Jacobs, Brennan and Dai, Sijie

Felix Muzny

CS 4120

June 18, 2025

CS 4120 Final Project Report

By Brennan Jacobs and Sijie Dai

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 and Motivation

For our CS 4120 final project, we focused on building a recommendation engine for anime for two reasons: to see what future anime could theoretically become popular enough to warrant its creation and to see if building a better media recommendation engine was possible.

First, on the matter of anime creation: the budget for anime production does not entirely come from their television airing, but instead must be made up from other sources [1][2]. This is due to a devil's bargain struck by Osamu Tezuka in order to produce the 1963-1966 *Astro Boy* animated television series (arguably the first anime and codified much of the production hacks that make anime possible), where Tezuka (via his studio, Mushi Productions) agreed to sell completed episodes of *Astro Boy* for less than the cost of what they were made [2]. In order to make up this deficit and turn a profit, Tezuka turned to merchandising deals with sponsors such as candy and toy companies, licensing out the titular Astro Boy's likeness in the process [2]. As a result of this, when new anime is created, a production committee to commercially exploit this anime will be formed of the anime's production studio, the original creator (if there is one, and they may be represented by their publisher in their stead), distributors of their work (such as television networks, cinema distributors), and any and all additional sponsors (ex: merchandise companies, record labels, financial investors, advertising agencies) [1]. Members of the production committee, as part of the capitalist system, want to make a profit off of their association with the anime (whether that be sales of anime goods, advertising deals, or physical media of the anime) [1]. This creates a certain limiting factor on what anime gets made: source material (such as manga, light novels, and video games) with large or devoted enough fanbases tend to have an easier time getting made into anime versus less-popular source material or original works, because said fanbases are seen as baked-in audiences/customers for whatever the show's sponsors are trying to sell [1]. In turn, this requires that data (or worse, anecdata) must be acquired in order to arrive at the size of and feasibility of the anime fanbase in question, to make production decisions [1]. As such, a recommendation engine that could find hidden potential fanbases might serve as a catalyst for the production of more and different anime within the existing system, thus providing a source of commercial feasibility.

Second, on the matter of media recommendation engines, there was a sense that between us, the two co-authors of this paper, the existing streaming service recommendation engines we use feel like they repeat the same ~30 titles over and over, and not cover the fullness of our interests in watching stuff versus the hand-crafted recommendations of our friends and family. When one of the co-authors (Brennan) texted his father to wish him a happy father's day, he also informed his father about the *Macross* (the other big Real Robot franchise) meta series's streaming residence of Hulu. This was something that he was unaware of, despite his noted *Robotech* (a re-cut and dub of the first *Macross* series with other, unrelated anime series, famed in science fiction for the dubbers' (Harmony Gold) litigiousness around use of IP contained within by other companies) fandom (he bought the complete series box set approximately a decade ago). Suffice it to say, that these precious experiences are also what motivate the creation of better recommendation engines, on our parts.

Data

The data we used for this task came from one source: a 2023 dump of the myanimelist.net website on Kaggle, a popular (within the Western anime fandom, at least) platform for reviewing, ranking, and listing anime seen by the site's user base (much like Letterboxd or Backloggd, but a bit older) [3]. The dump we used was an unofficial one found on Kaggle [3]. Of the 6 CSV files found within the dump, we used three such files: anime-dataset-2023.csv, final_animedataset.csv, and users-score-2023.csv. The first file, anime-dataset-2023.csv, has ~24,901 entries (each representing an anime) with 24 labels (each representing some facet of the anime on site). The second file, final_animedataset.csv, stems from 2018 and has ~35.3 million entries (each representing a user's rating of an anime in the database) with 13 columns (each representing an important piece of data on said rating). The third file, users-score-2023.csv has ~24.3 million entries (each representing a user's rating of an anime on the site) and 5 labels (essentially a pre-cleaned version of the second file's set). Of that existing dataset, for the first file, we used the "anime_id" (the database id of the anime in question), "synopsis" (the anime's synopsis), and "score" (the anime's overall score) columns of the file for use as our CSV file, clean_items.csv. For the second file, we used the "user_id" (the database id of the rating user), "anime_id" (the database id of the anime in question), and "my_score" (the user's score of the anime) columns of the file for use as part of our ratings matrix. For the third file, we used the "user_id" (same as prior one), "anime_id" (same as prior two), and "rating" (shares the same definition as the prior file's "my_score", but a different name and gets merged with it) columns of the file for use as the other part of our ratings matrix. For our ratings matrix, we randomly split it into three parts on a per-user basis using pandas.df.sample() (seed of 42) with one rating used in the test ratings set, one rating used in the dev ratings set, and the rest in the training ratings set. We then save the test ratings set as ratings_test.csv, the dev ratings set as ratings_dev.csv, and the training ratings set as ratings_train.csv.

Methods

Once we successfully downloaded and cleaned our dataset, we then fed it into our set of 3 models: our baseline model that recommends the top 10 most popular anime, our BERT-based model, and our SVD-based model.

