Ders Adı: Algoritma Analizi

Öğrenci Bilgileri: Ömer Buğrahan Çalışkan – 17011076

Ödev İçeriği: Kitap Öneri Sistemi

Günümüzde Youtube, Netflix, Amazon, Pinterestgibi internet ortamında milyonlarca kullanıcısı olan pek çok firma makine öğrenmesi tabanlı tavsiye sistemleri ile kullanıcılara kişiselleştirilmiş öneriler sunmaktadır.

Bu çalışmada işbirlikçi filtre (collaborative filtering) yöntemi ile bir kişinin önceki seçimlerine bakarak yeni kitap öneren bir sistem tasarlanacak ve gerçeklenecektir.

Kod Açıklaması:

Fonksiyonlarda bahsedilen index hangi kullanıcı ile işlem yapılacağını belirten matristeki satır indexidir.

- Average Fonksiyonu: İki matrisi, n değerini(kitap sayısı) ve NU ile U indexlerini parametre olarak alır. Burada bulunacak olan ortalama iki kullanıcının ortak okuduğu kitaplara bakılarak yani ilgili kitaba her ikisinin de verdiği puan 0 olmamak kaydıyla hesaplanmaktadır.
- Averageself Fonksiyonu: Bir matrisi, n değerini ve indexi parametre olarak alır. Burada average fonksiyonundan farklı olarak kullanıcıyı başka bir kullanıcıyla karşılaştırmadan kendi okuduğu kitaplar baz alınarak kendi ortalaması hesaplanmaktadır.
- Pearson Fonksiyonu: Her iki matrisi, n değerini ve indexleri parametre olarak alır. Pearson formülü hesaplanırken formülün üst ve alt kısmını ayrı ayrı hesapladım. Bu formül aslında Cov(x,y)/kök(var(x))* kök(var(y)) şeklinde ifade edilebilir. Bu değer hesaplanırken okuyucuların ikisinin de kitaba verdiği puanların 0 olmaması, yani kitabı okumaları gereklidir.
- Offer_book Fonksiyonu: sim matrisi(hesaplanan pearson değerlerinin en yüksek olan 3 değerini ve bu değerlerin ilgili okuyucu indislerini tutar), Her iki matrisi, NewUser indexini, karsılastırılacak kitabın indexini ve k girdisini alır. Verilen formüle uygun olarak değer hesaplanır.
- Sim_kont Fonksiyonu: sonuç değeri(benzerlik oranı), sim matrisi(en yüksek değerleri ve indexleri içeren matris), NewUser indexini, sonuç değerinin bulunduğu U kullanıcısının indexini ve k girdisini alır. Burada yapılan işlem k girdi sayısına göre matrise ilk önce karşılaştırma yapılmadan benzerlik değerlerinin konulması ardından yeni değerler geldiğinde minimum benzerlik değerinin bulunarak, eğer yeni oran bu değerden büyükse eski değerin üstüne yazılması.
- Readbooks Fonksiyonu: Books dizisi (içinde kitap isimlerini tutacak dizi) ve kitap sayısı parametrelerini alır. Csv dosyası read modunda açılır ilk satır alınır fakat ilk sütunda USERS yazdığı için bu hücre atlanır ardından kitap isimleri tek döngüde teker teker her değer strtok fonksiyonuyla ; değerleriyle ayrılır, son değerden sonra ; gelmeyeceği için book_size-2 hücresine ulaştığında satır atlama işlemini görene kadar okuma yapar.
- Readusers Fonksiyonu: User ve NewUser matrisini, User kullanıcı sayısını ve kitap sayısını alır. Csv dosyası read modunda açılır, ilk satır kitap isimlerini içerdiğinden atlanır, her bir yeni satıra gelindiğinde de ilk hücreler Kullanıcı isimlerini(U2-NU4..) içerdiğinden atlanır ardından her bir sayı değeri matris alınır.
- Main Fonksiyonu:
    1. Csv dosyasından okunacak User sayısı, NewUser sayısı, Kitap Sayısı, K girdisi alınır.
    2. Verilen değerlere uygun user_u ve user_n matrisi açılır.

