Movie Recommendation System Using NLP & TMDB API

Author: Jubair Abdulla

Date: 2025.06.26

CONTENTS

01

Dataset Source

Dataset Source

1. Dataset Link
   - Dataset:\https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata

2. Files used
   - tmdb_5000_credits.csv
   - tmdb_5000_movies.csv

02

Data Preprocessing

Data Preprocessing

Overview

- Initially found 3 missing overviews — manually created and updated them.

Selected columns

- genres, movie_id, keywords, overview, title, cast, crew

Data Extraction

- From cast, extracted top 3 actor names.
- From crew, selected only the director name.

Final Columns

- Combined all relevant text into a single column called tags.
- Final columns used: movie_id, title, tags

Preprocessing

- Applied spaCy model (en_core_web_sm) for preprocessing and lemmatization.

03

Feature Engineering

Feature Engineering

1 Tag Weighting

- Attempted custom weighting for tags and overview, but it didn't improve results, so discarded.

2 Comparison

- Compared CountVectorizer and TF-IDF:
- CountVectorizer gave better similarity results in this case.

04

Recommendation Logic

Recommendation Logic

Similarity Calculation

- Used cosine similarity on vectorized data.

Movie Retrieval

- For each selected movie:
- Retrieved top 5 similar movies using similarity scores.

05

TMDB API Integration

Details Fetched

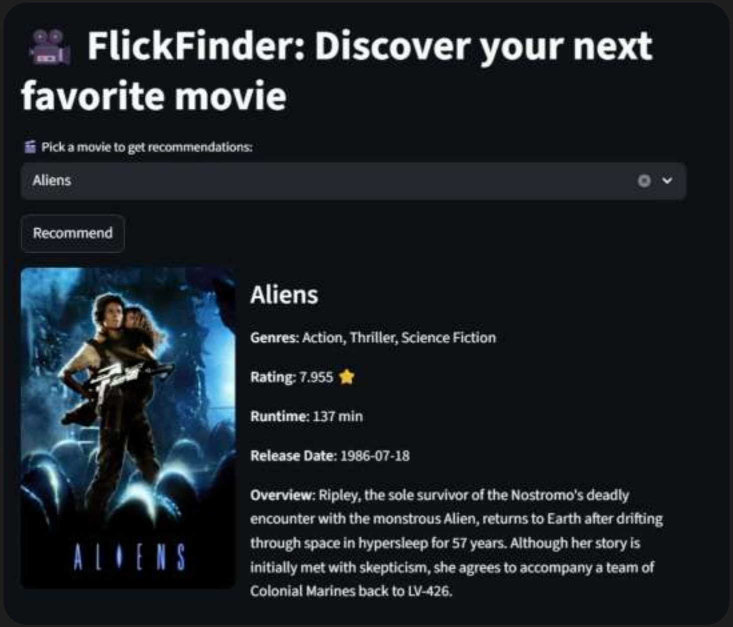
Used TMDB API to fetch details:

- Poster
- Title
- Genres
- Rating
- Overview
- Runtime
- Release Date

06

Streamlit Web App

Streamlit Web App



FlickFinder: Discover your next favorite movie

Pick a movie to get recommendations:

Aliens

Recommend

Aliens

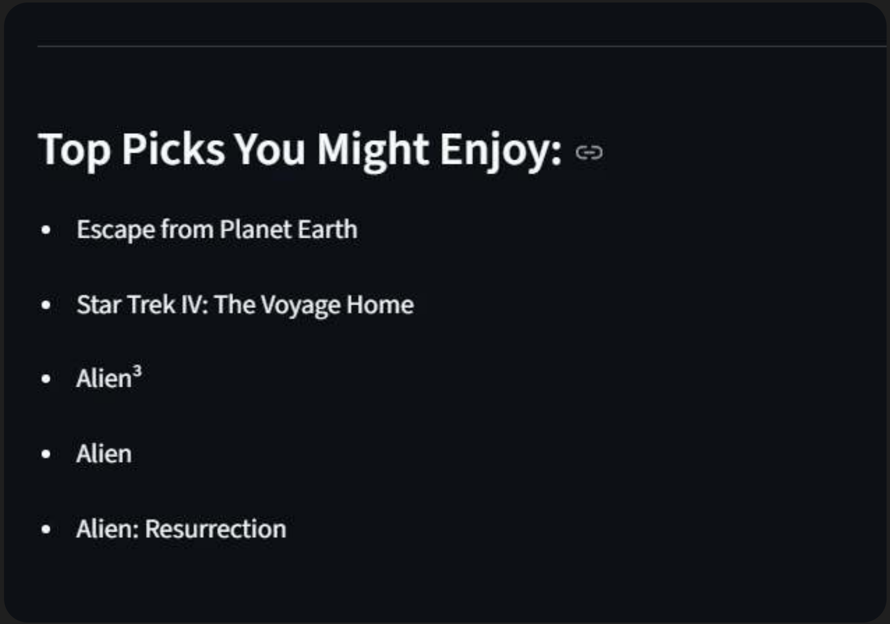
Genres: Action, Thriller, Science Fiction

