İlan Öneri Sistemi Raporu

Giriş

Bu proje, iş ilanları için kullanıcıların etkileşim davranışları ve ilanların içerik bilgilerine dayalı hibrit bir öneri sistemi geliştirmeyi amaçlamaktadır. Kullanıcıların tıklama ve satın alma davranışlarını analiz edilerek benzer iş ilanlarını tespit edebilmek ve bu sayede kullanıcılara ilgili olabilecekleri yeni iş ilanları önerebilmek hedeflenmiştir.

Veri Setleri

Proje iki ana veri seti kullanmaktadır:

1. **item_information.csv**: İş ilanlarına ait bilgileri içerir:
    a. item_id: İlan ID'si
    b. pozisyon_adi: İş pozisyonu
    c. item_id_aciklama: İş ilanı açıklaması
2. **user_event_data.csv**: Kullanıcıların iş ilanlarıyla etkileşimlerini içerir:
    a. client_id: Kullanıcı ID'si
    b. item_id: Etkileşimde bulunulan iş ilanı ID'si
    c. event_type: Etkileşim türü ('click' veya 'purchase')
    d. ds_search_id: Kullanıcının site içinde yaptığı arama ID'si
    e. timestamp: Etkileşim zamanı

Mimari Tasarım

Temel Bileşenler

1. Graph Kurulumu ve Analizi

- create_graph(): Kullanıcı-ilan etkileşimlerinden graph oluşturur.
- get_subgraph(): Bellek ve hesaplama kaynaklarını yönetmek için büyük graftan örnek alt graph oluşturulur. Kaynak yetersizliğinden dolayı bu projede yalnızca

10.000 nodeluk subgraph kullanılmıştır. Bunlardan 4337 tanesi ilan, 5663 tanesi kullanıcı node'udur.

- İki ayrı node vardır. Kullanıcı node'u U_ ön eki, iş ilanı node'u J_ ön eki ile isimlendirilir.(user and job)
- Kenarlar arasındaki ağırlıklar etkileşim türüne göre belirlenir: tıklama (click) = 1, satın alma (purchase) = 3. Aynı arama oturumunda görüntülenen ilanlar arasında 0.5 ağırlıklı kenarlar oluşturulur. (Önem arttıkça weight değeri artar)
- Kullanıcılar, etkileşim türlerine göre birçok iş ilanına bağlı olacaktır. Yüksek etkileşimli kullanıcılar, daha fazla iş ilanına bağlı düğümlere sahip olacaktır. İş ilanları, onları inceleyen veya tıklayan kullanıcılarla bağlanacaktır.
- Arama bağlantıları (yani, aynı aramada görünen işler) eklenmiş olduğundan, iş ilanları arasında birden fazla kenar bağlantısı da oluşacaktır. Bu, kullanıcıların bu işleri birlikte incelediğini gösterir.

2. Kullanıcı Verisi ile İlan Vektörü Üretme

- Node embedding yöntemi olarak Node2Vec algoritması kullanılmıştır. Bu sayede graph yapısındaki ilanlar sayısal vektörlere dönüştürür. İş ilanı nodeları için 64 boyutlu vektörler elde edilir.
- Node2Vec, rastgele yürüyüşler kullanarak her bir düğüm için vektörler üretir. Bu, hem yerel komşulukları hem de daha uzak komşulukları yakalayabilmesine olanak tanır. Büyük verilerde güçlü çalışır.
- get_job_embeddings(): Kullanıcı verileriyle ilan vektörü üretir.
- calculate_average_similarity(): İlanlar arasındaki kosinüs benzerliklerini hesaplar.
- Node embedding'in kalitesini optimize etmek için metrik olarak **Cosine Similarity** kullanılmıştır.
- **Cosine similarity** , iki vektör arasındaki benzerliği ölçen bir yöntemdir. İki vektör birbirine yaklaştıkça değer 1'e yaklaşır. Uzaklaştıkça 0'a yaklaşır

3. Metin İşleme ve İlan Bilgileri ile İlan Vektörü Üretme

- preprocess_text(): Ham metin verilerinden anlamlı bilgiler çıkarmak için çeşitli temizleme işlemleri yapar. Küçük harfe dönüştürme, özel karakterlerin ve sayıları temizlenmesi ve tokenization uygulanmıştır.
- create_embeddings_subgraph(): SentenceTransformer modeli kullanarak metin verilerinden embedding vectorleri oluşturur.

- SentenceTransformer modeli "paraphrase-multilingual-MiniLM-L12-v2" kullanılarak Türkçe metin verilerinden 384 boyutlu embedding vektörleri oluşturur.
- Paraphrase-MiniLM modeli, cümlelerin anlamını düşük boyutlu vektörlerle temsil eder ve özellikle eş anlamlı cümleler üzerine eğitilmiştir.
- Her bir iş ilanının başlığı ve açıklamasını alınarak, temizlenir ve ardından bu metin bir embedding vektörüne dönüştürülür.

4. Öneri Sistemi

- combine_embeddings(): İki ayrı embeddingi birleştirerek hibrit vektörler oluşturur.(448 boyutlu vektör elde edilir)
- create_faiss_index(): Hızlı benzerlik aramaları için FAISS indeksi oluşturur.
- get_top_n_similar_jobs(): Belirli bir iş ilanına en benzer ilanları döndürür.
- **Node2Vec** kullanılarak oluşturulan vektörler ve **SentenceTransformer** kullanılarak oluşturulan Job Embedding (iş açıklaması ve başlıklarından türetilmiş embedding) vektörleri birleştirilerek daha kapsamlı bir temsil oluşturulur.
- Concat edilmiş vektörler ile benzerlik hesaplama işlemi, iş ilanlarını birbirine yakınlıklarına göre sıralamak için yapılır. FAISS (Facebook AI Similarity Search) kütüphanesi, büyük veri kümelerinde yüksek hızlı benzerlik arama işlemleri yapmaya olanak tanır.
- Veritabanı olarak FAISS kullanılmıştır.
- Concat edilmiş vektörler üzerinde FAISS kütüphanesini kullanarak, L2 mesafe (Euclidean distance) üzerinden vektörleri karşılaştırılır. Bu, sayede iş ilanlarının benzerliklerini hızlı bir şekilde bulunur.
- Similarity Score değerinin küçük olması daha ilgili ilanlar bulunduğu anlamında gelir.