

Netflix Recommendation System — Project Report

Overview

This project focuses on building a **content-based movie recommender system** using the Netflix dataset. The system suggests movies or TV shows based on the textual similarity of their descriptions using **TF-IDF vectorization** and **cosine similarity**. The user interface is built using **Streamlit**, making the system interactive and user-friendly.

Dataset Description

- **Source:** `netflix_titles.csv` [from [Kaggle Netflix Movies and TV Shows dataset](#)]
- **Features:**
 - **title:** Name of the movie/show
 - **type:** Movie or TV Show
 - **description:** Textual synopsis
 - Other metadata: **director**, **cast**, **country**, **release_year**, **rating**,
 - etc.

Data Analysis & Preprocessing (as per `Netflix_analysis.ipynb`)

Key steps performed:

1. **Data Cleaning:**
 - Handled null values, especially in **description**, which was crucial for the model.
 - Dropped irrelevant columns for recommendation logic.
 - Standardized text by converting to lowercase.
2. **Exploratory Data Analysis (EDA):**
 - Count of content types: More movies than TV shows.
 - Country-wise content production: United States dominates the catalog.
 - Most common genres and keywords were explored using word clouds or bar plots.
3. **Insightful Observations:**
 - Netflix has more content released post-2010, reflecting its shift to original productions.
 - **Drama**, **Comedy**, and **Documentary** are the most recurring genres.
 - Certain countries like India and UK also have a strong presence.

Recommendation Engine (from `app.py`)

Methodology

1. **TF-IDF Vectorization:**
 - Description field is transformed using `TfidfVectorizer(stop_words='english')`.
 - This creates a matrix that numerically represents the importance of words relative to each description.
2. **Cosine Similarity:**
 - Calculates similarity scores between each pair of descriptions.
 - For a given title, the most similar titles (excluding itself) are recommended.
3. **Index Mapping:**
 - Titles are mapped to indices using a Pandas Series for fast lookups.

Streamlit Interface

- Users can enter a movie/show title.
- The app suggests top 5 similar titles.

- Feedback for invalid or unknown entries is handled gracefully.

Example Usage

Input: "Stranger Things"**Output** [Sample Recommendations]:

1. The OA
2. Dark
3. The Society
4. Black Mirror
5. The Rain

These results are thematically close, showcasing the model's understanding of sci-fi, thriller, and supernatural elements.

Project Strengths

- **Lightweight & Fast:** Purely content-based, does not require user data.
- **Interpretable:** Recommendations are based on description similarity.
- **Modular:** Easy to plug into larger systems or enhance with metadata.

Limitations & Future Improvements

Limitation Suggestion Doesn't consider user preferences Integrate collaborative filtering
Depends solely on description text Use additional metadata [genre, cast, director] No semantic understanding Incorporate embeddings [e.g., BERT, Sentence Transformers]
Case-sensitive input handling Improve normalization and fuzzy matching

Project Structure

```
└─ Netflix Recommender /  
  └─ app.py           # Streamlit App  
  └─ netflix_titles.csv # Dataset  
  └─ Netflix_analysis.ipynb # EDA and analysis
```

Learnings & Skills Demonstrated

- Applied **TF-IDF and cosine similarity** for NLP-based recommendations.
- Used **Streamlit** to deploy a clean and responsive user interface.
- Performed thorough **EDA** using Pandas and Matplotlib/Seaborn.
- Learned **text preprocessing, vector space modeling, and recommendation systems.**

Conclusion

This project successfully demonstrates a **content-based filtering recommender system** using real-world Netflix data. It effectively highlights how textual descriptions can be leveraged to find similar content. Future versions can adopt hybrid methods for even more robust recommendations.

If you'd like, I can export this report as a downloadable PDF or Word doc with formatted sections and visuals from the notebook. Let me know!