**YALOVA ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**PROJE ADI:**

**FİLM ÖNERİ SİSTEMİ**

**200101049 – YUNUS EMRE ERKELEŞ**

**200101016 – RECEP BAYKARA**

**200101045 – MUHAMMED EMİN SAYAN**

**200101108 – YALÇIN ÖZDEMİR**

1. **Film Verisi Kazıma Kodları Hakkında Doküman**

**Bu doküman, verilen iki Python dosyasının (“combined\_scrape.py” ve “httpx\_async\_demo.py”) işleyişini açıklamaktadır. Her iki dosya, IMDb gibi bir kaynaktan film verilerini asenkron bir şekilde kazıyarak toplamaya odaklanmıştır. Kodların işlevi ve kullanılan yaklaşımlar aşağıda detaylandırılmıştır.**

**1. Giriş: Kodların Amacı ve Kullanılan Teknolojiler**

**Bu iki dosya, film bilgilerini etkili bir şekilde toplamak ve saklamak için geliştirilmiştir. Şu teknolojiler kullanılmıştır:**

* **httpx: Asenkron HTTP istemcisi, web isteklerini hızlı bir şekilde gerçekleştirir.**
* **asyncio: Python'da asenkron programlama için temel kütüphane.**
* **Playwright: Tarayıcı otomasyonu için kullanılır, dinamik çerezler veya JavaScript render edilen sayfalardan veri toplamada etkilidir.**
* **JSON: Film verilerini işlemek ve saklamak için kullanılan format.**

**2. combined\_scrape.py Dosyası**

**Bu dosya, IMDb’den film linklerini toplar ve bu linklerin daha sonra detaylı bilgi kazımında kullanılması için bir temel oluşturur. Kodun genel işleyişi aşağıda detaylandırılmıştır:**

**Önemli Fonksiyonlar**

1. **write\_to\_file:**
   * **Film isimlerini ve URL’lerini movie\_info.txt dosyasına kaydeder.**
   * **Dosya yazımı esnasında her bir film için satır bazında formatlama sağlar.**
2. **continue\_movie\_info:**
   * **Daha önce kaydedilmiş bir URL listesi varsa, kaldığı yerden devam etmek için son URL’yi ve satır sayısını dönüştürür.**
   * **Eksik veya hatalı bir dosya durumunda başlangıç yapılabilmesini sağlar.**
3. **main:**
   * **Playwright kullanarak IMDb’nin dinamik olarak oluşturulan sayfalarına erişir.**
   * **Filmleri sıralamak, linklerini almak ve toplamda 50’şerli gruplar halinde sayfaları kazır.**
   * **Her yeni veri toplama adımından sonra httpx\_async\_demo.py dosyasıyla birlikte detaylı bilgi kazımını başlatır.**

**Genel Akış**

1. **Playwright ile IMDb’ye gidilir ve filmler listelenir.**
2. **Her film için isim ve URL bilgisi toplanarak dosyaya kaydedilir.**
3. **Kaydedilen linkler, detaylı kazıma yapmak üzere aktarılır.**

**Teknik Detaylar**

* **Dinamik Yüklemeler: IMDb sayfaları JavaScript ile dinamik olarak oluşturulduğu için Playwright kullanılmıştır.**
* **Asenkron Kontrol: Performansı maksimize etmek için, gereksiz resim isteklerini engelleyen özelleştirilmiş route ayarı kullanılmıştır.**

**3. httpx\_async\_demo.py Dosyası**

**Bu dosya, combined\_scrape.py tarafından toplanan URL listelerini kullanarak film detaylarını asenkron bir şekilde kazır. IMDb’nin sunduğu JSON-LD formatındaki verileri işler.**

