T.C.

YILDIZ TEKNİK ÜNİVERSİTESİ ELEKTRİK ELEKTRONİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

ALGORİTMA ANALİZİ

PROJE ÖDEVİ

İstanbul-2021

Hazırlayan:

Aslı Şeyda ÖZDEMİR

18011501

Dersin Yürütücüsü:

Doç. Dr. Mine Elif KARSLIGİL YAVUZ

Konu : Kitap Öneri Sistemi

Günümüzde Youtube, Netflix, Amazon, Pinterest gibi 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.

Collaborative filtering yönteminde benzerlik oranının bulunmasında kullanılan formül:

$$sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in P}(r_{a,p} - \bar{r}_a)^2}\sqrt{\sum_{p \in P}(r_{b,p} - \bar{r}_b)^2}}$$

Collaborative filtering yönteminde tahminlerde kullanılan predict formülü:

$$pred(a,p) = \overline{r_a} + \frac{\sum_{b \in N} sim(a,b) * (r_{b,p} - \overline{r_b})}{\sum_{b \in N} sim(a,b)}$$

Algoritma Yaklaşımı: Algoritmada maksimum değeri bulmak için findMax değeri kullanılmıştır. Bu fonksiyonun karmaşıklığı O(n) olup kullanıcının verdiği input değerine göre karmaşıklık k * O(n) olup k değeri küçük bir sayı olduğu için karmaşıklık O(N) değerine yakınsar.

CSV Dosyasının formatı:

İlk iki değer eski user ve yeni user sayısı olarak belirlenmiştir.

20								
5								
USERS	TRUE BELIE	THE DA VII	THE WORL	MY LIFE SC	THE TAKIN	THE KITE R	RUNNY BA	HARRY POTTER
U1	2	4	0	3	0	0	1	1
U2	0	5	0	0	3	2	1	0
U3	3	0	1	2	2	0	0	5
U4	0	3	0	0	4	1	0	3
U5	2	4	3	0	0	2	1	0
U6	5	4	0	3	1	0	3	1
U7	1	4	5	5	3	0	0	4
U8	2	2	0	0	4	5	1	0
U9	0	0	4	2	1	0	0	5
U10	3	5	1	0	0	0	4	4
U11	2	0	2	4	0	1	0	2
U12	5	4	0	2	0	1	1	3
U13	0	0	2	0	4	0	4	5
U14	0	1	2	3	4	0	5	5
U15	0	3	0	0	5	3	0	2
U16	0	3	2	1	1	0	4	0
U17	1	5	1	2	0	4	0	4
U18	5	0	4	0	2	1	3	5
U19	0	3	0	2	0	4	1	4
U20	2	5	1	1	5	4	0	4
NU1	4	0	5	3	2	3	0	4
NU2	0	5	2	5	3	0	2	0
NU3	2	1	0	0	2	3	4	1
NU4	3	5	1	1	0	1	0	3
NU5	0	2	3	2	1	0	3	0

Örnek Ekran Çıktıları:

NU1 – K: 3 için örnek ekran çıktısı:

```
---------This is a book recommendation program-------
-----------------------------------------------------

Your choices:
1.Recommendation book for user whom you choose
2.Recommendation book for all users
Enter your choice 1 or 2: 1

Enter your user for find similarity (like: NU1): NU1
Enter the K value: 3


similarities for NU1
 U16, 0.944911
 U5, 0.866025
 U9, 0.848528

THE DA VINCI CODE has ---> 3.348084 rating
RUNNY BABBIT has ---> 2.726468 rating

books that reccomended for user: THE DA VINCI CODE
-------------------------------------------
```

NU3 - K = 5 değeri için ekran çıktısı:

```
---------This is a book recommendation program-------
-----------------------------------------------------

Your choices:
1.Recommendation book for user whom you choose
2.Recommendation book for all users
Enter your choice 1 or 2: 1

Enter your user for find similarity (like: NU1): NU3
Enter the K value: 5


similarities for NU3
 U16, 0.500000
 U14, 0.498058
 U15, 0.345857
 U6, 0.114109
 U8, -0.080064

THE WORLD IS FLAT has ---> 0.724504 rating
MY LIFE SO FAR has ---> 0.971525 rating

books that reccomended for user: MY LIFE SO FAR
```

