Assignment No 2

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

movies = pd.read_csv("/content/RS-A2_A3_movie.csv")
tags = pd.read_csv("/content/RS-A2_A3_tag.csv")

merged = pd.merge(movies, tags[['movieId', 'tag']], on='movieId', how='left')
merged['tag'] = merged['tag'].fillna('')
merged['content'] = merged['genres'] + " " + merged['tag']

tfidf = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf.fit_transform(merged['content'])

cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)

indices = pd.Series(merged.index, index=merged['title']).drop_duplicates()

def recommend_movies(title, n=5):
    if title not in indices:
        print("Movie not found in dataset.")
        return
    idx = indices[title]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:n+1]
    movie_indices = [i[0] for i in sim_scores]
    print(f"\nMovies similar to '{title}':")
    print(merged['title'].iloc[movie_indices].to_string(index=False))

print("Dataset loaded successfully!")
print("Total movies:", len(merged))

recommend_movies("Heat (1995)", n=5)
recommend_movies("Toy Story (1995)", n=5)

print("\nEvaluation: Cosine similarity used to find nearest movies by content similarity.")
```

Viva-Ready Questions and Answers

Q: What is the main objective of this system?
A: To recommend movies similar to a given movie based on their content (genres and user tags) using text similarity techniques.

Q: Why do we use TF-IDF Vectorizer here?
A: TF-IDF converts text (genres, tags) into numerical feature vectors that highlight important, unique words while reducing the weight of common words.

Q: What does TF-IDF stand for?
A: Term Frequency–Inverse Document Frequency. It measures how important a word is to a document relative to all other documents.

Q: Why is cosine similarity used in this project?
A: Cosine similarity measures the angle between two text vectors and determines how similar two movies are based on their content, regardless of length.

Q: How does cosine similarity work?
A: It computes similarity between two vectors using the formula:

$$\text{cos}(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

Where A and B are TF-IDF vectors. A value close to 1 means movies are similar.

Q: What is the role of merging movie and tag datasets?
A: Merging enriches the movie's information by combining its genres and user-defined tags, leading to better content representation.

Q: Why are missing tags replaced with blank (“”)?
A: Because TF-IDF requires text input; missing (NaN) values would cause errors during vectorization.

Q: What are stop words and why are they removed?
A: Stop words are common words like “the”, “and”, “is” that don’t add meaning. Removing them improves model accuracy by focusing only on informative words.

Q: What is the output of this system?
A: A list of top-N movies that are most similar to the input movie, based on genres and tags.

Q: How do you evaluate the performance of this recommendation model?
A: Qualitatively (by observing if similar movies are recommended) or quantitatively using similarity precision metrics like Precision@K or Recall@K.

Q: What is the difference between content-based and collaborative filtering?
A:

- **Content-based uses item features (like genres, tags).**
- **Collaborative filtering uses user behavior (ratings, interactions).**

Q: What are the advantages of a content-based recommendation system?
A:

1. **Works well for new users (no need for ratings).**
2. **Transparent and explainable recommendations.**
3. **Can handle unseen users if content data is available.**

Q: What are the disadvantages or limitations?
A:

1. **Cold-start problem for new movies with no metadata.**

2. **Limited diversity — keeps recommending similar genre movies.**
3. **Doesn't use other users' preferences.**

Q: Why use cosine similarity instead of Euclidean distance?
A: Cosine similarity focuses on orientation (word usage pattern), not magnitude, which suits text-based data where frequency magnitude is less important.

Q: What libraries from scikit-learn are used here?
A: `TfidfVectorizer` for feature extraction and `cosine_similarity` for similarity computation.

Q: What is the purpose of the `indices` Series?
A: It maps each movie title to its dataset index, enabling quick lookup during similarity calculations.

Q: How does the `recommend_movies()` function work?
A:

1. **Finds the index of the input movie.**
2. **Retrieves similarity scores for all movies.**
3. **Sorts and picks top-N similar movies.**
4. **Displays movie titles as recommendations.**

Q: How would you improve this system?
A:

- **Include more metadata (cast, directors, plot summary).**
- **Use word embeddings (Word2Vec, BERT).**

- **Combine with collaborative filtering to form a hybrid recommender.**

Q: What is the cold start problem?
A: When a new movie with no metadata or a new user with no preferences is added, the system cannot generate recommendations.

Q: Why merge tags and genres together?
A: Tags give user-level descriptive context (like “crime”, “sci-fi”) that complements genres, improving similarity calculation accuracy.

Q: What happens if the movie title is not found in the dataset?
A: The function prints "Movie not found in dataset." since it cannot compute similarity without a valid index.

Q: How is this different from rating-based systems like Netflix’s?
A: Netflix uses hybrid recommenders — combining content-based filtering (movie features) with collaborative filtering (user behavior).

Q: What type of machine learning technique is used here?
A: This is an unsupervised, similarity-based recommendation system, not a predictive supervised model.