Projenin Amacı ve Kapsamı

- Bipartite graph yapısını C dilinde modellemek
- MovieLens-100K veri setini kullanarak basit öneri algoritmaları tasarlamak

Öneri Sistemleri Nedir?

Teknolojinin ve internetin gitgide gelişmesinden dolayı internette kullanıcılara sunulan ürünlerin yelpazesi genişlemektedir. Bu ürünler film, dizi, müzik, video ve oyun gibi internette bulunabilecek her şeydir. Bunların miktarı günden güne artmaktadır. Bu yüzden teknoloji şirketlerinin kullanıcıların ilgisini çekmesi için daha da fazla uğraşması gerekmektedir. Bunun için de öneri sistemleri kullanılmaktadır. Öneri sistemlerinin temel amacı kullanıcıya, ilgisini en çok çekecek ürünü sunmaktır. Bu işlemi yapmak için birden fazla algoritma bulunmaktadır. Örneğin; öneri sistemleri kullanıcıların davranışlarını, alışkanlıklarını, satın aldığı eşyaları, izlediği filmleri, dinlediği müzikleri vb. ürünleri analiz eder ve bir kullanıcıya, bu özelliklerinin en çok benzediği kullanıcının kullandığı ama kendisinin henüz kullanmadığı bir ürün önerebilir. Bir başka yöntemse kullanıcının kullandığı ürüne en çok benzeyen ürünleri önermektir. Bu gibi yöntemlerin tümüne öneri algoritmaları denir.

Yukarıda anlatıldığı üzere, öneri sistemleri teknoloji şirketlerinin varlıklarını sürdürebilmesi için önem arz etmektedir. Çünkü kullanıcı, kendisine en iyi öneriyi sunabilen bir şirketin ürünlerini kullanmak isteyecektir. Bu yüzden öneri sistemlerini iyi kullanabilen şirketler diğer şirketleri ezip pazarı tekeline alabilir.

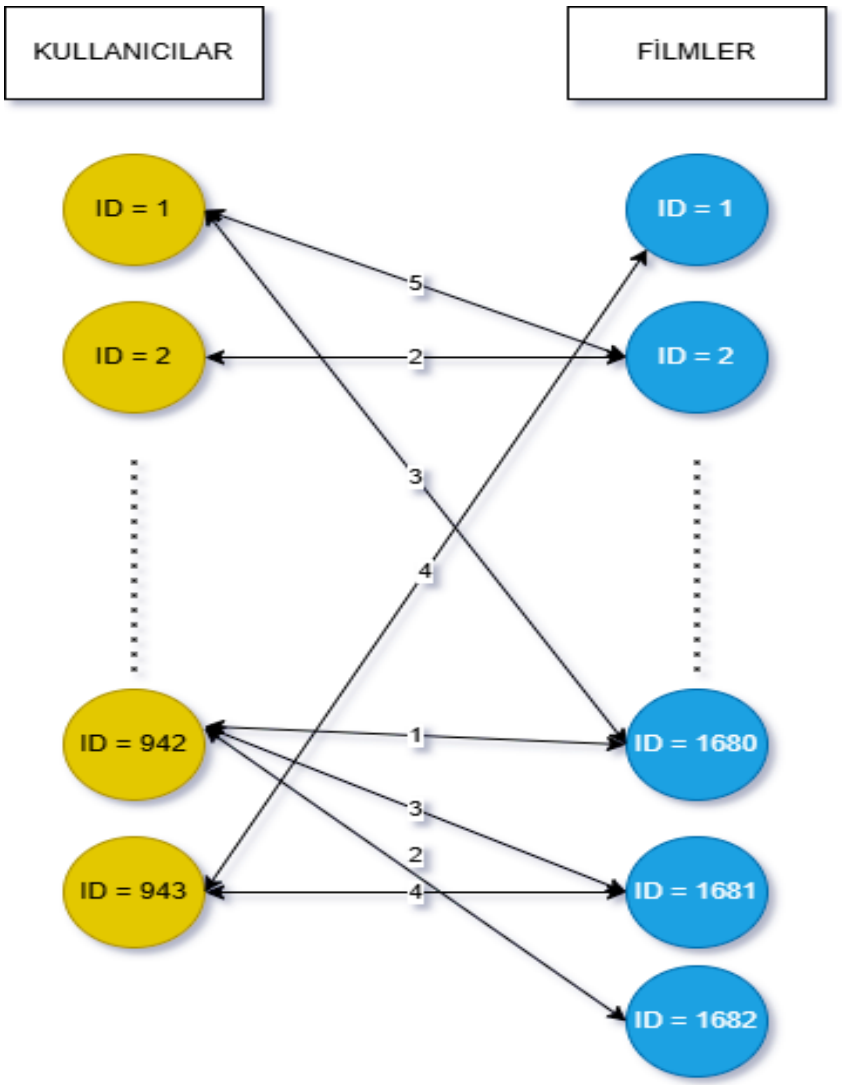
PROJE TASARIMI

Bipartite Graph Nedir?

