Film ÖneriSi Sistemi

Burcu Çeltik

28 Aralık 2024

1. **KODUN AMACI**

C dilinde yazılmış bir koddur. Bize verilen u.data dosyası içindeki film kullanıcı arasındaki ilişkiyi işleyerek bipartie graph oluşturur. Buradaki ana amaç oluşturduğumuz bipartie graph incelenerek kullanıcılar ve filmler arasındaki ilişkilere göre istenen kullanıcı için farklı yöntemlerle izlemediği filmler içinden öneri yapmaktır. Bu öneriyi 5 farklı şekilde yapar bu öneri yöntemleri raporumuzun ilerleyen kısımlarında detaylı olarak açıklanacaktır.

1. **KODUN İÇERİĞİ**

Kodum iki tane c dosyasından oluşmaktadır. İlk dosyamda ilk 4 soru bulunmaktadır. Diğer dosyamda da benim yaptığım algoritma ve rassal yürüyüş hesabıyla öneri algoritması bulunmaktadır.

İlk olarak u.data’da bulunan bilgileri okuyarak interactions.txt oluşturulur. Garph da bu interactions.txt dosyası okunarak oluşturulur.

1. **VERİ YAPISI**

Bipartie Graphımı array-linked list kullanarak oluşturdum. Array’ime en yukardan başlayarak 943 tane kullanıcıyı sonra da 1683 tane filmi yerleştirdim. Adjency list kullanarak array ile linked listi birbirine bağladım. Linked list’te kullanıcılar için onların izledikleri filmleri ve onlara verdikleri ratingleri tuttum. (Yani her kullanıcının izlediği film için node yapım var, bu yapı movie ıd ve rating değerlerini tutuyor) Filmlerde ise onları izleyen kullanıcıları tuttum. Bu yapı sayesinde kullanıcıdan filme filmden de kullanıcıya çift yönlü geçiş sağlamış oldum. Linked- list linked list kullanmadım çünkü kullanıcı+ film sayısını zaten biliyoruz o yüzden baştan hafızada belirli bir boyutta array oluşturabiliriz. Array ile bir filme ulaşmak O(1) kadar sürede yapılabiliyor ama linked list olsaydı O(n) kadar sürerdi. Linked-list linked-list yapısını önceden düğüm sayısını bilmediğimiz durumlarda kullanmak daha verimli olabilir.

1. **ÖNERİ YÖNTEMLERİ**

* Rastgele film önerme

void recommendRandom(int user, int rec)

watched[] isimli bir dizimiz var boyutu movies+1 ve başlangıçta sıfır. Linked list aracılığıyla kullanıcının izlediği filmler bulunur ve onlar watched[] dizisine 1 olarak işaretlenir, sonra izlememiş filmler totalmovies[] dizisi içerisine kaydedilir. En son olarak da bu izlenmeyen filmler arasından rand() fonksiyonu sayesinde rastgele olarak istenen sayıda öneri yapılır.

* En yüksek dereceli filmi önerme

void recommendHighDegree(int user, int rec) {

Derecesi en yüksek film demek en çok izlenen film demek. Yani hangi filmin en çok sayıda kullanıcı izlemişse derecesi en yüksek film odur o filme verilen ratinglerin hiçbir önemi yoktur. Random önermede de olduğu gibi bizim öneri almak istediğimiz kullanıcının izlediği filmler watched[] dizisinde tutulur bu sayede kullanıcının izlediği bir filmi önermemiş oluruz. Filmlerin derecesini movieDegree[] adlı bir dizi içerisinde tutarız. Başlangıçta hepsinin izlenme sayısı sıfırdır. Bipartie Graph’ımın en başındaki kullanıcıdan başlayarak izlediği filmlere tek tek bakılır ve o filmlerin hepsinin numarası movieDegree[] dizisi içerisinde bir arttırılır. Bütün kullanıcılar bitince her filmin derecesi bulunmuş olunur. Benim graphımın veri yapısını düşünürsek bu işlemi farklı bir şekilde de yapılabilirdim. Dizimi taramaya 1. Numaralı filmden başlardım. O filme bağlı olan linked listi dolaşarak toplam kaç kullanıcı onu izlemiş onu kaydeder sonra dizimin bir sonraki kutucuğuna geçerdim.

Her filmi kaç kişinin izlediğini yani derecelerini bulduktan sonra bu filmleri derecelerine göre en yüksekten en düşüğe sıralanır. Sonra en yukardan başlayarak eğer o film bizim watched[] dizimiz içinde yoksa (kullanıcı izlememişse) önerilir. Toplam istenen öneri sayısı kadar öneri yapılır.