**Önemli Fonksiyonlar**

1. **read\_movie\_info:**
   * **movie\_info.txt dosyasından URL listesini okur.**
   * **Daha önceki kazıma sürecine devam edebilmek için satır bazında başlangıç noktası seçilebilir.**
2. **write\_detail\_movie\_info:**
   * **Detay kazıma sonrasında elde edilen veriler, detail\_movie\_info\_async.txt dosyasına kaydedilir.**
   * **Veriler, analiz kolaylığı için belirli bir formatta saklanır.**
3. **fetch\_movie\_data:**
   * **IMDb’den JSON-LD formatındaki bilgileri çeker.**
   * **Çekilen veri; ad, poster URL, yayın tarihi gibi alanlara ayrılır ve JSON hatalarına karşı kontrol sağlar.**
   * **IMDb’nin IP banı uygulamasını engellemek için asenkron istekler sınırlandırılır (Semaphore).**
4. **scrape\_movies:**
   * **URL listesini işleyerek her biri için fetch\_movie\_data fonksiyonunu çalıştırır.**
   * **Tüm işlemler asenkron şekilde gerçekleştirilir ve performans optimize edilir.**

**Genel Akış**

1. **combined\_scrape.py tarafından sağlanan URL’ler okunur.**
2. **IMDb’ye HTTP istekleri gönderilerek JSON verisi çekilir.**
3. **Film detayları işlenerek dosyaya yazılır.**

**Teknik Detaylar**

* **Semaphore Kullanımı: IMDb’nin IP banından kaçınmak için bir seferde maksimum 9 istek gönderilmesi sağlanır.**
* **Hata Kontrolü: HTTP bağlantı hataları ve JSON decode hataları özenle yakalanır ve loglanır.**

**4. Asenkron Programlama Kullanımı**

**Bu kodlarda, asenkron programlama şu şekilde optimize edilmiştir:**

* **Performans Artışı:**
  + **Birden fazla isteğin eşzamanlı olarak gönderilmesi bekleme sürelerini azaltır.**
  + **Semaphore, IMDb’den ban yeme riskini azaltmak ve aynı anda gönderilen istek sayısını sınırlamak için kullanılır. Bu yöntem, sunucunun aşırı yüklenmesini önler ve IP engelleme ihtimalini düşürür.**
* **Tarayıcı Kontrolü:**
  + **Playwright, dinamik çerezler ve JavaScript’le oluşturulan öğeleri kazıma için etkilidir. Dinamik olarak oluşan HTML içeriğini algılamak, JavaScript tabanlı öğeleri işlemek ve kullanıcı hareketlerini taklit ederek veri toplamak için özellikle kullanışlıdır. Bu sayede, geleneksel HTTP istekleriyle erişilemeyen bilgilere ulaşılabilir.**

1. **Veri Temizleme(clean\_csv.py)**

**Bu kod, bir Excel dosyasındaki verileri okur, verileri temizler ve sonrasında temizlenmiş verileri bir CSV dosyasına kaydeder. Kod, verileri parçalara ayırarak işler, boşlukları temizler, tekrarlanan verileri kaldırır ve ardından verileri birleştirip son halini CSV formatında kaydeder. Şimdi, bu işlemleri adım adım açıklayalım:**

**1. Gerekli Kütüphaneler**

**İlk olarak, veri işleme için gerekli olan pandas ve re kütüphaneleri içe aktarılır. pandas, verileri düzenlemek ve işlemek için kullanılan bir Python kütüphanesidir. re kütüphanesi ise düzenli ifadelerle metin verilerini temizlemek için kullanılır.**

**2. Dosya Yolları**

**Kodun başında, verilerin bulunduğu Excel dosyasının yolu (xlsx\_file) ve verilerin kaydedileceği CSV dosyasının yolu (csv\_file) belirtilir. Bu dosya yolları, veri dosyasının bulunduğu dizine göre ayarlanmalıdır.**

**3. Veri Temizliği İçin Yardımcı Fonksiyon**

**Kodda, veri temizliği işlemi için clean\_whitespace adında bir yardımcı fonksiyon tanımlanır. Bu fonksiyon, metin değerlerinin başındaki ve sonundaki gereksiz boşlukları kaldırır ve metin içerisindeki birden fazla boşluğu tek bir boşlukla değiştirir. Böylece veri daha düzenli hale gelir.**