NU2 - K = 7 değeri için ekran çıktısı:

```
---------This is a book recommendation program-------
-----------------------------------------------------

Your choices:
1.Recommendation book for user whom you choose
2.Recommendation book for all users
Enter your choice 1 or 2: 1

Enter your user for find similarity (like: NU1): NU2
Enter the K value: 7


similarities for NU2
 U11, 1.000000
 U2, 0.981981
 U1, 0.944911
 U19, 0.866025
 U5, 0.755929
 U12, 0.755929
 U3, 0.755929

TRUE BELIEVER has ---> 2.781183 rating
THE KITE RUNNER has ---> 2.326902 rating
HARRY POTTER has ---> 2.947390 rating

books that reccomended for user: HARRY POTTER
```

Tüm kullanıcılarda k = 3 değeri için bulunan değerler:

```
similarities for NU1
 U16, 0.944911
 U5, 0.866025
 U9, 0.848528

THE DA VINCI CODE has ---> 3.348084 rating
RUNNY BABBIT has ---> 2.726468 rating

books that reccomended for user: THE DA VINCI CODE
-------------------------------------------------

similarities for NU2
 U11, 1.000000
 U2, 0.981981
 U1, 0.944911

TRUE BELIEVER has ---> 2.344468 rating
THE KITE RUNNER has ---> 2.028139 rating
HARRY POTTER has ---> 2.021630 rating

books that reccomended for user: TRUE BELIEVER
-------------------------------------------------

similarities for NU3
 U16, 0.500000
 U14, 0.498058
 U15, 0.345857

THE WORLD IS FLAT has ---> 0.761732 rating
MY LIFE SO FAR has ---> 0.760287 rating

books that reccomended for user: THE WORLD IS FLAT
```

```
similarities for NU4
 U2, 1.000000
 U13, 1.000000
 U10, 0.956183

THE TAKING has ---> 1.402734 rating
RUNNY BABBIT has ---> 2.019993 rating

books that reccomended for user: RUNNY BABBIT
-----------------------------------------------

similarities for NU5
 U9, 0.981981
 U18, 0.866025
 U7, 0.852803

TRUE BELIEVER has ---> 0.801639 rating
THE KITE RUNNER has ---> -0.796736 rating
HARRY POTTER has ---> 3.566851 rating

books that reccomended for user: HARRY POTTER
```