* En benzer kullanıcının beğendiği filmlerden öneri yapmak

int findMostSimilarUser(int targetUser)

ilk olarak yine diğer fonksiyonlarda da olduğu gibi öneri almak istediğimiz kullanıcının izlediği filmleri watched[] adlı bir dizide tutuyoruz sonra bipartie graphımızı oluştururken oluşturduğumuz array’in ilk kullanıcısından başlıyoruz ve o kullanıcının izlediği filmlere bakıyoruz eğer watched[] dizisi içerisinde bulunan herhangi bir filmi izlemişse commonMovies[] adlı dizimizdeki o kullanıcı için olan kutucuktaki değeri bir arttırıyoruz. En başta bu dizideki her değer sıfır. O kullanıcının bütün filmlerine bakıp karşılaştırdıktan sonra diğer kullanıcıya geçiyoruz. Bizim öneri istediğimiz kullanıcı dışındaki bütün kullanıcılara bu şekilde bakıp ortak film sayısını hesaplıyoruz. En sonunda commonMovies[] adlı dizide ilerleyerek en yüksek numaralı değeri tutan indexi buluyoruz. Bu numaralı kullanıcı bizim öneri almak istediğimiz kullanıcının izlediği filmleri en çok izleyen kullanıcı oluyor. Ve bu kullanıcı = targetUser diyoruz.

void recommendFromSimilarUser(int targetUser, int similarUser, int rec)

Bu fonksiyonda da yine watched[] dizisini kullanıyoruz bizim öneri almak istediğimiz kullanıcının izlediği filmleri tutuyor. Sonra graphımıza gidip benzer kullanıcının izlediği filmleri gösteren linked-list’i dolaşmaya başlıyoruz. Bu listede 5 puan verilen ve watched[] dizisinde olmayan yani öneri yapılacak kullanıcının izlemediği filmleri arıyoruz o şekilde bir film buldukça öneriyoruz. İstenen sayıda öneriye ulaşınca bitiriyoruz. Eğer hiç 5 puan verilen film yoksa 4 puan verilenlerden öner diye bir bölüm ekleyecektim ama baya denememe rağmen sadece 1 tane kullanıcıda bu sorunu gördüm kodu daha da uzatmamak için böyle bir ekleme yapmadım. Birçok kullanıcının 5 puan verdiği bir sürü film bulunuyordu. Eğer benzer kullanıcının 5 puan verdiği film yoksa öneri yapmıyor ama dediğim gibi bu veri setinde karşılaşacağımız bir sorun değil.

* Hedef Kullanıcıya en yakın ağırlıklı uzaklığa sahip filmlerden önerme

void dijkstra(Graph \*graph, int src, float \*distances)

Bipartie Graphımı oluştururken weight= 1/rating yapmıştım bu sayede en yüksek ratingli olan filmler en yakın mesafede olmuş oldu. Bu fonksiyonda ilk başta hedef kullanıcı hariç bütün nodların uzaklığı sonsuz yapıldı. Sonra en yakın filme gidildi ona minDist değişkeni içinde tutarız sonra onun bütün komşularına gidilip o komşuların uzaklığı hedef kullanıcıdan il noda olan uzaklık + o noddan bu düğüme olan uzaklık oldu. EN son bizim minDist içinde tuttuğumuz düğüm visited olarak işaretlendi visited olanların uzaklığı bir daha değiştirilmiyor. Sonra en yakın diğer düğüme gidildi onun için de aynı işlemler yapıldı. Mesela bir düğüm visited olarak işaretlenmemiş ve ona bulduğumuz yeni uzaklık eski uzaklığından küçükse bulunan uzaklık artık onun uzaklığı olarak değiştirilir. Bu şekilde değişimler yapılaraka en sonunda bütün düğümler visited olarak işaretlenir. Ağırlıkları yani uzaklıkları da distances[] adlı dizide tutarız. Bu mesafeler bütün düğümler visited olana kadar güncellenir.

void recommendNearestMovies (Graph \*graph, int userId, bool \* watchedMovies, int rec)

Dijkstra çalıştırılır. Kullanıcının izlemediği filmler moviDistences[] dizisine kaydedilir o filmlerin ıdleri de tutulur. Bu tutulan mesafelerden en kısa olanlar belirlenir sonra bu en yakın filmler closestMovies[] adlı dizide saklanır. Çoğu durumda en yakın mesafedekiler 1/5+1/5+1/5 = 0.6 mesafedeki filmler olur. Sonra bu filmler içinden rastgele seçer. Rastgele seçme yaptım çünkü diğer durumda hep daha küçük numaralı filmlerden öneriyordu. Mesela 0.60 uzaklıkta olan 1, 3, 70, 900 numaralı filmler varsa 2 öneri yap dediğimizde 1 ve 3 öneriliyor. Bu şekilde bir öneri çıktısı bana güzel gelmediği için en yakınları bulup bir dizi içinde saklayıp rastgele seçerek öneri yapmak istedim.

1. **BENİM ÖNERİ YÖNTEMİM**

İkinci c dosyamda benim yaptığım algoritma ve rassal yürüyüş mesafesi ile öneri yapan algoritmam bulunuyor.

