```
# Install dependencies
!pip install sentence-transformers textblob

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from textblob import TextBlob
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from sentence_transformers import SentenceTransformer
```

```
Requirement already satisfied: sentence-transformers in /usr/local/l
Requirement already satisfied: textblob in /usr/local/lib/python3.12
Requirement already satisfied: transformers<5.0.0,>=4.41.0 in /usr/l
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dis
Requirement already satisfied: torch>=1.11.0 in /usr/local/lib/pythc
Requirement already satisfied: scikit-learn in /usr/local/lib/python
Requirement already satisfied: scipy in /usr/local/lib/python3.12/di
Requirement already satisfied: huggingface-hub>=0.20.0 in /usr/local
Requirement already satisfied: Pillow in /usr/local/lib/python3.12/d
Requirement already satisfied: typing_extensions>=4.5.0 in /usr/loca
Requirement already satisfied: nltk>=3.9 in /usr/local/lib/python3.1
Requirement already satisfied: filelock in /usr/local/lib/python3.12
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/py
Requirement already satisfied: packaging>=20.9 in /usr/local/lib/pyt
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3
Requirement already satisfied: requests in /usr/local/lib/python3.12
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in /usr/local/li
Requirement already satisfied: click in /usr/local/lib/python3.12/di
Requirement already satisfied: joblib in /usr/local/lib/python3.12/d
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/pyt
Requirement already satisfied: setuptools in /usr/local/lib/python3.
Requirement already satisfied: sympy>=1.13.3 in /usr/local/lib/pythc
Requirement already satisfied: networkx>=2.5.1 in /usr/local/lib/pyt
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/d
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.6.77 in /u
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.6.77 in
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.6.80 in /u
Requirement already satisfied: nvidia-cudnn-cu12==9.10.2.21 in /usr/
Requirement already satisfied: nvidia-cublas-cu12==12.6.4.1 in /usr/
Requirement already satisfied: nvidia-cufft-cu12==11.3.0.4 in /usr/l
Requirement already satisfied: nvidia-curand-cu12==10.3.7.77 in /usr
Requirement already satisfied: nvidia-cusolver-cu12==11.7.1.2 in /us
Requirement already satisfied: nvidia-cusparse-cu12==12.5.4.2 in /us
Requirement already satisfied: nvidia-cusparselt-cu12==0.7.1 in /usr
Requirement already satisfied: nvidia-nccl-cu12==2.27.5 in /usr/loca
Requirement already satisfied: nvidia-nvshmem-cu12==3.3.20 in /usr/l
```

```
Requirement already satisfied: nvidia-nvtx-cu12==12.6.77 in /usr/loc
Requirement already satisfied: nvidia-nvjitlink-cu12==12.6.85 in /us
Requirement already satisfied: nvidia-cufile-cu12==1.11.1.6 in /usr/
Requirement already satisfied: triton==3.5.0 in /usr/local/lib/pytho
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3
Requirement already satisfied: tokenizers<=0.23.0,>=0.22.0 in /usr/l
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/li
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/pyt
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/loca
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/
```

```
from google.colab import files
uploaded = files.upload()   # select the file you downloaded, e.g.

import pandas as pd
df = pd.read_csv("kdrama_DATASET.csv")   # use the exact filename s
print(df.shape)
print(df.columns)
df.head()
```

Choose Files no files selected Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving kdrama_DATASET.csv to kdrama_DATASET (1).csv
(350, 9)
Index(['Rank', 'Title', 'Year of release', 'Number of Episodes', 'Ra
       'Description', 'Genre', 'Tags', 'Actors'],
      dtype='object')
```

	Rank	Title	Year of release	Number of Episodes	Rating	Description	Genre
0	#1	Move to Heaven	2021	10	9.2	Geu Roo is a young autistic man. He works for ...	Life, Drama, Family