Ödevin Kodu

```
#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<math.h>

int **oldUserData; //eski kullanıcıların verilerini tutan dizi

int **newUserData; //yeni kullanıcıların verilerini tutan dizi

double *maxArr;//dizi kopyalamak için kullanılacak dizi

int count = 0, inputValue; //kullanıcıdan bir kere alınıp program boyunca
değişmeyecek değerler.

int trainData, data;//dosyadan alınan eski ve yeni okuyucu sayısı

//okunmuş olan kitaplar için aritmetik ortalamayı bulan fonksiyon

double arithmetic_meanv2(int * data, int size) //boyut ve diziyi parametre
olarak alır

{

    double total = 0; //toplam değerin saklanacağı değişken

    int count = size; //ortalamada boyut

    int i;

    for(i = 0; i < size; i++){
```

```
      if(!data[i]){ //eğer okunmadıysa
            count--;
            }
      else
            total += data[i]; //okunduysa
      }
    return total / count; //aritmetik ortalamayı döndürür.
}
//predict için ikisinin de okuduğu ortak kitapları bulan art. ort.
double arithmetic_mean(int * a, int *b, int size)
{
    double total = 0;  //toplam değerin saklanacağı değişken
    double reg = size;  //ortalamada boyut
    int i;
    for(i = 0; i < size; i++){
      if(a[i] * b[i] == 0){ //eğer okunmadıysa
            reg--;
            }
      else
            total += a[i]; //okunduysa
      }
    return total / reg; //aritmetik ortalamayı döndürür.
}
//iki okuyucu arasındaki similarity oranını bulan fonks.
double sim(int a[count], int b[count]){
      double mean1 = arithmetic_mean(a,b, count); //ortalama hesabı
      double mean2 = arithmetic_mean(b,a, count); //ortalama hesabı
      int i;
      double cov = 0.0, normalize_1 = 0.0, normalize_2 = 0.0, similar =
0.0;
      for(i = 0; i < count; i++){
            if(a[i]*b[i] == 0); //ikisinden biri rate edilmediyse
            else{
            cov += (a[i] - mean1) * (b[i] - mean2);//covarience bulur
```

```
            normalize_1 += pow(a[i] - mean1, 2);//payda için normalizasyon
işlemleri
            normalize_2 += pow(b[i] - mean2, 2);      //payda için
normalizasyon işlemleri
            }
      }
      similar = cov / (sqrt(normalize_1) *
sqrt(normalize_2));//similaritynin bulunması
      return similar;
}
//kitap tahmini yapan fonksiyon
void predict(int index, int closest[inputValue], char books[count][30]){
//kitapları, kullanıcının indeksini, en yakın değerleri parametre olarak
alan fonks.
      double meanofNewUser = arithmetic_meanv2(newUserData[index], count);
//yeni kullanıcı için aritmetik ortalamayı bulmaya yarar
      double sum = 0, divide, max; //predict için kullanılan değişkenler
      int indexv2 = 0, i, j;
      for(i = 0; i < count; i++){
            if(newUserData[index][i] == 0){
                  indexv2++;  //kaç tane okumadığı kitap olduğunu bulur.
            }
      }
      int *booksNotRead = booksNotRead = (int *)malloc(indexv2 *
sizeof(int));
      indexv2 = 0;
            for(i = 0; i < count; i++){
            if(newUserData[index][i] == 0){
                  booksNotRead[indexv2] = i; //okunmayan kitapların
bulunmasını sağlar
                  indexv2++;
            }
      }
      double pred[indexv2][indexv2]; //kitabı ve o kitap için predict
değerini tutacak bir matris tanımlanır.
      for(j = 0; j < indexv2; j++){
            for(i = 0; i < inputValue; i++){
                  //predictionın yapılabilmesi için kullanılır.-->verilen
tahmin fonksiyonuna göre
```

```
                  sum += sim(newUserData[index], oldUserData[closest[i]]) *
(oldUserData[closest[i]][booksNotRead[j]] -
arithmetic_meanv2(oldUserData[closest[i]], count));
                  divide += sim(newUserData[index],
oldUserData[closest[i]]);
            }
            pred[0][j] = (sum / divide) + meanofNewUser; //predict değeri
            pred[1][j] = booksNotRead[j]; //tavsiye edilecek kitap.
            sum = 0;
            divide = 0;
      }
      printf("\n");
      max = pred[0][0]; //en yüksek oranlı kitap için ilklendirme
      index = 0;
            for(i = 0; i < indexv2; i++){
                  printf("\n%s has --->", books[(int)pred[1][i]]);
//kitabı yazdırır
                  printf(" %f rating", pred[0][i]);  //kitabın oranını
yazdırmaya yarar
                  if(max < pred[0][i]){
                        max = pred[0][i]; //max değerinin bulunması
                        index = i;
                  }
            }
      printf("\n\nbooks that reccomended for user: %s",
books[(int)pred[1][index]]); //kitabın önerilmesi
      printf("\n-----------------------------------------------");
}
//maksimum değer bulan fonksiyon
int findMax(int size){
      int c, location = 0;
      int number = 0;
      for (c = 1; c < size; c++)
    if (maxArr[c]> maxArr[location]){ //büyüklük karşılaştırması -> O(N)
complexity
      location = c;
      number = location;
```

```
      }
     maxArr[location] = -99; //bir sonraki max işleminde aynı değere
bakılmamasını sağlar.
     return number; //indisi döndürmeye yarar
}
//similarity için gerekli ayarlamaları yapan fonksiyon
int findSimilarity(int data, char similarityUser[10], char
books[count][30],char oldUsers[data][count], char
newUsers[trainData][count]){
      int i = 0, index, loop = 0;
      int closest[inputValue]; //en yakın değerleri tutacak  dizi
      double *similarity, *tempSimilarity, distance;
      similarity = (double*)malloc(data * sizeof(double)); //benzerlik
oranlarını tutacak dizi
      maxArr = (double*)malloc(data * sizeof(double));
      while(strcmp(similarityUser, newUsers[i])) {
            i++; //yeni kullanıcının bulunması
