INDIVIDUAL PROJECT REPORT ON

Movie Recommendations System

UNDER THE GUIDANCE OF DR. PRADEEP KANDULA

This report outlines the implementation of a movie recommendation system using TF - IDF (Term Frequency-Inverse Document Frequency) and cosine similarity.

1. Feature Extraction using TF - IDF(Term Frequency - Inverse Document Frequency)
TfidfVectorizer is a powerful tool from the sklearn.feature extraction.text module.
Based on the importance of words, it converts unprocessed text data into numerical vectors.

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

The TF-IDF approach takes into account both inverse document frequency, or the importance of a word across all documents, and term frequency, or how frequently a word appears in a document. The significance of every word in the dataset is shown in the feature matrix that is produced.

```
tfidf = TfidfVectorizer()
```

Here first initialize a TfidfVectorizer object named tfidf.

```
features = tfidf.fit_transform(df['clean_input'])
```

The fit_transform method is applied to the clean_input column of a DataFrame (df), creating a matrix of TF-IDF features for each movie title.

2. Cosine Similarity Calculation
Cosine similarity measures the similarity between two non-zero vectors in an inner product space.It figures the cosine of the point between the vectors, which is comparable to the speck item separated by the result of their lengths.Key points:
Cosine similarity does not depend on vector magnitudes, only their angle.
It ranges from -1 (opposite directions) to 1 (same direction), with 0 indicating orthogonality.In the context of movie recommendation, it helps find similar movies based on their feature vectors.

```
from sklearn.metrics.pairwise import cosine_similarity
cosine_sim = cosine_similarity(features, features)
```

3. Recommendation Function
Here we define a function called recommend_movies(title) that takes a movie title as input.
Function Definition;

```
def recommend_movies(title):
    movies = []
    idx = index[index == title].index[0]
    score = pd.Series(cosine_sim[idx]).sort_values(ascending=False)
     top10 = list(score.iloc[1:11].index)
```

If index of the input movie title is equal to an index then Calculate the cosine similarity scores between the input movie and all other movies.
Sort these scores in descending order.
Extract the top 10 similar movie titles based on the cosine similarity scores.

Example :
recommend_movies('Gladiator')
If we call recommend_movies('Gladiator'), it will return a list of movies similar to "Gladiator."
Output:
['Ben-Hur', 'The Sting', 'On the Waterfront', 'Deadpool', 'Star Wars: Episode VI - Return of the Jedi', 'Roman Holiday', 'Léon: The Professional', 'The Martian', 'Kill Bill: Vol. 1', 'The Godfather: Part II']

Additionally, we attempt other film titles like 'The Shawshank Reclamation' or 'The Justice fighters' to get proposals.

BY
VEMANA SUMANTH