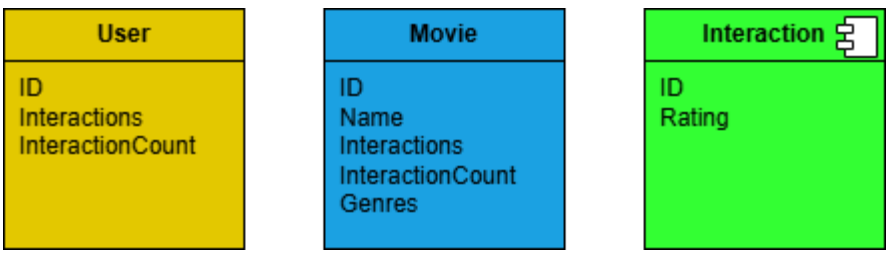
Bipartite graph, graphların özel bir durumudur. İki farklı türde düğüm vardır ve aynı tür düğümler arasında bağlantı yoktur. Bipartite graphlar öneri sistemlerinde kullanılabilir. Bir grubun kullanıcılar diğer grubun ise öğeler olduğunu kabul edersek, bir kullanıcı bir öğeyi puanlandırırsa, aralarında bir bağlantı oluşur. Bu bağlantı kullanıcılara, beğendikleri öğelere göre öğeler önerilmesine yardımcı olur.

Bu projede veri yapısı Bipartite graph şeklindedir. İlk küme kullanıcıları(users), ikinci küme filmleri(movies) temsil etmektedir. Bu projede bipartite graphın özellikleri aynen geçerlidir;

- Sadece iki farklı türde düğüm içerecek. (users ve movies)
- Aynı tip düğümler arasında bağlantı olmayacak. (user-user veya movie-movie etkileşimi bulunmayacaktır.)
- Farklı tip düğümler arasında bağlantı yönsüz olacaktır.
- Bağlantıların ağırlığı olacaktır. (Bu projede ağırlık, kullanıcının filme verdiği puandır)



STRUCTLAR



Veri Yapısındaki Bazı Tanımlamalar

u, users kümesinden bir kullanıcı ve i, movies kümesinden bir film olmak üzere;

- Bir düğümün derecesi o düğümün toplamda bağlı olduğu düğüm sayısını ifade eder.
- *u* ve *i* şeklindeki iki düğüm arasındaki yol uzunluğu *u*'dan başlayıp *i*'de biten bağlantı serisinin uzunluğuna eşittir.
- u ve *i* düğümleri arasındaki en kısa yol uzunluğuna jeodezik mesafe adı verilir.
- *u* ve *i* şeklindeki iki düğüm arasındaki ağırlıklı yol uzunluğu, *u* düğümünden başlayıp *i* düğümünde biten bağlantı serisinin ağırlıklarının toplamına eşittir. Yani, yol boyunca geçen kenarların ağırlıkları toplanarak bu yolun ağırlıklı uzunluğu hesaplanır.

MovieLens-100K Dosya İçeriği

u.user: Kullanıcılar hakkında bilgiler içerir. Her bilgi '|' ile ayrılmıştır. Sırasıyla;
" ID numarası | yaş | cinsiyet | meslek | posta kodu " şeklindedir.

u.item: Filmler hakkında bilgiler içerir. Her bilgi '|' ile ayrılmıştır. Sırasıyla;
" ID numarası | film adı | yayınlanma tarihi | video yayınlanma tarihi | IMDB adresi | tür " şeklindedir.

u.data: Kullanıcılar ve filmler arasındaki bağlantılar hakkında bilgiler içerir. Her bilgi boşlukla ayrılmıştır. Sırasıyla;
"kullanıcı ID numarası film ID numarası puan puanın verilme tarihi" şeklindedir.