For our baseline model, we built a model that required no training and instead just recommended the top 10 most popular anime in the training dataset to every user in the test dataset, regardless of individual taste in anime (or lack thereof). The model did this by loading ratings_train.csv as the training set and ratings_test.csv as the test set using pandas.read_csv. The model found the top 10 most popular anime in the training dataset by effectively grouping train by anime_id. It then found the sizes of the anime_id groups, and then sorted them by size in descending order, and then served the first 10 entries in the resultant list (using the anime id) to every user in the test set. It then saved the resulting predictions to top_pop_recs.csv using pandas.DataFrame.to_csv() and exited.

For our first actual model, we created a sequence classifier using a BERT variant. The BERT variant we used was "bert-base-uncased" from transformers [4][5], and will be referred to as AL-BERT (AnimeList-BERT) for the rest of this paper. For this model's dataset, we used a random 60% (found using pandasDataFrame.sample() with a seed of 42) of clean_items.csv loaded with pandas.read_csv() and stripped of any and all rows with missing synopses or ratings using pandas.DataFrame.dropna() for removal and pandas.DataFrame.assign() and pandas.to_numeric() to convert all score entries to actual numbers. We then split the dataset into training (80% of the dataset), dev (10% of the dataset), and test (10% of the dataset) sets using pandas.DataFrame.sample() with the same seed as previous occurrences and pandas.DataFrame.drop to remove existing indices used in other sets. For the overall hyper parameters used in this model, there was a max length of 256 tokens, a batch size of 8, 2 training epochs, a learning rate of $2e^{-5}$, RMSE as the metric to select the best models, and AL-BERT as our tokenizer using transformers.AutoTokenizer.from_pretrained(). We further preprocessed our datasets by pre-tokenizing their synopses using AL-BERT and storing them as the "labels" column in their dataframes and removing the anime-id. We built our model using a transformers.AutoModelForSequenceClassification.from_pretrained() with the problem type of regression, AL-BERT as the base model, and number of labels as 1. We then trained our model using transformers.Trainer passing our model, hyper parameters, tokenizer, data collator (transformers.DataCollatorWithPadding() with our tokenizer as the parameter), training set, and dev set as appropriate. We then saved our model in the checkpoints folder as appropriate. We then used the model to predict our non-trading datasets and saved those as bert_item_preds_mid.csb as appropriate with the "anime_id", "score", and "pred_score" columns and exited.

For our second actual model, we tried to predict the top 10 anime for each user using a Singular Value Decomposition model from the Surprise library [6]. To be specific, we used a Singular Value Decomposition++ model. For this model's training data set, we used ratings_train.csv loaded using pandas.read_csv and a smaller version of it containing 3% of the full version's entries, chosen using pandas.DataFrame.sample() with the shared random seed. For this model's test data set, we used ratings_test.csv loaded using the same function as the training set's and a smaller version composed only out of 5000 users' reviews. For this model's hyper parameters, we used 80 factors, 5 training epochs, a learning rate of 0.005, a regularization of 0.02, and the same random seed. We then initialized the model as a surprise.SVDpp with the aforementioned hyper parameters and trained it on the smaller version of the training set. We then saved the model out. We then generated 10 anime recommendations for users in the test and saved it along with doing the same for users in the dev set. We then evaluated the files and then exited.

Results (k = 10)

For our first model (top-pop) we got a precision value of 0.0093, a recall value of 0.0931 and an NDCG score of 0.0458. For our SVD++ test set run, we got a precision value of 0.0011, a recall value of 0.0114, and an NDCG score of 0.0055. For our SVD++ dev set run, we got a precision value of 0.0010, a recall value of 0.0100, and an NDCG of 0.0055. For our AL_BERT run we got an RMSE value of 0.714.

