Movie Recommendation System

Intro to Problem Solving & Programming – Mini Project

Domain: Entertainment

Author: Deeksha Parmar

25BCY10211

VIT Bhopal University

---

1. Introduction

Entertainment platforms offer thousands of movies, making it difficult for people to decide what to watch next.
Recommendation systems help solve this problem by suggesting movies similar to the user's preferences.

This project presents a simple **content-based movie recommendation system** developed using Python and Jupyter Notebook.
The system analyses the *genres* and *overviews* of movies to compute similarity and suggest related titles.

This project demonstrates the application of basic programming concepts, text processing, and machine learning techniques learned in the course.

---

2. Problem Statement

The problem addressed in this project is:

"How to recommend movies similar to a given movie based on content?"

Users often look for films that match their taste—such as similar genres or story themes.
A recommendation system automates this by comparing movie descriptions and generating suggestions.

---

3. Objectives

The primary objectives of this project are:

- To design a simple movie recommendation model using Python.
- To understand and apply text preprocessing techniques.
- To use TF-IDF vectorization to convert text into numerical features.
- To compute similarity between movies using cosine similarity.
- To provide relevant movie recommendations to users.

---

4. Tools & Technologies Used

Component	Description
Python	Programming language
Jupyter Notebook	Interactive development environment
Pandas	Data handling and preprocessing
Scikit-Learn	TF-IDF Vectorizer & cosine similarity
Machine Learning	Content-based recommendation

---

5. Dataset Description

A small, custom dataset was created **inside the notebook**.
It contains the following attributes:

- **title** — name of the movie
- **genres** — movie category (Action, Thriller, Romance, Sci-Fi, etc.)
- **overview** — short plot summary
- **combined** — merged text (genres + overview) for vectorization

Example dataset:

Title	Genre	Overview
Avatar	Action Adventure	A marine on an alien planet…
Titanic	Drama Romance	A tragic love story on the Titanic…
Inception	Sci-Fi Thriller	A thief enters people's dreams…

---

6. Methodology

This project follows a structured problem-solving approach:

6.1 Step 1 — Data Creation

A small sample dataset was created manually inside the Jupyter Notebook using a Python dictionary and converted into a Pandas DataFrame.

6.2 Step 2 — Text Preprocessing

Genres and overview were combined into one column.
Missing values were filled with empty strings.

```
df["combined"] = df["genres"] + " " + df["overview"]
```

6.3 Step 3 — Feature Extraction (TF-IDF)

TF-IDF (Term Frequency–Inverse Document Frequency) converts text into numerical vectors.

```
vectorizer = TfidfVectorizer(stop_words='english')
vectors = vectorizer.fit_transform(df["combined"])
```

6.4 Step 4 — Similarity Calculation

Cosine similarity evaluates how close one movie is to another based on their vector representations.

6.5 Step 5 — Recommendation Function

A function was created to return the top similar movies:

```
def recommend(movie_title, n=3):
    index = df[df["title"] == movie_title].index[0]
    similarity_scores = cosine_similarity(vectors[index],
vectors).flatten()
    similar_indices = similarity_scores.argsort()[::-1][1:n+1]
    return df["title"].iloc[similar_indices].tolist()
```

---

7. Results

Example Output:

Input:

```
recommend("Avatar")
```

Output:

```
['Avengers', 'Interstellar', 'Inception']
```

The system correctly identifies films similar in themes such as Sci-Fi, action, and adventure.

---

8. Advantages

- Simple, easy-to-understand implementation
- Works with small datasets
- Requires no database or external API
- Uses basic machine learning concepts

---

9. Limitations

- Limited dataset
- Does not consider user ratings or behavior
- No collaborative filtering
- Recommendations depend heavily on text quality

10. Future Enhancements

- Use large real-world datasets like TMDB
- Add user-based collaborative filtering
- Implement a GUI or web interface
- Add more metadata (actor, director, year, keywords)
- Deploy using Flask or Streamlit

---

11. Conclusion

This project successfully demonstrates a **content-based movie recommendation system** using fundamental programming concepts.
It bridges theory and real-world application by using text vectorization and similarity scoring to recommend movies.

It highlights core problem-solving concepts:

- Modular design
- Data preprocessing
- Algorithm development
- Testing and evaluation

---

12. References

- Scikit-Learn Documentation
- Python Pandas Documentation
- Course Notes (Intro to Problem Solving & Programming)