\Disconnect from the database

}
```

Function that parse and resolve COMMANDS from the user:

```
void * CLIENT_CMD(void* arg){

thData ThreadData;
ThreadData= *((struct thData*)arg);

MYSQL * dbConnect=ThreadData.dbConnect;
int client=ThreadData.client;
int ThreadIndex=ThreadData.idThread;

int usernameSize,passwordSize,userID=0;

while(userID==0){//Login and register part
...
}
    int pid;
bool EXIT=false;
while(!EXIT){
BYTE typeOfCommand;
     READ_CLIENT_CHECK(client,&typeOfCommand,sizeof(BYTE));
switch(typeOfCommand){
case 0: {/*EXIT*/
printf("[SERVER]: CLIENT EXIT\n");
EXIT=true;
break;
}
case 1: {/*Search and Download by title*/
...
}
case 2:{/*BROWSE all or by genre,author*/
...
}
case 3:{//SUGGEST
...
}
case 4: {//RATE
...
}
}


}
    close (client);
}
```

Conectarea la Baza De date

```
int addUserToDB(MYSQL * dbConnect, const char * username){

    MYSQL_RES *res_set;
    MYSQL_ROW row;
    mysql_query (dbConnect,"select id,username from users;");
    unsigned int i =0;
    res_set = mysql_store_result(dbConnect);
    unsigned int numrows = mysql_num_rows(res_set);
    while (((row= mysql_fetch_row(res_set)) !=NULL )){
        if(strcmp(row[i+1],username)==0){
            return atoi(row[i]);
        }
    }
    char MYSQLcommand[1000];
    sprintf(MYSQLcommand,"insert into users (username) VALUES (\"%s\");",username);
    mysql_query (dbConnect,MYSQLcommand);

    mysql_query (dbConnect,"select id,username from users;");
    i =0;
    res_set = mysql_store_result(dbConnect);
    numrows = mysql_num_rows(res_set);
    while (((row= mysql_fetch_row(res_set)) !=NULL )){
        if(strcmp(row[i+1],username)==0){
            return atoi(row[i]);
        }
    }


    return 0;
}
```

Suggestion Algorithm: The proram use a combination of 4 different algorithms for finding a book that can suit the user.
1. User-Based Collaborative Filtering

The first recommender on our list is the user-based colloborative filter. This form of recommender is based on the assumption that users who have agreed in the past are likely to agree again in the future. With our user-article table, we first need to find a list of users similar to the target user. We do this by comparing the item ratings of the target user to the item ratings of every other user in the table. For example, since users 1 and 4 have both rated items 2 and 5, we can compute a similarity score between the 2 users based on those 4 ratings. Note that we are not comparing the entire user row; we are only comparing items which both users have rated. The Algorithms that i used for this is Cosine Similarity. Once we compute a similarity score between the target user and every other user in the table, we can sort these similar users (or neighbors) and keep just the most similar (or nearest) ones. This algorithm is appropriately called the K-Nearest Neighbors (KNN) algorithm. Now that we have a list of the users most similar to the target user (for which we are getting a recommendation), we can grab items read by these neighbors and recommend them to the target user.

2.Item-Based Collaborative Filtering(2 methods: based on favorite genre and author or based on second favorite genre and author)

Item-based systems work very similarily to user-based systems. When trying to find a suggestion for a given user, they find items similar to items the user has already seen.

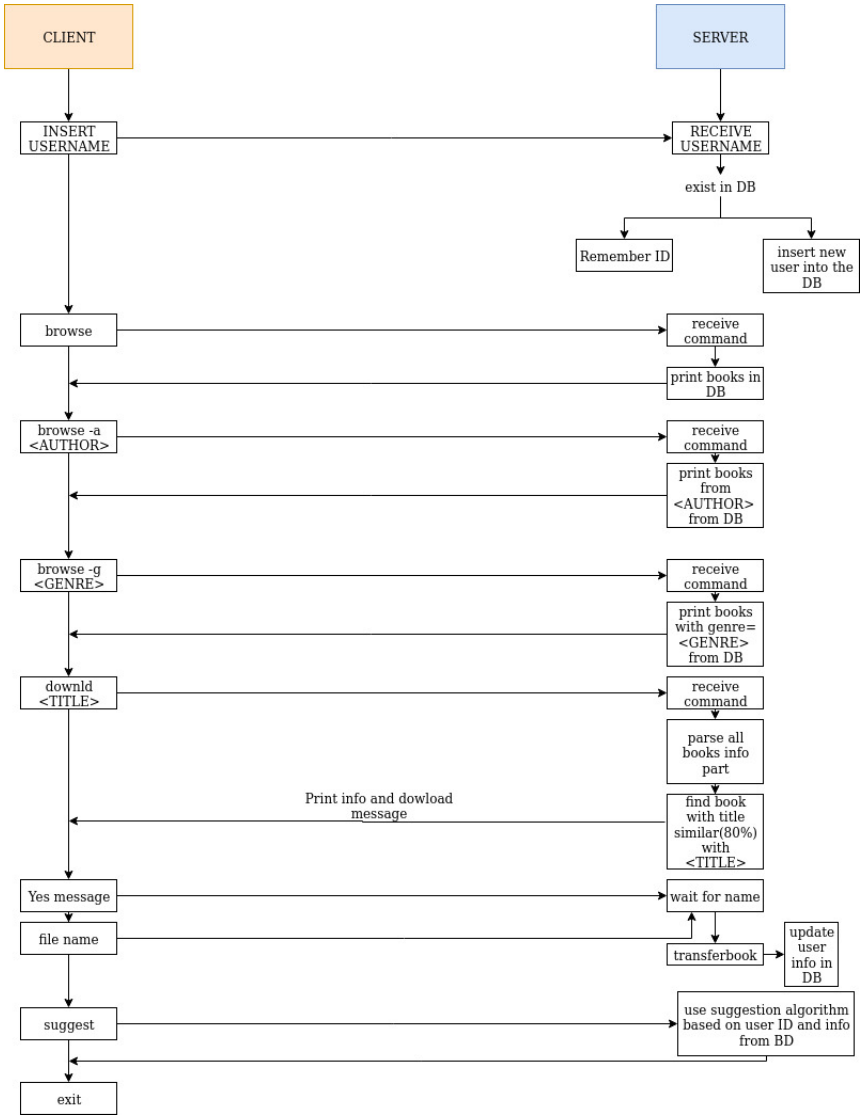
The primary difference is that instead of finding the nearest users, we find the nearest items. For example, since user 4 liked item 2, we can find items similar to item 2 (such as item 5) and suggest it to user 4.

3.Self-centered filtering

The algorithm randomly select a book not owned that has the author= favorite author(70% percentage of that to happen) or =second favorite and same with the genres.

The server combines all this method so that 40% of the time we use 1. 30% of the time we use method 2. 20% of the time we use method 3 with author and 10% of the time with genre.

5 Scenarii de utilizare



6 Concluzie

Aplicația prezentată o să fie acompaniată de o interfață grafică și pentru client cât și pentru server.Este o aplicație ce se poate dezvolta tot mai mult, prin adăugarea de cărți de către server sau chiar de către anumiți clienți(în funcție de poziția bazată pe un sistem de ranking). Precizia sugestiilor poate să fie îmbunătățită folosind un algoritm care se bazează pe rețele neuronale.

7 Bibliografie

[1] https://ro.wikipedia.org/wiki/Transmission_Control_Protocol
[2] https://www.mysql.com
[3] https://en.wikipedia.org/wiki/Cosine_similarity
[4] https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm