Project Report: Content-Based Movie Recommendation System

Course: Data Mining

Team Members:

- **Mohamed Morsy** – ID: 8199
- **Youssef Awad** – ID: 8179
- **Ahmed Samir** – ID: 8120
- **Abdelrahman Shaaban** – ID: 8041

---

1. Executive Summary

This project focuses on the development of a Content-Based Recommendation System designed to suggest movies to users based on textual similarities in plot descriptions, genres, and metadata. Utilizing a dataset of over 25,000 IMDb entries, the system employs Natural Language Processing (NLP) techniques to transform unstructured text into numerical vectors. Two distinct approaches were implemented and compared: a traditional machine learning pipeline using TF-IDF and Principal Component Analysis (PCA), and a deep learning approach using Sentence-BERT embeddings. The final model groups movies into semantic clusters using Gaussian Mixture Models (GMM) to provide accurate and genre-specific recommendations.

2. Introduction

In the era of information overload, recommendation systems are critical for personalized user experiences. The objective of this project is to build a system capable of understanding the semantic content of a movie—its plot, director, casting, and keywords—to recommend similar items. Unlike collaborative filtering, which relies on user interaction history, this content-based approach relies solely on the attributes of the items themselves, effectively solving the "cold start" problem for new items.

3. Data Acquisition and Preprocessing

The system processes the 25k_IMDb_movie_Dataset.csv. Extensive data cleaning and feature engineering were performed to ensure data quality.

3.1 Data Cleaning

- **Handling Null Values:** Rows with missing critical attributes (Title, Overview, Top 5 Casts) were removed to maintain data integrity.
- **Duplicate Removal:** Duplicate entries were identified based on movie title and release year and subsequently dropped.
- **Noise Reduction:** Movies with placeholder descriptions (e.g., "none") were filtered out. The final cleaned dataset comprises approximately 23,500 unique movies.

3.2 Feature Engineering (The "Movie Profile")

A unified text feature, movie_profile, was created for every movie by concatenating the following columns:

- *Overview, Plot Keywords, Genres, Director, and Top 5 Cast*.

Text Normalization:

To optimize the text for vectorization, the following NLP techniques were applied:

- **Token Merging:** Names were consolidated (e.g., "Steven Spielberg" $\rightarrow$ "director_StevenSpielberg") to preserve the distinction between roles (directors vs. actors).
- **Lemmatization:** Used WordNetLemmatizer to reduce words to their base roots (e.g., "running" $\rightarrow$ "run").
- **Stop Word Removal:** Common English stop words were removed to reduce noise.

4. Methodology

4.1 Approach A: Traditional Machine Learning (TF-IDF + PCA)

This approach relies on statistical word frequency to determine similarity.

1. **Vectorization:** The TfidfVectorizer was used to convert the movie_profile text into numerical vectors, limiting the feature space to the top 20,000 n-grams (1-gram and 2-gram).
2. **Dimensionality Reduction:** Due to the sparsity of the TF-IDF matrix, Principal Component Analysis (PCA/TruncatedSVD) was applied to reduce the data to 70 components. This retained approximately 10% of the explained variance while significantly reducing computational cost.

4.2 Approach B: Deep Learning (SBERT Embeddings)

To capture semantic context beyond simple word matching, we utilized **Sentence-BERT (SBERT)**.

1. **Model:** The pre-trained sentence-transformers/all-mpnet-base-v2 model was used to encode the movie profiles.
2. **Advantage:** Unlike TF-IDF, SBERT generates dense vector representations that capture the *meaning* of phrases, allowing the system to detect similarity even if two movie descriptions do not share the exact same keywords.
3. **Manifold Learning:** UMAP (Uniform Manifold Approximation and Projection) was employed to reduce these embeddings for visualization and clustering, preserving local neighborhood structures better than PCA.

5. Clustering and Outlier Detection

To improve recommendation speed and relevance, movies were grouped into clusters (semantic genres).

- **Outlier Detection (DBSCAN):** The DBSCAN algorithm was used to detect and remove noise points (outliers) that did not belong to any dense cluster. Approximately 54 outlier movies were removed to clean the dataset.
- **Clustering Algorithms:**

    - **K-Means:** Partitioned data into distinct, non-overlapping groups.
    - **Gaussian Mixture Models (GMM):** A probabilistic approach that assumes clusters follow a Gaussian distribution.
- **Evaluation:** The optimal number of clusters (k) was determined by analyzing Silhouette Scores and AIC/BIC metrics. The analysis suggested that 10 to 12 clusters provided the best segmentation of the movie space.

6. Recommendation Engine Logic

The final recommendation function, recommend_movies, operates on the following logic:

1. **Input:** User queries a specific movie title (e.g., "The Avengers").
2. **Cluster Identification:** The system retrieves the assigned cluster for the input movie.
3. **Similarity Calculation:** Instead of searching the entire database, the system calculates the **Cosine Similarity** between the input movie and other movies *within the same cluster*.
4. **Output:** The system returns the top 10 movies with the highest similarity scores.

Sample Output:

- **Input:** *Interstellar*
- **Recommendations:** *Tenet, Inception, Contact, The Martian*.

7. Conclusion

The project successfully implemented a robust movie recommendation system. The comparison between traditional TF-IDF and modern SBERT embeddings highlighted the superiority of deep learning approaches in capturing semantic nuance. The integration of clustering (GMM) allows the system to scale efficiently by narrowing the search space, ensuring that users receive relevant recommendations in real-time.