```
# Flexible column mapping: rename to a standard schema if present
```

```
col_map_candidates = {
    'title': ['Title', 'title', 'Drama_Name', 'Drama Name', 'Name']
    'year': ['Year', 'Released Year', 'year', 'Release Year'],
    'episodes': ['Episodes', 'Episode', 'episodes'],
    'rating': ['Rating', 'Score', 'rating'],
    'description': ['Description', 'Plot', 'Synopsis', 'description
    'genre': ['Genre', 'Genres', 'genre'],
    'language': ['Language', 'Lang', 'language', 'Country']  # fall
}

def pick_col(df, candidates, default=None):
    for c in candidates:
        if c in df.columns:
            return c
    return default

std_cols = {k: pick_col(df, v) for k, v in col_map_candidates.items

# Create a clean dataframe with standardized names
df_clean = pd.DataFrame({
    'title': df[std_cols['title']] if std_cols['title'] else df.ilo
    'year': df[std_cols['year']] if std_cols['year'] else np.nan,
    'episodes': df[std_cols['episodes']] if std_cols['episodes'] el
    'rating': df[std_cols['rating']] if std_cols['rating'] else np.
    'description': df[std_cols['description']] if std_cols['descrip
    'genre': df[std_cols['genre']] if std_cols['genre'] else "",
    'language': df[std_cols['language']] if std_cols['language'] el
})

# Basic cleaning
for c in ['title', 'description', 'genre', 'language']:
    df_clean[c] = df_clean[c].fillna("").astype(str).str.strip()

for c in ['year', 'episodes', 'rating']:
    df_clean[c] = pd.to_numeric(df_clean[c], errors='coerce')

df = df_clean.drop_duplicates(subset=['title']).reset_index(drop=Tr
print("Normalized columns:", df.columns.tolist())
df.head()
```