Kullanılacak Algoritmalar

1. Hiçbir kriteri önceliklemeyen kullanıcının daha önce izlemediği filmler arasından tamamen rassal bir film öneri algoritması.
2. Kullanıcının daha önce izlemediği filmler arasından derecesi en yüksek olan filmleri öneren bir algoritma.
3. Hedef kullanıcının en fazla ortak film izlediği kullanıcı ile benzerlik kurarak, bu kullanıcının en çok beğendiği filmleri öneren bir algoritma.
4. Hedef kullanıcıya en yakın ağırlıklı uzaklığa sahip filmleri öneren bir algoritma.
5. Hedef kullanıcının beğendiği filmlerden yararlanarak beğendiği türü bulup, kullanıcıya bu türde en çok izlenen filmleri öneren bir algoritma.

UYGULAMA DETAYLARI

Kodlama Süreci

Bu projede C programlama dili kullanılmıştır. C'nin "stdio.h, stdlib.h, string.h, time.h" kütüphanelerinden faydalanılmıştır. Kullanıcıların bilgilerini saklayabilmek için "User" adında struct tanımlanmıştır. Bu struct'ın içinde kullanıcının ID numarası, bağlantı sayısı ve bağlantıları saklanmaktadır. Filmlerin bilgilerini saklamak içinse "Movie" adında struct tanımlanmıştır. Bu struct'ın içinde filmin ID numarası, filmin adı, filmin türü, bağlantı sayısı ve bağlantıları saklanmaktadır. Bağlantıların kendisi de "Interaction" adında struct'ta saklanmaktadır. Bu struct'ın içinde ID numarası ve puan yer almaktadır. Hem User hem de Movie struct'ının içinde Interaction türünde pointer vardır ve çalışma esnasında dinamik olarak hafıza tahsis edilecektir.

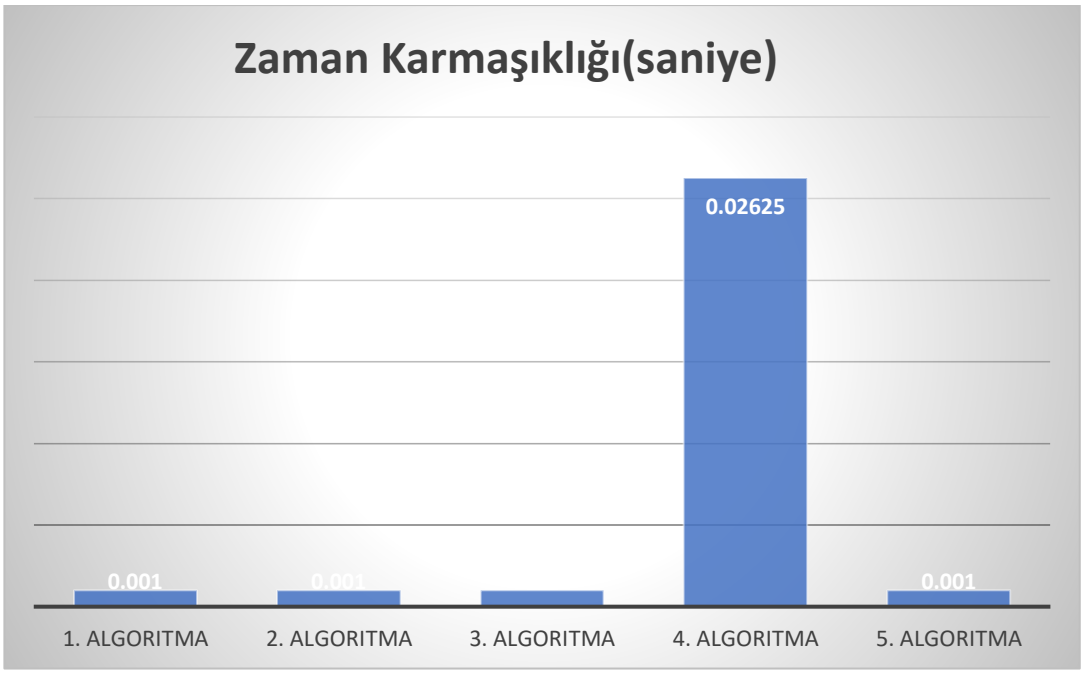
Graph yapısı "dizi + bağlı liste" şeklinde uygulanmıştır. Tüm kullanıcılarının ve filmlerin ID'sine göre kaydedildiği 2 tane dizi vardır. Dizilerin her bir indeksinde, o indekse eşdeğer ID'ye sahip User veya Movie'nin yerini gösteren pointer vardır. Bu User ve Movie'lere de çalışma esnasında dinamik olarak hafıza tahsis edilecektir.

