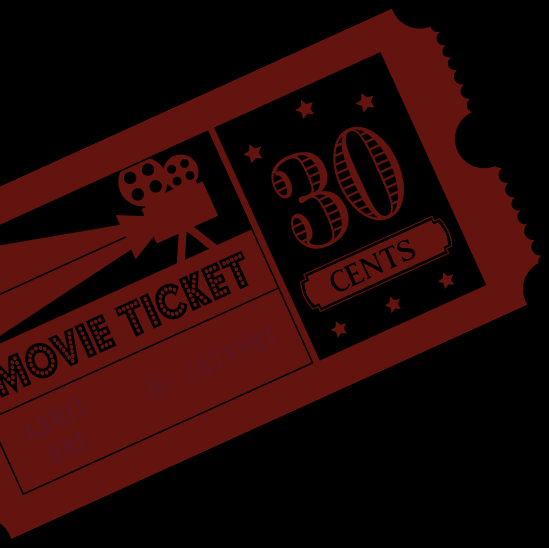
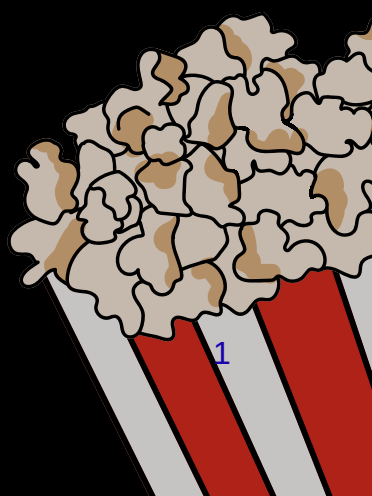
Cosine Similarity
Kullanılarak Geliştirilen

Film Öneri Sistemi



Zeynep SARI
22120205003

Derin Öğrenme



1. Özet

Bu projede, The Movie Database (TMDB) veri seti kullanılarak tamamen kullanıcı tarafından eğitilen, içerik tabanlı bir film öneri sistemi geliştirilmiştir. Film özetleri, türler, anahtar kelimeler, oyuncu ve yönetmen bilgileri birleştirilerek metinsel veri elde edilmiştir. Bu metinler, öncelikle sayısal vektörlere dönüştürülmüş, ardından çok katmanlı bir derin öğrenme modeli (AutoEncoder) kullanılarak daha düşük boyutlu ve anlamlı temsiller öğrenilmiştir.

Eğitilen modelden elde edilen film vektörleri arasındaki benzerlikler cosine similarity metriği ile hesaplanmış ve kullanıcıya içerik açısından en benzer filmler önerilmiştir. Ayrıca sistem; büyük/küçük harf duyarsız arama, yazım hatalarına tolerans ve Streamlit tabanlı web arayüzü ile kullanıcı dostu hale getirilmiştir.

2. Gelişme

2.1 Preprocess.py (Veri Seti ve Ön İşleme Süreci)

Projede Kaggle üzerinden temin edilen TMDB 5000 Movies ve TMDB 5000 Credits veri setleri kullanılmıştır. Bu veri setleri sırasıyla film bilgilerini ve oyuncu/yapım ekibi bilgilerini içermektedir.
preprocess.py dosyasında gerçekleştirilen işlemler:

- Film ve kredi verileri id ve movie_id alanları üzerinden birleştirilmiştir.
- Türler, anahtar kelimeler ve oyuncu bilgileri JSON formatından ayrıştırılarak metin haline getirilmiştir.
- Yapım ekibi verisi içerisinden yalnızca yönetmen (Director) bilgisi seçilmiştir.
- Film özeti, türler, anahtar kelimeler, oyuncular ve yönetmen bilgileri tek bir metin alanında birleştirilmiştir.
- Metinler küçük harfe dönüştürülmüş, noktalama işaretleri ve gereksiz karakterler temizlenmiştir.

Bu adımın amacı, derin öğrenme modeline verilecek metinsel girdinin tutarlı, temiz ve anlamlı hale getirilmesidir.

2.2 Feature Extraction (Metnin Sayısal Temsili)

Ön işleme sonrası elde edilen metinler doğrudan derin öğrenme modeline verilebilmesi için TF-IDF yöntemi ile sayısal vektörlere dönüştürülmüştür. Bu yöntem, kelimelerin doküman içindeki ve tüm veri setindeki önemini dikkate alarak her filmi yüksek boyutlu bir vektör ile temsil etmektedir.
Bu aşamada her film, derin öğrenme modeline girdi olacak şekilde standart bir boyutta temsil edilmiştir.