      }
      index = i;
      for(i = 0; i < data; i++){
            similarity[i] = sim(oldUserData[i], newUserData[index]); //
benzerliğin bulunması
      }
      for(loop = 0; loop < data; loop++) {
      maxArr[loop] = similarity[loop]; //kopyalama işlemi
   }
   for(i = 0; i < inputValue; i++){
            closest[i] = findMax(data); // max bulmaya yarar benzerlik
bulduğumuz için karmaşıklık sayi * o(n) kadar olup sonuçta O(n)
getirecektir
   }
   printf("\n\nsimilarities for %s", similarityUser);
   for(i = 0; i < inputValue; i++){
            printf("\n %s, %f", oldUsers + closest[i],
similarity[closest[i]]); //benzerliklerin ve kitapların ekrana verilmesi.
   }
   predict(index, closest, books);
}
```

```
void readFile(char similarityUser[10], int n){
      FILE *fp = fopen("recdataset.csv", "r");  //dosyanın okunmak üzere
açılması
      int i = 0, control = 0, olddata = 0;
      char line[256], booksLine[256], name[10]; //dosyadan satır satır
okumaya yarayacak değişken tanımları
      int row = 0, columns, number;
    if (fp == NULL) {
        printf("Cannot open file \n");//dosya açılmadı
        exit(0);
      }
      //eski kullanıcı sayısının alınması
      fgets(line, sizeof(line), fp);
      data = atoi(line);
      //yeni kullanıcı sayısının alınması
      fgets(line, sizeof(line), fp);
      trainData = atoi(line);
      //kitap satırlarının alınması
      fgets(line, sizeof(line), fp);
      strcpy(booksLine, line);
      char* token = strtok(line, ";");
      token = strtok(NULL, ";");
    while (token != NULL) {
        count++; // kitap sayısının bulunması
        token = strtok(NULL, ";");  // ilerleme
    }
    char books[count][30]; //kitapların alınması için değişken
    i = 0;
    token = strtok(booksLine, ";");
      token = strtok(NULL, ";");
    while (token != NULL) {
      strcpy(books[i], token); //kitapların alınması
            i++;
        token = strtok(NULL, ";");
```

```
    }
    for(i = 0; i < strlen(books[count -1]); i++);
    books[count-1][i-1] = NULL; //son kitaptaki \n karakterinin atılması
    oldUserData = (int **)malloc(data * sizeof(int*)); //eski kullanıcı
sayısına göre memory allocation'ı
      for(i = 0; i < data; i++)
            oldUserData[i] = (int *)malloc(count * sizeof(int));//eski
kullanıcılar için memory allocation'ı


      newUserData = (int **)malloc(trainData * sizeof(int*)); //kullanıcı
sayısına göre memory allocation'ı
      for(i = 0; i < trainData; i++)
            newUserData[i] = (int *)malloc(count * sizeof(int));//yeni
kullanıcıların verilerini saklayacak memory allocation'ı
      char oldUsers[data][count]; //eski user isimlerin alınması için dizi
      char newUsers[trainData][count]; //yeni user isimlerin alınması için
dizi
      while (fgets(line, sizeof(line), fp)) {
            i = 0;
            token = strtok(line, ";");
            strcpy(name, token); //ilk sütundaki ismin alınması
            token = strtok(NULL, ";");
            while (token != NULL) {
                  if(row < data){ //old userların alınması
                  number = atoi(token); //alınan değer string olduğu için
cast işleminin yapılması
                  strcpy(oldUsers[row], name); //eski userların adının
saklanması
            oldUserData[row][i] = number; //eski userların verilerinin
saklanması
                  i++;
            token = strtok(NULL, ";");
            }
            else{
                  number = atoi(token); //alınan değer string olduğu için
cast işleminin yapılması
                  strcpy(newUsers[row - data], name); //ismin verilmesi
                  newUserData[row - data][i] = number; //verilerin
saklanması
```

```
                        i++;
                token = strtok(NULL, ";");
                }
        }
              row++;
        }
        if(n == 1){
              findSimilarity(data, similarityUser, books, oldUsers,
newUsers); //benzerlik bulmaya yarayan fonks.->spesifik bir kullanıcı için
        }
        else{
              for(i = 0; i < trainData; i++)
                    findSimilarity(data, newUsers[i], books, oldUsers,
newUsers); //benzerlik bulmaya yarayan fonks.->tüm kullanıcılar için
        }
}


int main(){
        char similarityUser[10]; //yeni kullanıcının alınmasını sağlayan
tanımlama
        int choice = 0, n; //kullanıcının programa devam etmek isteyip
istememesini alan değişken
        printf("---------This is a book recommendation program-------\n");
        printf("-----------------------------------------------------\n");
        do{
              count = 0;
              printf("\nYour choices:\n1.Recommendation book for user whom
you choose\n2.Recommendation book for all users");
              printf("\nEnter your choice 1 or 2: ");
              scanf("%d", &n);
              if(n == 1){
                    printf("\nEnter your user for find similarity (like:
NU1): ");
                    scanf("%s", similarityUser);
                    printf("Enter the K value: ");
                    scanf("%d", &inputValue);
                    readFile(similarityUser, n);
```

```
            }
            else if(n == 2){
                  printf("\nEnter the K value: ");
                  scanf("%d", &inputValue);
                  readFile(similarityUser, n);
            }
            else{
                  printf("\nyou entered wrong value ");
                  return 0;
            }
            printf("\n Do you want to exit if yes press 1 else press 0: ");
            scanf("%d", &choice);
      }while(choice == 0);
      return 0;
}
```