3. Readusers ve Readbooks fonksiyonları çağırılarak matrise, kitap isimleri uygun değerlerle doldurulur.
4. Oneri yapılacak NU kullanıcısı sorulur.
5. Sim matrisi açılır başlangıç değerleri verilir ve pearson değerleri sırayla döngüyle hesaplandıktan sonra sim matrisinde ilgili NewUser indexine göre benzerlik matrisi oluşturulur. Bu matriste her NewUser kullanıcısının k girdisine bağlı olarak en yüksek benzerlik oranının değerleri ve NewUser ile bu değerleri elde edilen U kullanıcısının indexi bulunur.
6. Sim matrisi kullanılarak istenen NewUser kullanıcısının k değerine bağlı olarak en benzer User kullanıcılarının listesi ekranda gösterilir.
7. Ardından offer_book fonksiyonu kullanılarak, İstenilen kullanıcının okumadığı kitaplara verebileceği tahmini puanlar hesaplanır ve en yüksek puanla birlikte bu puanın verileceği kitabın indexi tutulur.

- Son adımda istenen NewUser kullanıcısına önerilecek olan kitap bastırılır.

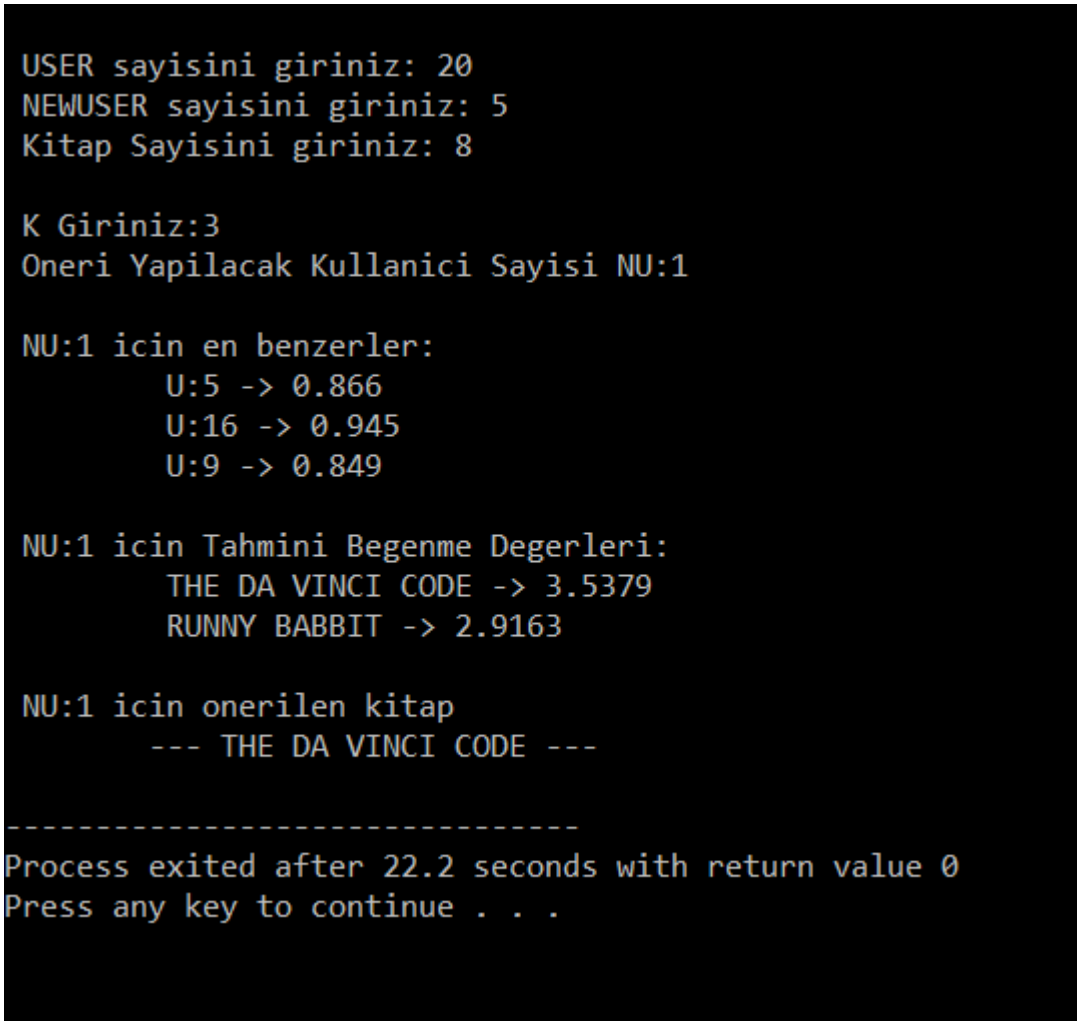
NOT: Csv dosyasında okunmayan kitapların boş hücre değerlerini 0 ile değiştirdim, Amaç Güvensan hocaya sordum ve bu şekilde yapmamın uygun olduğunu, yapabileceğimi söyledi.