```
Normalized columns: ['title', 'year', 'episodes', 'rating', 'descrip
```

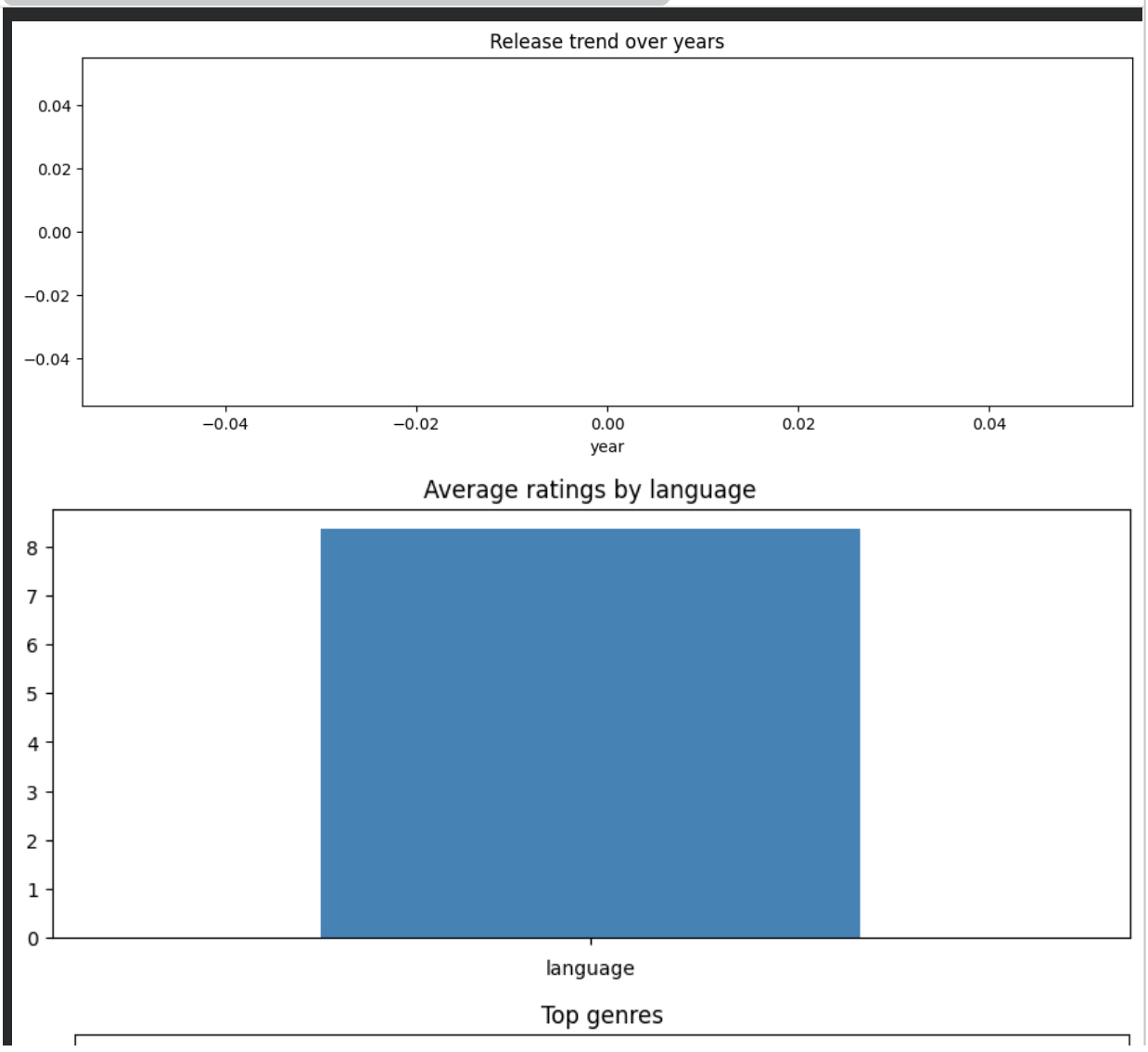
	title	year	episodes	rating	description	genre	languag
0	Move to Heaven	NaN	NaN	9.2	Geu Roo is a young autistic man. He works for ...	Life, Drama, Family	

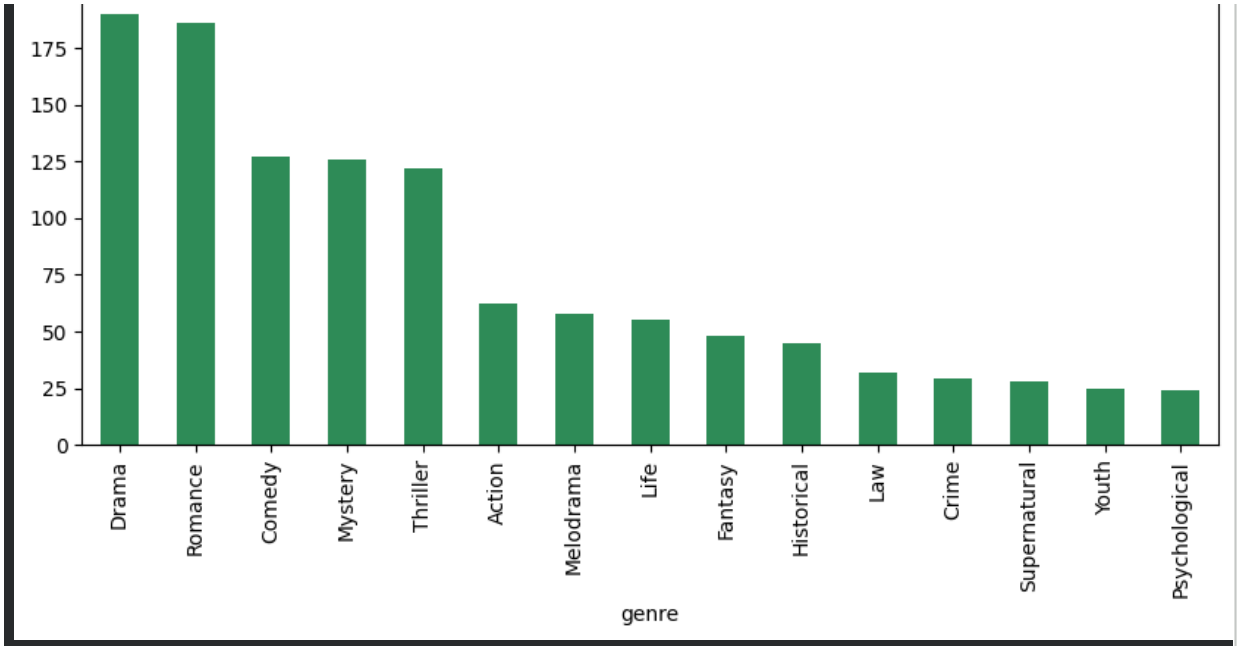
1	Twinkling Watermelon	NaN	NaN	9.2	In 2023, high school student Eun Gyeol, a CODA...	Romance, Youth, Drama, Fantasy