**4. Excel Dosyasının Okunması**

**Excel dosyasını açmak için pd.ExcelFile fonksiyonu kullanılır. Bu fonksiyon, dosyadaki tüm sayfa isimlerini (sheet\_names) alır ve her sayfayı sırasıyla işler. Kodda her sayfa için ayrı ayrı işlemler yapılacaktır.**

**5. Veri Temizleme Adımları**

* **Eksik Verilerin Silinmesi: Her sayfada bulunan eksik (NaN) değerler (dropna() fonksiyonu ile) silinir.**
* **Boşlukların Temizlenmesi: Sayfadaki metin kolonlarındaki boşluklar temizlenir. Bu işlem, her metin verisi üzerinde clean\_whitespace fonksiyonu uygulanarak yapılır.**
* **"Name" Kolonunun Düzenlenmesi: Eğer sayfada "Name" adında bir sütun varsa, bu sütundaki tüm verilerin başındaki ve sonundaki boşluklar temizlenir.**
* **Tekrarlanan Verilerin Kaldırılması: "Name", "Description", ve "Genre(s)" sütunları baz alınarak tekrarlanan satırlar silinir. Ayrıca sadece "Name" sütununa göre de tekrarlanan satırlar silinir.**

**6. Verinin Parçalara Ayrılması**

**Veri çok büyükse, tüm veriyi bir seferde işlemek bellek sorunlarına yol açabilir. Bu nedenle, veriler parçalara (chunk) ayrılır. Belirtilen chunk\_size kadar satır işlenir, bu sayede daha küçük veri parçaları halinde işlem yapılır. Bu işlem, verinin tamamı işlenene kadar devam eder.**

**7. Verilerin Birleştirilmesi**

**İşlem sonunda, her bir parçada işlenen veriler bir listeye (all\_data) eklenir. Verinin tüm parçaları işlendiğinde, bu parçalar birleştirilerek tek bir DataFrame haline getirilir.**

**8. CSV Dosyasına Kaydetme**

**Son olarak, işlenmiş veri birleştirilip CSV dosyasına kaydedilir. to\_csv() fonksiyonu kullanılarak, veriler belirtilen dosya yoluna (csv\_file) kaydedilir. Ayrıca, CSV dosyasına başlık satırı eklenmez ve veriler UTF-8 formatında kaydedilir.**

**9. İşlemin Tamamlanması**

**İşlem tamamlandığında, ekrana bir başarı mesajı yazdırılır. Bu mesajda, verilerin başarıyla işlenip kaydedildiği belirtilir.**

**Bu adımlar, verilerin düzenlenmesi ve CSV formatına dönüştürülmesi sürecini tamamlar. Bu işlemle, ham veriler temizlenmiş, gereksiz boşluklar kaldırılmış ve tekrarlanan veriler silinmiş olur. Sonuç olarak, daha düzgün ve kullanıma uygun bir CSV dosyası elde edilir.**

1. **Kodun Genel Amacı**

**Bu kod, bir film öneri sistemi geliştirmek için bir Python uygulamasıdır. Kullanıcı, bir film ismi girer ve sistem, seçilen filme benzer diğer filmleri önerir. Öneriler, filmlerin türleri, anahtar kelimeleri, açıklamaları ve IMDB puanlarına dayalı olarak yapılır. Kodda, Autoencoder tabanlı bir yapay sinir ağı (ANN) modeli kullanılarak filmler arasındaki benzerlikler öğrenilir ve bu benzerliklere dayalı öneriler sunulur.**

**Kodun İşleyişi ve Yapısı**

**1. Kullanılan Kütüphaneler**

**Veri Analizi ve İşleme**

* **pandas: Veri çerçeveleri üzerinde işlem yapmak için.**
* **numpy: Sayısal işlemler ve matris/varyans hesaplamaları için.**

**Makine Öğrenimi**

* **scikit-learn:**
  + **MultiLabelBinarizer: Çift etiketli verileri ikili formatta temsil eder.**
  + **TfidfVectorizer: Metin verilerini sayısal özelliklere dönüştürür.**
  + **StandardScaler: Verilerin normalize edilmesi için kullanılır.**
  + **cosine\_similarity: Vektörler arası benzerlik ölçümü yapar.**

