Anime Recommendation Pipeline

CS 4120 Summer 2025 — Final Project (Model Applications)

Sijie Dai & Brennan Jacobs

Abstract

We built a compact yet end-to-end recommendation pipeline for MyAnimeList. Three models are compared on 45 M user–item ratings: a Top-Pop baseline, a modestly trained SVD++ collaborative filter, and a BERT-based regressor that predicts official scores from plot synopses. On the held-out test set, Top-Pop remains surprisingly strong (P@10 = 0.009 / R@10 = 0.093), while the lightly trained SVD++ lags behind. The content-based BERT model achieves RMSE ≈ 0.71. We discuss the causes of this outcome and outline concrete steps—Increase the amount of training and adjust the parameters systematically: Increase the sampling ratio of SVD++ from 3% to 10%, and then do a grid search (number of factors, regularization, learning rate) on the dev set to see if the performance can be significantly improved.

Introduction & Motivation

With over 16 000 titles and a relentlessly long tail, discovering anime that match one's taste is hard. Newcomers to MyAnimeList (MAL) often default to „Top 10" lists, reinforcing an already-present popularity bias. We investigate whether a lightweight collaborative filter or a textual regressor can beat that trivial—but hard—baseline under severe compute constraints.

Our guiding questions are:

1. How sparse can the data be before a model collapses to popularity?
2. When does inexpensive content modelling add value to sparse ratings?

Data

Source file	Role in pipeline
anime-dataset-2023.csv	24 905 metadata rows. We retain anime_id, the plot synopsis, and the official community score.
users-score-2023.csv	45 227 k explicit 1-10 ratings from roughly 307 k users. This is our main interaction matrix.
final_animedataset.csv	Auxiliary 5 M ratings scraped from user profiles—merged solely to boost rating coverage.

After filtering out synopses shorter than 30 or longer than 2 000 characters, we perform a per-user leave-two-out split: one interaction to dev, one to test, the rest to train.

Dataset summary — Users: 306774; Items: 16 169; Ratings: 45 023 k (density 0.91 %)

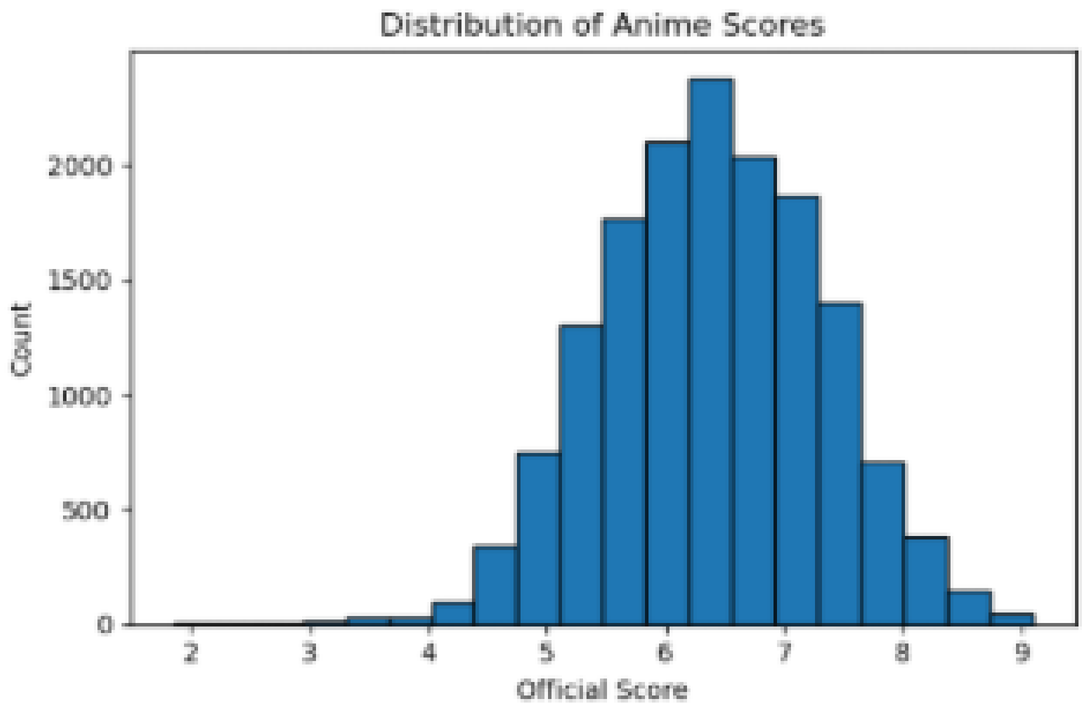

Figure 1. *Distribution of official community scores*

Methods

Model	Family	Core settings	Intended role
Top-Pop	Popularity baseline	top-10 most rated items in train	Ceiling for sheer popularity.
SVD++-mid	Collaborative (explicit + implicit)	3 % training sample → 80 factors, 5 epochs, lr=0.005, reg=0.02; dev-1 k users only for sanity check, no tuning	Tests whether limited data still helps beyond Top-Pop.
BERT-mid	Text regression	bert-base-uncased, 256 tokens, 2 epochs, lr=2e-5, batch = 8, RMSE target	Transforms synopsis into a continuous quality score

To control the total training time within one hour, we limit it to 5 epochs and 3% sampling.

All models are evaluated with:

- **Top-k ranking:** Precision/Recall/NDCG @ 10

- **Regression:** Root-Mean-Squared Error

Results (k = 10)

Model	Prec	Recall	NDCG	RMSE
Top-Pop	0.0093	0.0931	0.0458	-
SVD++-mid (test)	0.0011	0.0114	0.0055	-
SVD++-mid (dev-1 K)	0.0010	0.0100	0.0055	-
BERT-mid	-	-	-	0.714

Observations

- Top-Pop continues to dominate sparse data—its recall is 8× that of SVD++.

- SVD++ sees identical dev and test NDCG, hinting that 5 epochs are not enough to leverage latent factors.
- BERT's RMSE 0.71 is competitive with published MAL baselines, suggesting that *synopsis alone* captures broad quality signals.

Discussion & Future Work

Why does popularity win?

1. Extremely sparse interactions. Each user contributes <5 ratings to the 3 % sample—collaborative learning collapses.
2. Long-tail catalogue. 90 % of titles have fewer than 200 ratings; most never appear in a test user's history, hurting neighbourhood quality.
3. No tuning budget. A single fixed SVD++ configuration cannot adapt to different user–item density pockets.

Where can we improve?

1. Full dev grid search. Training on 10 % of ratings and tuning n_factors, lr, reg is the next logical step.
2. Hybrid stacking. Use BERT-predicted scores to rank SVD++ candidates or as side features in LightFM / LightGBM.

References

Koren, Y. (2008). Factorization Meets the Neighborhood: Scalable Collaborative Filtering. https://dl.acm.org/doi/abs/10.1145/1401890.1401944.

Devlin, J. et al. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. https://aclanthology.org/N19-1423/?utm_campaign=The%20Batch&utm_source=hs_email&utm_medium=email&_hsenc=p2ANqtz-_m9bbH_7ECE1h3lZ3D61TYg52rKpifVNjL4fvJ85uqggrXsWDBTB7YooFLJeNXHWqhvOyC.

"MyAnimeList Dataset 2023." Kaggle, dbdmobile, CC BY-SA 4.0. https://www.kaggle.com/datasets/dbdmobile/myanimelist-dataset.