Ekran Görüntüleri:

K=4 ve NU2 Kullanıcısı için Örnek Ekran Çıktısı

```
USER sayisini giriniz: 20
NEWUSER sayisini giriniz: 5
Kitap Sayisini giriniz: 8

K Giriniz:4
Oneri Yapilacak Kullanici Sayisi NU:2

NU:2 icin en benzerler:
        U:1 -> 0.945
        U:2 -> 0.982
        U:11 -> 1.000
        U:19 -> 0.866

NU:2 icin Tahmini Begenme Degerleri:
        TRUE BELIEVER -> 1.6899
        THE KITE RUNNER -> 2.3591
        HARRY POTTER -> 2.3541

NU:2 icin onerilen kitap
       --- THE KITE RUNNER ---


--------------------------------
Process exited after 4.333 seconds with return value 0
Press any key to continue . . .
```

K=3 ve NU1 Kullanıcısı için Örnek Ekran Çıktısı.



C Kodu:

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>

// NOT CSV DOSYASINDAKİ BOŞ HÜCRELERİN YERİNE 0 DEĞERİ YAZIP O ŞEKİLDE OKUMA YAPTIM AMAÇ HOCA BU ŞEKİLDE YAPILAN ÇÖZÜMÜN UYGUN OLDUĞUNU SÖYLEDİ.

float average(int n, int **A, int **B, int index_a, int index_b){
    int i;
    float sum=0;            // NU ve U okuyucularının ortak okudukları kitaba göre ortalamaları
    int count=0;
    for(i=0;i<n;i++){
```

```
            if(A[index_a][i]!=0  && B[index_b][i]!=0){
                sum += A[index_a][i];
                count++;
            }
    }
    sum /= count;
    return sum;
}

float averageself(int n, int **A, int index){
    int i;
    float sum=0;   // Okuyucunun kendi okuduğu kitaplara göre ortalaması
    for(i=0;i<n;i++)
            sum += A[index][i];

    return (sum/n);
}

float pearson(int **X, int **Y, int n, int index_x, int index_y) {
    int i=0;

    float var_x = 0;
    float var_y = 0;
    float cov = 0;
    float ort_x = average(n,X,Y,index_x,index_y);       // ilgili U kullanıcısının NU
kullanıcısıyla okuduğu ortak kitaplara göre ortalaması
    float ort_y = average(n,Y,X,index_y,index_x);       // ilgili NU kullanıcısının U
kullanıcısıyla okuduğu ortak kitaplara göre ortalaması



   while(i<n){
    if(Y[index_y][i]!=0 && X[index_x][i]!=0)  // İkisinin de kitabı okuması gerekli
            cov += ((X[index_x][i]-ort_x) * (Y[index_y][i]-ort_y));
    i++;
    }

   for(i=0;i<n;i++){
            if(Y[index_y][i]!=0 && X[index_x][i]!=0)  // Variance hesaplaması
                var_x += (pow((X[index_x][i]-ort_x),2));
     }

   for(i=0;i<n;i++){
            if(Y[index_y][i]!=0 && X[index_x][i]!=0)  // Variance hesaplaması
                var_y += (pow((Y[index_y][i]-ort_y),2));
     }
```