```
plt.figure(figsize=(12,4))
df['year'].dropna().value_counts().sort_index().plot(kind='line', c
plt.title("Release trend over years")
plt.show()

plt.figure(figsize=(10,4))
df.groupby('language')['rating'].mean().sort_values(ascending=False
plt.title("Average ratings by language")
plt.show()

plt.figure(figsize=(10,4))
df['genre'].str.split(',').explode().str.strip().value_counts().hea
plt.title("Top genres")
plt.show()
```




175
150
125
100
75
50
25
0
Drama
Romance
Comedy
Mystery
Thriller
Action
Melodrama
Life
Fantasy
Historical
Law
Crime
Supernatural
Youth
Psychological
genre

```
# Sentiment polarity on description
def get_sentiment_label(text):
    p = TextBlob(text).sentiment.polarity
    if p > 0.2: return "positive"
    if p < -0.2: return "negative"
    return "neutral"

df['sentiment'] = df['description'].apply(get_sentiment_label)

# Unified text feature: genre + description + language + sentiment
df['features_text'] = (
    df['genre'].fillna("") + " | " +
    df['language'].fillna("") + " | " +
    df['sentiment'].fillna("") + " | " +
    df['description'].fillna("")
)
```

```
# Multilingual semantic embeddings (much stronger than TF-IDF)
model = SentenceTransformer('paraphrase-multilingual-MiniLM-L12-v2'
embeddings = model.encode(df['features_text'].tolist(), convert_to_