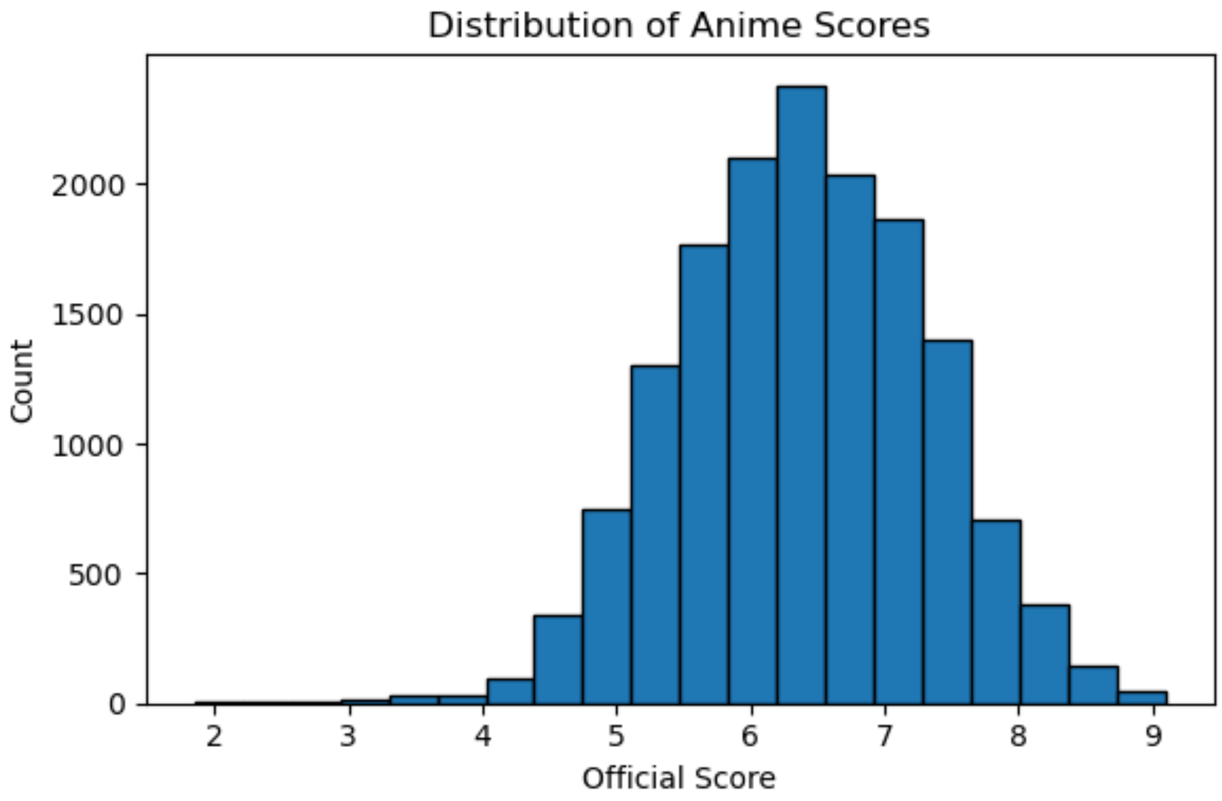
Discussion and Future Work

Our results show us that the SVD++ models do not outperform the baseline model, which suggests that top popularity turns out to be a far better predictor of what people might want to watch than we could have predicted, versus singular value decomposition. More training data (even if it's just increased percentages of existing data set) and longer training time may be required to improve scores and harness latent factors (including increased training epochs). However, our BERT-based model's RMSE performed at par with published MAL baselines, suggesting that the synopsis alone contains enough data to determine actual quality.

Circling back to the discussion of popularity, it may be working better because of 3 factors: users are contributing very little to the sample used to train the SVD++ model, most entries in MAL's database have very few ratings, and that we had no tuning budget to work with. In the SVD++ training sample, each user contributed less than 5 ratings destroying any real chance at collaborative learning. Secondly, it is estimated that 90% of titles in the database have fewer than 200 ratings, thus limiting the odds of it showing up in a test user's history, thereby damaging neighborhood quality. Furthermore, many titles may have low numbers of ratings due to reasons ranging from the entry was only shown for a very limited period of time in a limited location (such as a theme park or side of a mall) and there is no home video release, the entry was targeted at a niche fanbase in Japan, and that the entry was never brought to the West in any capacity (even by fans). Finally, we had no tuning budget to work with because of computational limitations, thus limiting us to a single unadaptable SVD++ configuration that can't deal with user-item density pockets.

We can improve on these models in the future by further tuning hyper parameters in existing models. We can also incorporate our BERT-predicted scores to rank possible SVD++ candidates or as side features in LightFM / LightGBM. Hopefully, this will lead to better predictions in the future.

Appendix: Score Chart for MyAnimeList



Works Cited

[1] J. Sevakis, “The Anime Economy - Part 1: Let’s Make An Anime!,” *Anime News Network*, Mar. 05, 2012. https://www.animenewsnetwork.com/feature/2012-03-05 (accessed Jun. 18, 2025).

[2] M. Steinberg, Anime’s media mix : franchising toys and characters in Japan. Minneapolis: University Of Minnesota Press, 2012.

[3] S. Uddin, “Anime Dataset 2023,” *www.kaggle.com*, 2023. https://www.kaggle.com/datasets/dbdmobile/myanimelist-dataset (accessed Jun. 18, 2025).

[4] T. Wolf et al., “Transformers: State-of-the-Art Natural Language Processing,” Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pp. 38–45, Oct. 2020, doi: https://doi.org/10.18653/v1/2020.emnlp-demos.6.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” *Proceedings of the 2019 Conference of the North*, vol. 1, pp. 4171–4186, Jun. 2019, doi: https://doi.org/10.18653/v1/n19-1423.

[6] N. Hug, “Surprise: A Python library for recommender systems,” *Journal of Open Source Software*, vol. 5, no. 52, p. 2174, Aug. 2020, doi: https://doi.org/10.21105/joss.02174.