```
   float p_corr = cov / (sqrt(var_x) * sqrt(var_y));         // sim formülünün sonucunun
hesaplanması
   return p_corr;
}

float offer_book(float **sim, int **X, int **Y, int index_n, int book, int k, int book_size){
     int i,j;
     float ust=0, alt=0;
     int index_x=0;

     for(j=0;j<k;j++){        // pred formülünün pay kısmının hesaplanması, sim hesaplanırken
zaten en benzerlerin deger ve indexlerini sim matrisinde tutmustuk
               index_x = (int)sim[index_n][j+k];     // o yüzden degerleri direkt matristen
cekebiliyoruz.
               ust +=  sim[index_n][j] * ( X[index_x][book] - averageself(book_size,X,index_x) );
     }

     for(j=0;j<k;j++){        // pred formülünün payda kısmının hesaplanması
               alt += sim[index_n][j];
     }

     float sonuc = averageself(book_size,Y,index_n) + ust/alt;   // pred formülünün sonucunun
hesaplanması
     return sonuc;
}

void sim_kont(float sonuc, float **sim, int index, int indexofsonuc, int k){
     int i,j;
     int flag = 0;

     for(i=0;i<k;i++){
               if(sim[index][i] == -2 && flag == 0){          // benzerlik degerlerini ve bu degerlerin
hangi okuyucuya ait olduğu(index) degerlerinin matris bosken karsılastırma
                         sim[index][i] = sonuc;                                   // yapmadan direkt
atanması
                         sim[index][i+k] = indexofsonuc;
                         flag = 1;
               }
     }

     if(flag == 0){  // ilk atamalar gerceklestiyse en benzerler arasındaki minimum değer bulunur
ve eger yeni benzerlik oranımız minimumdan büyük ise üstüne yazılır
               float min = sim[index][0];
               int min_index = 0;
     for(i=1;i<k;i++){
               if(min > sim[index][i]){
                         min = sim[index][i];
```

```
                    min_index = i;
                }
        }

        if(min < sonuc){
                sim[index][min_index] = sonuc;
                sim[index][min_index+k] = indexofsonuc;
                }
        }
}

void readbooks(char **books, int book_size){
        char buffer[2048];
        char *record, *line;
        int i=0, j=0;

        FILE *fstream = fopen("RecomendationDataSet.csv","r"); // Dosyanın acılması
        if(fstream == NULL){
                printf("Dosya acilamadi");
                exit(0);
        }
        line=fgets(buffer,sizeof(buffer),fstream); // İlk satırdan kitap isimleri alınır

                record = strtok(line,";");
                record = strtok(NULL,";");

                for(j=0;j<book_size;j++){
                        books[j] = record;
                        if(j == book_size-2)    record=strtok(NULL,"\n");
            else record = strtok(NULL,";");
        }
        fclose(fstream);
}

void readusers(int **user_u, int **user_n, int size_u, int book_size){
        char buffer[2048];
        char *record, *line;
        int i=0, j=0;

        FILE *fstream = fopen("RecomendationDataSet.csv","r"); // Dosyanın acılması
        if(fstream == NULL){
                printf("Dosya acilamadi");
                exit(0);
        }
        line=fgets(buffer,sizeof(buffer),fstream);
```

```
    for(i=0;i<size_u;i++){          // csv dosyasında hücre değerleri ; ile ayrıldığından strtok ile ilk önce U okuyucularının kitaplara verdiği puanlar matrise alınır
            line=fgets(buffer,sizeof(buffer),fstream);
                record = strtok(line,";");      // ilk değer okuyucu adını icerdigiden atlanır.
                record = strtok(NULL,";");

            for(j=0;j<book_size;j++){
                user_u[i][j] = atoi(record);
                if(j == book_size-2)    record=strtok(NULL,"\n");
        else record = strtok(NULL,";");
            }
    }
            i=0;
    while((line=fgets(buffer,sizeof(buffer),fstream))!=NULL){                   // U okuyucuları okunması bittiginde bir sonraki satırdan okunacak satırlar bitene kadar NU okuyuclarının puanları matrise alınır
            record = strtok(line,";");
            record = strtok(NULL,";");