Kendi yaptığım algoritmada adjency matrices kullandım. Normalde çok büyük boyutlu veri setleri inceleniyorsa pek önerilen bir yöntem değil. Bizim veri setimizde 943 kullanıcı ve 1682 film bulunuyor. Matris ile 943 x 1682= 1586626 tane hücre oluşturuyoruz. Her hücre bir tane integer boyutu kadar yer kaplıyor (4 bayt) yani 6 MB kadar bellek kullanıyoruz. Çoğu kutu da boş olduğundan bu kadar yer kaplamasına değmez diye düşünebiliriz ama bu yapı kullanarak bir kullanıcının bir filmi izleyip izlemediğini kontrol etmek O(1) kadar sürede hesaplanabilir. Çok hızlı bir şekilde işlem yapabildiğimiz için bu kadarlık alan kaplamanın bizim için bir sorun yaratmayacağını düşündüğümden kodumun bu kısmında matris kullanmak istedim.

3 tane adjency matrices yapısı kullandım.

-int bipartite\_matrix[MAX\_USERS][MAX\_MOVIES];

Kullanıcının izlediği filmlere karşılık gelen kutucuklarda 1 diğerlerinde 0 yazar.

void build\_bipartite\_matrix(const char \*filename) fonksiyonu u.data’dan aldığı verilerle kullanıcının izlediği filmlerle olan ilişkisini gösteren matrisi oluşturur.

-int movie\_genres[MAX\_MOVIES][MAX\_GENRES];

void build\_movie\_genres(const char \*filename) fonksiyonu u.item’dan aldığı verilerle filmlerin türlerini gösteren matrisi oluşturur.

-int user\_genre\_count[MAX\_USERS][MAX\_GENRES];

bipartite\_matrix içinden kullanıcının izlediği filmler için gidip movie\_genres matrisinden filmin türünü bulur ve bunu kullanıcının hangi türden kaç tane izlediğini tutan matrisimize kaydeder. Kullanıcının hangi türden kaç tane izlediğini hesaplamış oluruz.

Bu matriste her kullanıcı için sütunlar karşılaştırılarak hangi türden en fazla izlediği bulunabilir.

recommend\_movies fonksiyonuna istenen kullanıcı, kaç tane öneri istendiği ve demin bulduğumuz en çok izlediği tür değişkenleri verilir.

Sıradan kullanıcının izlemediği ve en sevdiği türe ait olan bir filmi bulur sonra tüm kullanıcıları tarayarak ona verilen tüm puanlardan onun ortalama ratingini hesaplar. Sonra diğer filme gider bu şekilde tüm filmler taranmış olur sonra bu imdb puanlarını sıralar ve en yukardan başlayarak istenen kadar öneri yapar.

1. **ZAMAN VE MEKAN KARMAŞIKLIĞI**

**recommendRandom :**Teorik time complexity: O(1)

10 film önerisi için geçen süre: 0.000040 saniye

100 film önerisi için geçen süre: 0.000079 saniye

1.000 film önerisi için geçen süre: 0.000188 saniye

-Deneysel sonuca göre time complexity: O(1)

-Mekan karmaşıklığı: İki tane array kullanıyoruz. Biri kullanıcının izlediği filmleri biri izlemediği filmleri tutuyor. O(n)

**recommendHighDegree:** Teorik time complexity: O(n)

10 film önerisi için geçen süre: 0.000037 saniye

100 film önerisi için geçen süre: 0.000295 saniye

1.000 film önerisi için geçen süre: 0.002741 saniye

-Deneysel sonuca göre time complexity: O(n)

-Mekan karmaşıklığı: İki tane array kullanıyoruz. Biri kullanıcının izlediği filmleri biri ise en yüksek dereceli (en çok izlenmiş) filmleri tutuyor. O(n)

**addEdge**:bu fonksiyonda bir kenar eklemek O(1) ama biz bunu asıl createGraph fonksiyonu içinde çağırıp kullanıyoruz onun time complexity ise O(n).

100 Kenar Eklemek için geçen süre: 000015 saniye

1.000 Kenar Eklemek için geçen süre: 0.000080 saniye

10.000 Kenar Eklemek için geçen süre: 0.000762 saniye

-Deneysel sonuca göre time complexity: O(n)

-Mekan karmaşıklığında addEge de createGraph da O(n).

Edge eklerken mekan karmaşıklığı edge yani kenar sayısına bağlı createGraph fonksiyonunda ise toplan vertice yani düğüm sayısına bağlı.

**recommend\_movies**:(Benim yaptığım algoritma) Teorik time complexity: O(n^2)

1 film önerisi süresi: 0.001144 saniye

10 film önerisi süresi: 0.003951 saniye

100 film önerisi süresi: 0.001095 saniye

Bu hesaplamalara göre time complexity 1 ve 10 arası o(n^2) sonra o(n) çıkıyor

Birden fazla kullanıcı için süre hesabı yaptım ve öneri sayısı arttıkça sürelerin değişimi farklılık gösteriyor bu veri setindeki film kullanıcı ilişkisinin düzenli olmamasından kaynaklanıyor olabilir sonuçta bazı yerlerde daha fazla yığılma olabilir. Elimizde düzenli bir veri seti yok.

Mekan Karmaşıklığı: O(n)

**dijkstra**: teorik time complexity: O(n^2)

Mekan karmaşıklığı: mesafeleri ve ziyaret edilenleri tutuyoruz O(n)