```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import random
import tensorflow as tf
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import Input, Conv2D, UpSampling2D, Concatenate
from tensorflow.keras.models import Model
import tensorflow.keras.backend as K
from keras.models import Model, load_model
from keras.layers import Input, BatchNormalization, Activation, Dense, Dropout
import matplotlib.pyplot as plt
import cv2
from tensorflow.keras.applications.mobilenet import preprocess_input
from keras.models import Model, load_model
from keras.layers import Input, BatchNormalization, Activation, Dense, Dropout
from flask import Flask, request, render_template
```

```
app = Flask(__name__)
```

```
import pandas as pd

df = pd.read_csv('/content/drive/MyDrive/Recommender System database/movies.csv')
df.head()
```

	id	title	genres	original_language	overview	popularity	production_companies	release_date	budget	revenu
0	615656	Meg 2: The Trench	Action-Science Fiction-Horror	en	An exploratory dive into the deepest depths of...	8763.998	Apelles Entertainment-Warner Bros. Pictures-di...	2023-08-02	129000000.0	3.520565e+0
1	758323	The Pope's Exorcist	Horror-Mystery-Thriller	en	Father Gabriele Amorth Chief Exorcist of the V...	5953.227	Screen Gems-2.0 Entertainment-Jesus & Mary-Wor...	2023-04-05	18000000.0	6.567582e+0
2	533535	Deadpool & Wolverine	Action-Comedy-Science Fiction	en	A listless Wade Wilson toils away in civilian ...	5410.496	Marvel Studios-Maximum Effort-21 Laps Entertai...	2024-07-24	200000000.0	1.326387e+0
3	667538	Transformers: Rise of the Beasts	Action-Adventure-Science Fiction	en	When a new threat capable of destroying the en...	5409.104	Skydance-Paramount-di Bonaventura Pictures-Bay...	2023-06-06	200000000.0	4.070455e+0
4	693134	Dune: Part Two	Science Fiction-Adventure	en	Follow the mythic journey of Paul Atreides as ...	4742.163	Legendary Pictures	2024-02-27	190000000.0	6.838137e+0

```
df.drop('id',axis = 1, inplace =True)
```

Now we'll compute Term Frequency-Inverse Document Frequency (TF-IDF) vectors for each overview. This is to be done for word preprocessing.

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

```
df.drop_duplicates(inplace=True, ignore_index=True)
#df = df.groupby('title').first().reset_index()
df.fillna(value={i: ' ' for i in ['overview', 'genres', 'keywords', 'credits']}, inplace=True)

# lambda func for str split join
strOp= lambda x: ' '.join(x.split('-'))

df.overview = df.overview + df.keywords.apply(strOp) + df.genres.apply(strOp) + df.credits.apply(lambda x: ' '.join(x.replace(' ', '').split

#Define a TF-IDF Vectorizer Object. Remove all english stop words such as 'the', 'a'
tfidf = TfidfVectorizer(stop_words='english')

#Construct the required TF-IDF matrix by fitting and transforming the data
tfidf_matrix = tfidf.fit_transform(df['overview'])

display(pd.DataFrame(
    tfidf_matrix[:10, 7000:7070].toarray(),
    columns= tfidf.get_feature_names_out()[7000:7070],
    index = df.title[:10]).round())

print(tfidf_matrix.shape)
```

title	aadisesh	aadisingh	aaditha	aaditiagarwal	aaditipohankar	aaditya	aadityapandey	aadityasingh	aadityav	aadland	...
Meg 2: The Trench	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
The Pope's Exorcist	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
Deadpool & Wolverine	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
Transformers: Rise of the Beasts	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
Dune: Part Two	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
Ant-Man and the Wasp: Quantumania	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
Creed III	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
Insidious: The Red Door	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
Despicable Me 4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
Spider-Man: Across the Spider-Verse	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...

10 rows × 70 columns

```
(722303, 963018)
```

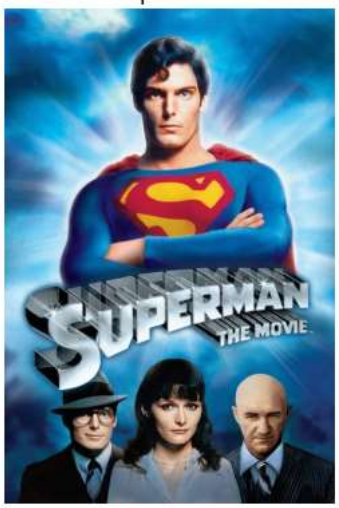
```
from sklearn.metrics.pairwise import cosine_similarity
```

```
from skimage import io
```

```
# Function that takes in movie title as input and outputs most similar movies
def get_recommendations(title):
    # Get the index of the movie that matches the title
    idx = df.index[df['title'] == title][0]
    # show given movie poster
    try:
        a = io.imread(f'https://image.tmdb.org/t/p/w500/{df.loc[idx, "poster_path"]}')
        plt.imshow(a)
```