2.3 Train.py (Modelin Eğitimi)

train.py dosyasında AutoEncoder modelinin eğitim süreci gerçekleştirilmiştir.
Bu dosyada:

- Model PyTorch kullanılarak tanımlanmıştır.

- Kayıp fonksiyonu olarak Mean Squared Error (MSE) kullanılmıştır.
- Optimizasyon işlemi Adam optimizer ile yapılmıştır.
- Model, belirlenen epoch sayısı boyunca tüm veri üzerinde eğitilmiştir.
- Eğitim sürecinde kayıp değerleri izlenerek modelin öğrenme başarımı kontrol edilmiştir.
- Bu aşama, filmler arasındaki içerik ilişkilerinin model tarafından öğrenildiği temel aşamadır.

2.4 Recommender.py (Öneri Mekanizması)

recommender.py dosyasında öneri sistemi mantığı uygulanmıştır.
Gerçekleştirilen işlemler:

- Kullanıcıdan alınan film adı büyük/küçük harf duyarsız şekilde işlenmiştir.
- Yazım hatalarına tolerans sağlamak amacıyla en yakın film adı difflib kütüphanesi ile belirlenmiştir.
- Seçilen filmin AutoEncoder tarafından üretilen latent vektörü ile tüm filmlerin vektörleri arasında cosine similarity hesaplanmıştır.
- En yüksek benzerlik skoruna sahip filmler öneri olarak sunulmuştur.

Bu sayede öneriler, tamamen modelin öğrendiği içerik temsillerine dayalı olarak oluşturulmuştur.

2.5 Main.py (Ana Program Akışı)

main.py dosyası, projenin terminal tabanlı çalışmasını sağlamaktadır.
Bu dosyada:

- Veri seti yüklenmekte ve ön işleme uygulanmaktadır.
- TF-IDF vektörleri oluşturulmaktadır.
- AutoEncoder modeli eğitilmektedir.
- Kullanıcıdan film adı alınmakta ve öneriler hesaplanarak ekrana yazdırılmaktadır.
- Hatalı girişlerde kullanıcıya en yakın film adı önerilmektedir.

Bu yapı, projenin modüler, okunabilir ve geliştirilebilir olmasını sağlamaktadır.

2.6 App.py (WEB Arayüzü)

Projeye kullanıcı deneyimini artırmak amacıyla Streamlit kullanılarak web tabanlı bir arayüz geliştirilmiştir.
app.py dosyası:

- Kullanıcıdan film adı alır
- Eğitilmiş model üzerinden önerileri hesaplar
- Sonuçları web ortamında kullanıcıya sunar

Bu sayede sistem, teknik bilgi gerektirmeden herkes tarafından kullanılabilir hale getirilmiştir.

3.Model Seçimi ve Gerekçesi

Bu projede model olarak AutoEncoder tabanlı çok katmanlı bir sinir ağı tercih edilmiştir.
Modelin Seçilme Nedenleri:

- AutoEncoder'lar etiket gerektirmeyen (unsupervised) yapıları sayesinde öneri sistemleri için uygundur.
- Metinsel verilerin yüksek boyutlu temsillerini daha düşük boyutlu, anlamlı latent vektörlere dönüştürebilmektedir.
- Model tamamen kullanıcı tarafından eğitildiği için hazır gömme (embedding) modellerine bağımlılık ortadan kalkmıştır.
- Öğrenilen temsiller, film içerikleri arasındaki benzerlikleri daha iyi yansıtabilmektedir.

Model Yapısı:

- Giriş katmanı: TF-IDF vektörleri
- Gizli katmanlar: Çok katmanlı, doğrusal olmayan dönüşümler
- Daraltılmış latent katman: Filmin öğrenilmiş temsili
- Çıkış katmanı: Girişin yeniden oluşturulması
- Model, giriş verisini yeniden üretmeye çalışarak en iyi ara temsili öğrenmektedir.

3.1 Model Değerlendirme Yaklaşımı

Modelin başarımı, filmler arasındaki içerik benzerliğinin ne kadar tutarlı temsil edildiği üzerinden değerlendirilmiştir. AutoEncoder tarafından öğrenilen latent vektörler arasındaki benzerlik, cosine similarity metriği ile ölçülmüştür.
Yüksek benzerlik skoru, filmlerin içerik açısından daha yakın olduğunu göstermektedir. Bu yaklaşım, kullanıcı geçmişi bulunmayan durumlar için etkili bir çözüm sunmaktadır.