**Derin Öğrenme**

* **tensorflow.keras: Model oluşturma ve eğitim işlemleri için.**
  + **Sequential: Sıralı model yapısı sağlar.**
  + **Dense: Tam bağlantılı katmanlar oluşturur.**
  + **Dropout: Aşırı öğrenmeyi engeller.**
  + **BatchNormalization: Katmanlar arası normalizasyon sağlar.**
  + **EarlyStopping: Eğitim sırasında erken durdurma uygular.**

**2. Veri İşleme ve Model Eğitimi**

**2.1. Veri Temizleme ve Özellik Çıkartma**

* **Veri Temizleme (clean\_text):**
  + **Film açıklamaları küçük harfe dönüştürülür.**
  + **Özel karakterler ve fazla boşluklar kaldırılır.**
* **Etiketlerin Binarize Edilmesi:**
  + **MultiLabelBinarizer kullanılarak film türleri ve anahtar kelimeler sayısal formata dönüştürülür.**
* **TF-IDF Özellik Çıkartımı (TfidfVectorizer):**
  + **Film açıklamaları üzerinden kelime sıklığı (TF) ve ters belge sıklığı (IDF) hesaplanır.**
* **Rating Özellikleri:**
  + **Eksik IMDB puanları ortalama değerle doldurulur.**
  + **Veriler, StandardScaler ile normalize edilir.**

**2.2. Özelliklerin Birleştirilmesi**

**Farklı özellikler birleştirilerek tek bir özellik matrisi oluşturulur. Ağırlıklar:**

* **Genres: 1.0**
* **Keywords: 1.5**
* **Description: 2.0**
* **Rating: 1.0**

**3. Modelin Mimari Yapısı**

**3.1. Encoder (Kodlayıcı)**

* **Katman 1: 512 nöronlu tam bağlantılı katman, ReLU aktivasyonu.**
* **Katman 2: 256 nöronlu tam bağlantılı katman, ReLU aktivasyonu.**
* **Batch Normalization ve Dropout: Her katmanın ardından eklenir.**

**3.2. Bottleneck (Sıkıştırma Katmanı)**

* **128 nöronlu bir katman, sıkıştırma işlemini yapar.**

**3.3. Decoder (Çözümleyici)**

* **Katman 1: 256 nöronlu tam bağlantılı katman.**
* **Katman 2: 512 nöronlu tam bağlantılı katman.**
* **Çıktı Katmanı: Orijinal boyutta veri döndürür, Sigmoid aktivasyonu kullanır.**

**4. Model Eğitimi**

* **Epoch Sayısı: 2 epoch boyunca eğitim yapılır.**
* **Batch Size: 64 örnek/batch.**
* **Validation Split: %20 doğrulama verisi ayrılır.**
* **EarlyStopping: 5 epoch boyunca iyileşme olmazsa eğitim durdurulur.**

**5. Öneri Yapma**

1. **Film Seçimi: Kullanıcı, bir film adı girer.**
2. **Özelliklerin Kodlanması: Model, tüm filmlerin özelliklerini kodlar.**
3. **Kosinüs Benzerliği: Filmler arasındaki benzerlik hesaplanır.**
4. **Türler Arası Benzerlik: Benzer türdeki filmler ekstra puan alır.**
5. **Benzerlik Eşiği: 0.3'ün üzerinde olan filmler seçilir.**
6. **Öneri Listesi: Filmler benzerlik skorlarına göre sıralanır.**

**6. Kodun Genel Yapısı**

1. **Veri Okuma ve Hazırlama: Veri kümesi okunur ve temizlenir.**
2. **Özelliklerin Hazırlanması: TF-IDF ve normalize işlemleri yapılır.**
3. **Modelin Eğitilmesi: Autoencoder modeli eğitilir.**
4. **Öneri Alımı: Kullanıcı girişine göre öneriler sunulur.**