Algoritmaların Zaman ve Mekan Karmaşıklığı

m: Kullanıcının kenar sayısı, n: Filmin kenar sayısı, s: istenilen öneri sayısı, t: toplam film sayısı u: toplam kullanıcı sayısı olmak üzere;

- 1. Algoritma: İlk önce rastgele bir film ID'si üretir. Kullanıcının bu filmi izleyip izlemediğini, kullanıcının izlediği filmleri tarayarak kontrol eder. Eğer kullanıcı filmi izlemediyse kullanıcıya filmi önerir, eğer izlediyse tekrar film ID'si üretir. Yani rekürsif bir şekilde çalışır.
  En iyi durumda, yani tek seferde başarılı bir şekilde film ID'si üretilirse Mekan Karmaşıklığı = Ω (1) olacaktır. En kötü durumda, yani her seferinde izlediği film ID'lerinden biri üretilirse Mekan Karmaşıklığı = O(m) olacaktır.
  Her durumda Zaman Karmaşıklığı = O(m*s) olacaktır.
- 2. Algoritma: Her film teker teker gezilir ve en çok izlenen filmler için kullanıcının o filmi izleyip izlemediğinin kontrolü yapılır. Kullanıcı filmi izlemediyse film kullanıcıya önerilir.
  Her durumda Mekan Karmaşıklığı = O(1) olacaktır. Sadece fonksiyon için bir yer ayrılacaktır.
  Her durumda Zaman Karmaşıklığı = O(m*t) olacaktır.

- 3. Algoritma: Kullanıcının en çok ortak film izlediği kullanıcı bulunur ve onun beğendiği filmler kullanıcıya önerilir.
  Her durumda Mekan Karmaşıklığı = O(u) olacaktır. Çünkü bu algoritmada toplam kullanıcı sayısı boyutunda bir tablo oluşturuluyor ve kullanıcının diğer kullanıcılarla olan ortak film sayısı burada tutuluyor.
  Her durumda Zaman Karmaşıklığı = O(m1*t +m1*m2) olacaktır. (m1; film önereceğimiz kullanıcının kenar sayısı ve m2; bu filmleri izleyen kullanıcıların ortalama kenar sayısıdır.)
- 4. Algoritma: Kullanıcının izlemediği ve kullanıcıya ağırlıklı uzaklığı en yakın olan film veya filmler bulunup kullanıcıya önerilir. Kullanıcının filmlere uzaklığını kaydetmek için toplam film sayısı boyutunda bir tablo oluşturuluyor.
  Her durumda Mekan Karmaşıklığı = O(t) olacaktır.
  Her durumda Zaman Karmaşıklığı = O(m1*m2*n + t) olacaktır. (m1; film önereceğimiz kullanıcının kenar sayısı, n; bu kullanıcının izlediği filmlerin ortalama kenar sayısı ve m2; bu filmleri izleyen kullanıcıların ortalama kenar sayısıdır.)
  Bu algoritma muhtemelen diğerlerine göre daha yüksek bir Zaman Karmaşıklığına sahiptir.
- 5. Algoritma: Kullanıcın beğendiği filmlerin türünü bulup, kullanıcıya o türde izlemediği ama diğer insanlar tarafından çok izlenilen filmleri önerir. Eğer kullanıcının beğendiği film bulunamamışsa rastgele film önerisi yapılır. c toplam tür sayısı olmak üzere;
  Eğer kullanıcıya rastgele film önermek zorunda kalınmazsa Mekan Karmaşıklığı = O(c) ve Zaman Karmaşıklığı = O(c*m + t) olacaktır.
  Eğer kullanıcıya rastgele film önerilmek zorunda kalınırsa da Mekan Karmaşıklığı = O(c + m) ve Zaman Karmaşıklığı = O(c*m + t + m*s) olacaktır.



SONUÇ VE DEĞERLENDİRME

Grafiğe göre 4. algoritma en yavaş çalışandır. Zaten beklenilen de oydu. Tüm hesapları yapıp kesin çözümü sunduğu için uzun sürmesi beklenen bir sonuçtur. Bu yüzden bu projede kullanılan nispeten küçük veriler için uygundur ama daha yüksek verilerde daha farklı algoritmalar kullanılması daha doğru olacaktır. Diğer algoritmaların daha yüksek verilerde de kullanılmasına devam edilebilir.