# Cosine similarity matrix
sim_matrix = cosine_similarity(embeddings, embeddings)
```

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your se
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to
  warnings.warn(
modules.json: 100% 229/229 [00:00<00:00, 4.08kB/s]
config_sentence_transformers.json: 100% 122/122 [00:00<00:00, 2.94kB/s]
README.md: 3.89k/? [00:00<00:00, 67.9kB/s]
sentence_bert_config.json: 100% 53.0/53.0 [00:00<00:00, 2.75kB/s]
config.json: 100% 645/645 [00:00<00:00, 12.1kB/s]
model.safetensors: 100% 471M/471M [00:09<00:00, 47.3MB/s]
tokenizer_config.json: 100% 480/480 [00:00<00:00, 41.9kB/s]
tokenizer.json: 100% 9.08M/9.08M [00:00<00:00, 32.9MB/s]
special_tokens_map.json: 100% 239/239 [00:00<00:00, 22.4kB/s]
config.json: 100% 190/190 [00:00<00:00, 8.63kB/s]
```

```
def recommend_titles(title, df, sim_matrix, top_k=10):
    # Look up index safely
    matches = df.index[df['title'].str.lower() == title.lower()].to
    if not matches:
        return []
    idx = matches[0]

    scores = list(enumerate(sim_matrix[idx]))
    scores = sorted(scores, key=lambda x: x[1], reverse=True)

    recs = []
    for j, s in scores[1:top_k+1]:
        recs.append({'title': df.iloc[j]['title'], 'score': float(s
    return recs

# Example
seed = "The Tale of Nokdu"  # change as you like
recs = recommend_titles(seed, df, sim_matrix, top_k=10)
recs[:5]
```

```
[{'title': 'Hide', 'score': 0.6120591163635254},
 {'title': 'Bitch X Rich', 'score': 0.5949466228485107},
 {'title': 'Kokdu: Season of Deity', 'score': 0.591451108455658},
 {'title': 'Flex X Cop', 'score': 0.5794633626937866},
 {'title': 'High School Return of a Gangster', 'score':
0.5778119564056396}]
```

```
# Overlap-based metrics
def precision_at_k(recommended_titles, relevant_titles, k=5):
    rec_at_k = [r['title'] for r in recommended_titles[:k]]
    hits = len(set(rec_at_k) & set(relevant_titles))
    return hits / max(k, 1)

def recall_at_k(recommended_titles, relevant_titles, k=5):
    rec_at_k = [r['title'] for r in recommended_titles[:k]]
    hits = len(set(rec_at_k) & set(relevant_titles))
    return hits / max(len(relevant_titles), 1)

def ndcg_at_k(recommended_titles, relevant_titles, k=5):
    rec_at_k = [r['title'] for r in recommended_titles[:k]]
    dcg = 0.0
    for i, t in enumerate(rec_at_k):
        if t in relevant_titles:
            dcg += 1.0 / np.log2(i + 2)
    idcg = sum(1.0 / np.log2(i + 2) for i in range(min(len(relevant
    return (dcg / idcg) if idcg > 0 else 0.0
```

```
# Helper: auto-generate a plausible relevant set from genre + langu
def auto_relevant_set(seed_title, df, max_size=10):
    row = df[df['title'].str.lower() == seed_title.lower()]
    if row.empty:
        return []
    g = set([x.strip().lower() for x in row.iloc[0]['genre'].split(
    lang = row.iloc[0]['language'].strip().lower()

    candidates = df[
        (df['language'].str.lower().str.strip() == lang) &
        (df['title'].str.lower() != seed_title.lower())
    ].copy()

    def genre_overlap(genres):
        gg = set([x.strip().lower() for x in str(genres).split(',')
        return len(g & gg)

    candidates['genre_overlap'] = candidates['genre'].apply(genre_o
    relevant = candidates.sort_values(['genre_overlap', 'rating'],
    return relevant

# Dashboard: prints multiple metrics
def evaluate_title(seed_title, df, sim_matrix, k=5):
    recommended = recommend_titles(seed_title, df, sim_matrix, top_
    relevant = auto_relevant_set(seed_title, df, max_size=10)

    p = precision_at_k(recommended, relevant, k)
    r = recall_at_k(recommended, relevant, k)
    n = ndcg_at_k(recommended, relevant, k)

    # Coverage: across a sample of seeds
    sample_seeds = df['title'].head(50).tolist()
    all_recs = []
    for s in sample_seeds:
        all_recs += [x['title'] for x in recommend_titles(s, df, si
    coverage = len(set(all_recs)) / len(df)

    # Diversity: average pairwise dissimilarity in the recommendati
    idxs = []
    titles_in_rec = [x['title'] for x in recommended[:k]]
    for t in titles_in_rec:
        m = df.index[df['title'].str.lower() == t.lower()].tolist()
        if m: idxs.append(m[0])
    if len(idxs) > 1:
        sims = []
        for i in range(len(idxs)):
            for j in range(i+1, len(idxs)):
```

```
                sims.append(sim_matrix[idxs[i], idxs[j]])
        diversity = 1 - (np.mean(sims) if sims else 1.0)
    else:
        diversity = 0.0

    # Novelty: recommend less-popular (proxy: lower mean rating ran
    # If rating missing for some rows, treat as mid-range
    ratings = []
    for t in titles_in_rec:
        rr = df.loc[df['title'].str.lower() == t.lower(), 'rating']
        ratings.append(rr.iloc[0] if len(rr) else np.nan)
    ratings = pd.Series(ratings, dtype=float)
    novelty = 1 - ratings.rank(pct=True).mean() if len(ratings) els

    print(f"Seed: {seed_title}")
    print(f"Recommended (top {k}): {[x['title'] for x in recommende
    print(f"Auto relevant set (size {len(relevant)}): {relevant}")
    print(f"Precision@{k}: {p:.3f}")
    print(f"Recall@{k}: {r:.3f}")
    print(f"NDCG@{k}: {n:.3f}")
    print(f"Coverage (sample 50 seeds): {coverage:.3f}")
    print(f"Diversity (1 - avg similarity): {diversity:.3f}")
    print(f"Novelty (1 - avg rating percentile): {novelty:.3f}")

# Example evaluation
evaluate_title("The Tale of Nokdu", df, sim_matrix, k=5)
```

```
Seed: The Tale of Nokdu
Recommended (top 5): ['Hide', 'Bitch X Rich', 'Kokdu: Season of Deit
Auto relevant set (size 10): ['Alchemy of Souls', 'Mr. Queen', 'Strc
Precision@5: 0.000
Recall@5: 0.000
NDCG@5: 0.000
Coverage (sample 50 seeds): 0.411
Diversity (1 - avg similarity): 0.407
Novelty (1 - avg rating percentile): 0.400
```

```
import ipywidgets as widgets
from IPython.display import display

titles = sorted(df['title'].unique().tolist())[:500]  # cap for drop

dropdown = widgets.Dropdown(options=titles, description='Pick a dram
out = widgets.Output()

def on_change(change):
    if change['name'] == 'value' and change['new']:
        out.clear_output()
        with out:
            evaluate_title(change['new'], df, sim_matrix, k=5)

dropdown.observe(on_change, names='value')
display(dropdown, out)
```