            for(j=0;j<book_size;j++){
                user_n[i][j] = atoi(record);
                if(j == book_size-2)    record=strtok(NULL,"\n");
        else record = strtok(NULL,";");
            }
            i++;
    }
    fclose(fstream);
}

int main(){
    int i,j,k,n_user;
    int size_u, size_n, book_size;
    float sonuc;

    printf("\n USER sayisini giriniz: ");
    scanf("%d",&size_u);

    printf(" NEWUSER sayisini giriniz: ");
    scanf("%d",&size_n);

    printf(" Kitap Sayisini giriniz: ");
    scanf("%d",&book_size);
    char *books[book_size];

    printf("\n K Giriniz:");
    scanf("%d",&k);
```

```
    int **user_u;
    user_u = (int**)malloc(sizeof(int*)*size_u);
            for(i=0;i<size_u;i++)
                    user_u[i] = (int*)malloc(sizeof(int)*book_size);

    int **user_n;
    user_n = (int**)malloc(sizeof(int*)*size_n);
            for(i=0;i<size_n;i++)
                    user_n[i] = (int*)malloc(sizeof(int)*book_size);

    readusers(user_u,user_n,size_u,book_size);
    readbooks(books,book_size);

    printf(" Oneri Yapilacak Kullanici Sayisi NU:");
    scanf("%d",&n_user);

    float **sim = (float**)malloc(sizeof(float*)*size_n);          // Sim matrisi k değerine bağlı
olarak en yüksek benzerlik oranlarını ve bu oranların elde edildiği indexleri tutacak
    for(i=0;i<size_n;i++)
            sim[i] = (float*)malloc(sizeof(float)*(2*k));

    for(i=0;i<size_n;i++){
            for(j=0;j<k;j++){
                    sim[i][j] = -2;
                    sim[i][j+k] = 0;
            }
    }

    for(i=0;i<size_n;i++){
            for(j=0;j<size_u;j++){
                    sonuc = pearson(user_u,user_n,book_size,j,i);         // Her bir NewUser
kullanıcısının U kullanıcılına benzerlik oranlarının hesaplanması
                    sim_kont(sonuc,sim,i,j,k);                                          // Ardından
bu degerlerin ilgili fonksiyona gönderilerek en yüksek değerlilerin bulunması
            }
    }

            printf("\n NU:%d icin en benzerler:\n",n_user);
            for(j=k;j<(2*k);j++){
                    printf("\t U:%.0f -> %.3f\n",sim[n_user-1][j]+1,sim[n_user-1][j-k]);        //
İstenilen NU kullanıcısının k girdisine bağlı olarak en yüksek benzerlikli
            }
                                                    // U kullanıcılarının listelenmesi

    printf("\n NU:%d icin Tahmini Begenme Degerleri: \n",n_user);

    float offer;                    // Her bir pred değerini tutacak değişken
```

```
    int offer_index;                    // Tutulan pred değerinin hangi kitaba ait olduğunu tutacak
değişken
    float offer_temp = -1; // Bir önceki pred değeriyle karşılaştırabilmek için tutulan temp değer
ilk adımda çalışabilmesi için -1 değeri verildi

            for(j=0;j<book_size;j++){
                    if(user_n[n_user-1][j] == 0){ // İstenilen NU kullanıcısının okumadığı
kitaplara göre tahmin oluşturulacak
                            offer = offer_book(sim,user_u,user_n,n_user-1,j,k,book_size);      //
Tahmin değerinin hesaplanması
                                    printf("\t %s -> %.4f\n",books[j],offer);
                                if(offer > offer_temp){         // Eğer öneri değeri önceki öneri
değerinden büyükse bu değer önerilen kitap olacak
                                    offer_index = j;         // Önerilecek kitabın indexi
tutulur
                                    offer_temp = offer; // Diğer değerlerle
karşılaştırabilmek için öneri değeri offer_temp değişkeninde tutulur
                                }
                            }
                    }

            printf("\n NU:%d icin onerilen kitap\n\t--- %s ---\n",n_user,books[offer_index]);//
İstenilen kullanıcıya önerilecek kitabın görüntülenmesi

    free(sim);
    return 0;
}
```