Overview

This project builds several board-
game recommender systems using the `All Files` dataset scraped from BoardGameGeek.
It covers collaborative filtering, hybrid deep learning, a neural matrix factorization model,
an advanced NLP content-based recommender, and an optional ensemble of the trained models.

Data

- `All Files/cleaned_statistics`: 119,978 games with metadata (players, play time, ratings, categories, mechanics, etc.)
- `All Files/ratings1.csv`…`ratings38.csv`: ~3.7 M user ratings for the 2,000 most reviewed games (game id, user, rating, comment)
- Additional artifacts (e.g., `games_to_*`, `games_with_descriptions`) can be regenerated by re-running the scrapers, but the notebook works off the provided files.

Workflow

1. **Setup**

Upload the `All Files` folder to the
Colab workspace. `advanced_bgg_recommender_system.ipynb` auto-detects the location
of `cleaned_statistics` and the ratings CSVs.For reasonable training time, enable a T4
GPU in Google Colab (Runtime → Change runtime type → GPU). GPU detection is
automatic and the notebook adjusts batch sizes and memory growth.

2. **Preprocessing**
   - Load the data into Pandas, convert ratings to numeric, and filter out invalid rows.
   - Restrict to the most-reviewed games (2,000 by default) and label-encode game/user IDs.
   - Standardize numeric features and create combined text fields for content-based models.
   - A configuration block lets you sample down the dataset (`USE_SAMPLE=True`) for much faster experimentation while maintaining structure (e.g., cap at 500 games / 100k ratings).
3. **Models**
   - **Neural Collaborative Filtering (NCF)**
   - **Hybrid Model** (user/item embeddings + game feature vectors)
   - **Neural Matrix Factorization**
   - **Advanced NLP Content-Based** recommender (Sentence Transformers or TF-IDF)
   - **Ensemble** (optional combination of the trained neural models)
4. **Why the Ensemble is in a Separate Cell**

Ensemble evaluation is more time-consuming because it constructs a game-
feature lookup and scores each test interaction with multiple underlying models.
To save time during repeated runs, this step is isolated behind a toggle (`RUN_ENSEMBLE_EVAL`).

Leaving the toggle off keeps experiments quick;
turning it on computes ensemble metrics when you're ready.

5. **Evaluation Metrics**
    - Collaborative models are regressing ratings, so they use error-based scores (RMSE/MAE/MSE).
    - The NLP content-based recommender produces ranked lists rather than numeric ratings. Therefore we assess it with ranking metrics: `Hit@K`, `Precision@K`, and `MRR@K`, plus coverage statistics. These better reflect how well the model surfaces relevant games in a top-N list.
    - The visualization cell plots collaborative errors separately from the content-based ranking scores.
6. **Visualization & Recommendation Examples**
    - Comparative bar charts for collaborative RMSE/MAE/MSE.
    - Separate bar chart for content-based ranking metrics.
    - Utility cells demonstrate top-N recommendations for specific users and the most similar games given a title.

Recommended Workflow

1. Run the notebook from top to bottom after uploading data.
2. Start with `USE_SAMPLE=True` for rapid iteration;
   switch to full data for final evaluation (GPU strongly recommended).
3. Train NCF, Hybrid, and Neural MF models; evaluate collaborative metrics.
4. Run the content-based evaluation cell to obtain ranking metrics.
5. Optionally toggle ensemble evaluation when you need the aggregated model metrics.
6. Review the comparison plots and recommendation examples,
   then adjust configuration or model settings as needed.