Rating: 7.955 ⭐

Runtime: 137 min

Release Date: 1986-07-18

Overview: Ripley, the sole survivor of the Nostromo's deadly encounter with the monstrous Alien, returns to Earth after drifting through space in hypersleep for 57 years. Although her story is initially met with skepticism, she agrees to accompany a team of Colonial Marines back to LV-426.



Top Picks You Might Enjoy:

- Escape from Planet Earth
- Star Trek IV: The Voyage Home
- Alien^3
- Alien
- Alien: Resurrection

07

Known Issue

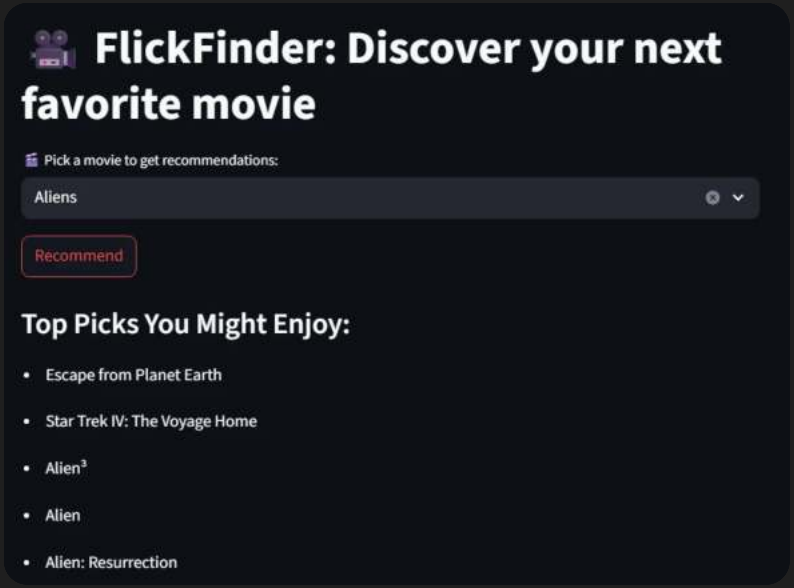
Known Issue

1 API Reliability

- TMDB API sometimes fails.

2 Fallback Mechanism

- Fallback: Shows only the recommended movie titles without extra details.



FlickFinder: Discover your next favorite movie

Pick a movie to get recommendations:

Aliens

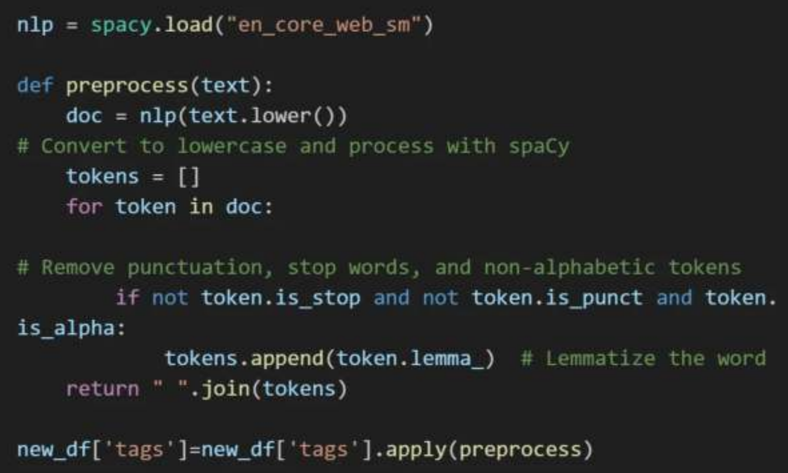
Recommend

Top Picks You Might Enjoy:

- Escape from Planet Earth
- Star Trek IV: The Voyage Home
- Alien³
- Alien
- Alien: Resurrection

Code Snippets

Preprocessing and Lemmatization



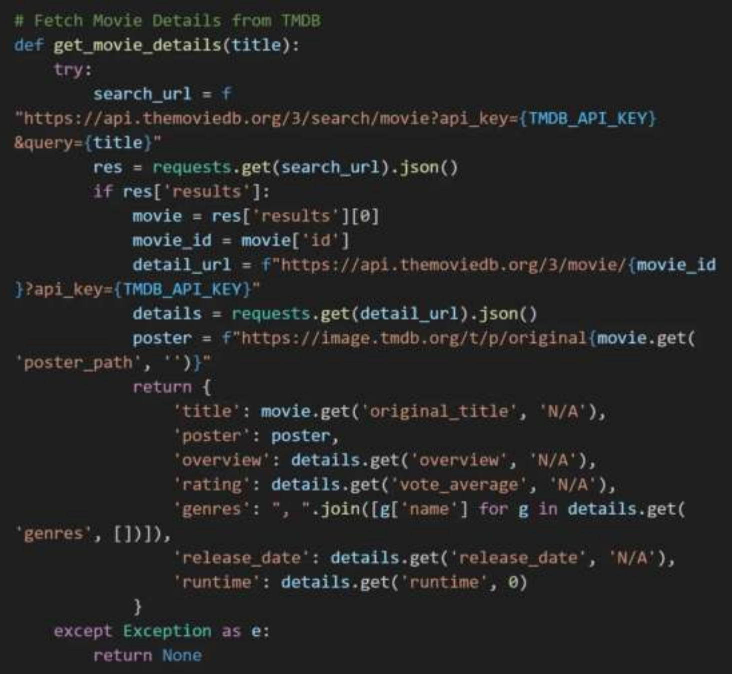
```
nlp = spacy.load("en_core_web_sm")

def preprocess(text):
    doc = nlp(text.lower())
# Convert to lowercase and process with spaCy
    tokens = []
    for token in doc:

# Remove punctuation, stop words, and non-alphabetic tokens
        if not token.is_stop and not token.is_punct and token.
is_alpha:
            tokens.append(token.lemma_)  # Lemmatize the word
    return " ".join(tokens)

new_df['tags']=new_df['tags'].apply(preprocess)
```

API Inegration



```
# Fetch Movie Details from TMDB
def get_movie_details(title):
    try:
        search_url = f
"https://api.themoviedb.org/3/search/movie?api_key={TMDB_API_KEY}
&query={title}"
        res = requests.get(search_url).json()
        if res['results']:
            movie = res['results'][0]
            movie_id = movie['id']
            detail_url = f"https://api.themoviedb.org/3/movie/{movie_id
}?api_key={TMDB_API_KEY}"
            details = requests.get(detail_url).json()
            poster = f"https://image.tmdb.org/t/p/original{movie.get(
'poster_path', '')}"
            return {
                'title': movie.get('original_title', 'N/A'),
                'poster': poster,
                'overview': details.get('overview', 'N/A'),
                'rating': details.get('vote_average', 'N/A'),
                'genres': ", ".join([g['name'] for g in details.get(
'genres', [])]),
                'release_date': details.get('release_date', 'N/A'),
                'runtime': details.get('runtime', 0)
            }
    except Exception as e:
        return None
```

Vector Conversion and Recommendation

```
# Initialize CountVectorizer Vectorizer
cv=CountVectorizer(max_features=5000)
# Applying it on 'tags' column
vectors_countvector = cv.fit_transform(new_df[
'tags']).toarray()
#calculating similarity score
similarity_countvector=cosine_similarity(
vectors_countvector)

# Function to display recommended movies.
def recommed(movie):
    movi_index=new_df[new_df['title']==movie
].index[0]
    distance=similarity_countvector[movi_index]

    movies_list=sorted(list(enumerate(distance
)),reverse=True,key=lambda x:x[1])[1:6]
    for i in movies_list:
        print(new_df.iloc[i[0]].title)

recommed('Avatar')
#output
# Falcon Rising
# Aliens
# Independence Day
# Titan A.E.
# Small Soldiers
```

Thank You