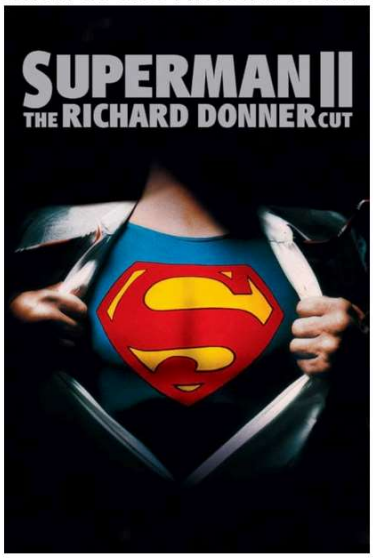
```
            plt.axis('off')
            plt.title(title)
            plt.show()
        except:pass

        print('Recommendations\n')


        # Get the pairwsie similarity scores of all movies with that movie
        sim_scores = list(enumerate(
            cosine_similarity(
                tfidf_matrix,
                tfidf_matrix[idx])))

        # Sort the movies based on the similarity scores
        sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

        # Get the scores of the 10 most similar movies
        sim_scores = sim_scores[1:10]

        # Get the movie indices
        movie_indices = [i[0] for i in sim_scores]

        # Return the top 10 most similar movies
        result = df.iloc[movie_indices]

        # show reco. movie posters
        fig, ax = plt.subplots(3, 3, figsize=(15,20))
        ax=ax.flatten()
        for i, j in enumerate(result.poster_path):
            try:
                ax[i].axis('off')
                ax[i].set_title(result.iloc[i].title,fontsize=22)
                a = io.imread(f'https://image.tmdb.org/t/p/w500/{j}')
                ax[i].imshow(a)
            except: pass
        fig.tight_layout()
        fig.show()


    get_recommendations("Superman")
```

Superman




```
Recommendations
```

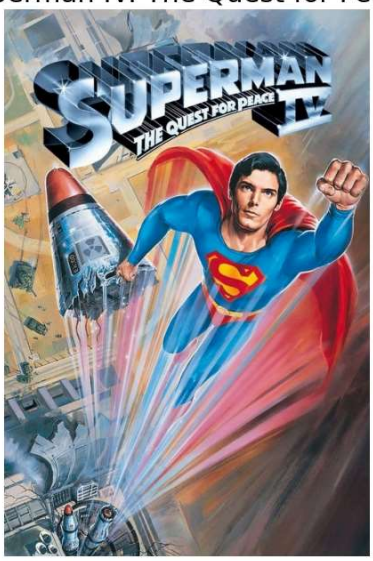
Superman II: The Richard Donner Cut



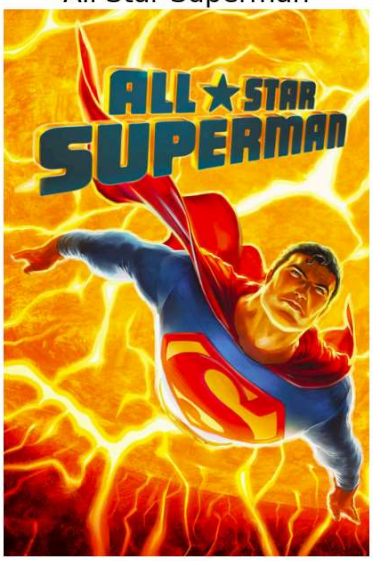
Superman II

Superman is Here!

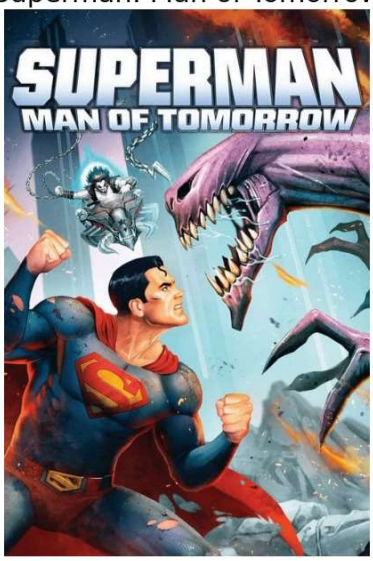
Superman IV: The Quest for Peace



All Star Superman



Superman: Man of Tomorrow



Superman Returns

"Superman Can't Kill People" - A Kryptonian Epic Short

Stealth


SUPERMAN
RETURNS