4.Sonuç

Bu çalışmada, TMDB veri seti kullanılarak tamamen kullanıcı tarafından eğitilen, içerik tabanlı bir film öneri sistemi başarıyla geliştirilmiştir. Film özetleri, türler, anahtar kelimeler, oyuncu ve yönetmen bilgileri birleştirilerek oluşturulan metinsel veriler; ön işleme adımlarından geçirilmiş ve TF-IDF yöntemi ile sayısal vektörlere dönüştürülmüştür. Elde edilen bu yüksek boyutlu vektörler, çok katmanlı AutoEncoder tabanlı bir derin öğrenme modeli ile eğitilerek daha anlamlı ve kompakt temsil alanlarına indirgenmiştir.
Eğitilen model sayesinde filmler arasındaki içerik benzerlikleri, hazır gömme (embedding) modelleri kullanılmadan, doğrudan veri üzerinden öğrenilmiştir. AutoEncoder tarafından elde edilen latent temsiller kullanılarak cosine similarity metriği ile benzerlik hesaplamaları yapılmış ve kullanıcıya içerik açısından en yakın filmler önerilmiştir. Bu yaklaşım, özellikle kullanıcı geçmişi bulunmayan senaryolarda içerik tabanlı öneri sistemlerinin etkinliğini açıkça ortaya koymaktadır.

Sistem, büyük/küçük harf duyarsız arama ve yazım hatalarına tolerans gibi özelliklerle kullanıcı girişlerindeki belirsizlikleri başarılı şekilde yönetebilmekte; Streamlit tabanlı web arayüzü sayesinde teknik bilgi gerektirmeden kullanılabilmektedir. Modüler yazılım mimarisi sayesinde veri ön işleme, model eğitimi, öneri mekanizması ve kullanıcı arayüzü bileşenleri birbirinden bağımsız ve geliştirilebilir şekilde tasarlanmıştır.
Sonuç olarak, bu proje; derin öğrenme temelli, denetimsiz bir modelin içerik tabanlı öneri sistemlerinde etkin bir şekilde kullanılabileceğini göstermektedir. Gelecek çalışmalarda modelin farklı ağ mimarileri ile geliştirilmesi, kullanıcı etkileşimlerinin sisteme dahil edilmesi ve içerik tabanlı yaklaşımın işbirlikçi filtreleme yöntemleri ile birleştirilerek hibrit öneri sistemlerine dönüştürülmesi planlanmaktadır.

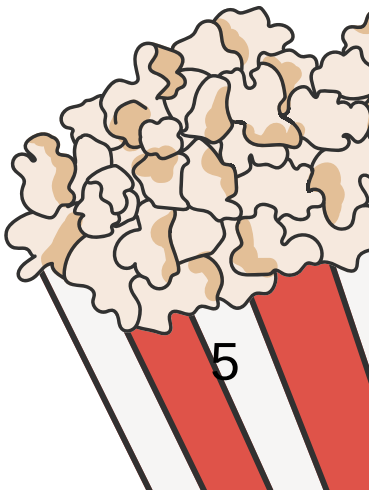
5. Modele Ait Görüntüler

Görsel 1 : The dark knight için öneriler

Görsel 2 : Avengers için öneriler veritabanında eşleşmesi “The Avengers”

Görsel 3 : spectr için öneriler veritabanında eşleşmesi “Sprectre”

Eğer film adını kullanıcının yazdığı gibi bulamazsa veritabanındaki en yakın film adı için önerileri getiriyor.



Kaynakça

[1] Kaggle, TMDB 5000 Movie Dataset.
Erişim adresi: https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata
[2] Ricci, F., Rokach, L., & Shapira, B. (2011).
Introduction to Recommender Systems Handbook.
Springer, Boston, MA.
[3] Aggarwal, C. C. (2016).
Recommender Systems: The Textbook.
Springer International Publishing.
[4] Salakhutdinov, R., Mnih, A., & Hinton, G. (2007).
Restricted Boltzmann Machines for Collaborative Filtering.
Proceedings of the 24th International Conference on Machine Learning (ICML).
[5] Hinton, G. E., & Salakhutdinov, R. R. (2006).
Reducing the Dimensionality of Data with Neural Networks.
Science, 313(5786), 504–507.
[6] Pedregosa, F., et al. (2011).
Scikit-learn: Machine Learning in Python.
Journal of Machine Learning Research, 12, 2825–2830.
[7] Ramos, J. (2003).
Using TF-IDF to Determine Word Relevance in Document Queries.
Proceedings of the First Instructional Conference on Machine Learning.
[8] Manning, C. D., Raghavan, P., & Schütze, H. (2008).
Introduction to Information Retrieval.
Cambridge University Press.
[9] Chollet, F. (2018).
Deep Learning with Python.
Manning Publications.
[10] Streamlit Inc.
Streamlit: A Fast Way to Build Data Apps.
https://streamlit.io