Recommender Algorithm for Japanese Animes

BT5153 Applied Machine Learning in Business Analytics – Team Anime-niacs Project Report

Sam Too Shang Yao (A0206528E), Quek Khim Geok (A0206471L), Lim Yew Kuan (A0206517J),
Tan Wei Jie (A0036333N), Ratna Herlina (A0206506M)

Abstract

Recommender systems have been widely adopted and successfully used by e-commerce websites and content streaming websites to improve sales, enhance customer experience, increase engagement time. In this work, we explored using content-based filtering and collaborative filtering to develop a recommender system for Animes, which are increasingly gaining popularity globally. Under the content-based filtering, we derived the cosine similarity score between each pair of animes using either the synopsis of the animes or the images of the animes to determine which animes are similar and hence, to be recommended to users. Under collaborative filtering, we explored techniques such as singular vector decomposition based on matrix factorization and an autoencoder to predict the ratings of animes which users did not interact before. Our results showed that the collaborative filtering approach performed better, with the autoencoder model achieving the highest hit rate of 43% when evaluated using the top 10 animes to be recommended to users.

1. Introduction

Anime refers to animation in Japan and was originally produced for consumption among the Japanese. It has become extremely popular, has been translated into many different languages, and gained a huge following from millions of international fans. Unlike conventional cartoons from American/European producers (i.e., Walt Disney) which are predominantly catered to children, anime is highly popular amongst adults as it delves into diverse topics and themes that are relatable to a mature audience. However, due to the diversity of genres, ratings of animes could be highly subjective. For instance, animes that might be highly rated by seasoned fans might not be perceived favourably by new viewers as they require background knowledge of other animes or Japanese culture. In addition, with more than seventeen thousand anime titles to choose from, viewers could potentially get lost and waste hours scrolling through and watching animes that do not appeal to them.

2. Problem Statement

Recommender systems have been widely adopted and successfully used by e-commerce websites such as Amazon (Amazon, 2019) and content streaming websites like Netflix (Netflix, 2020) to improve sales, enhance customer experience, increase viewership and engagement time. However, there is currently a lack of interest in the area of enhancing user experience for viewers of animes. In addition, popular anime websites such as MyAnimelist (MyAnimeList, 2021) (MAL) and Anime-Planet (Anime-Planet, 2021) have recommender systems embedded in them. However, one common trait observed in their recommender system is that it relies on members who have watched the animes to make recommendations. For example, in MAL, the Anime Recommendations tab provides data on the pairing of the recommendations made by a user and the reason for the recommended pairing and when it was made (MyAnimeList, 2021). This is similarly observed in Anime-Planet (Anime-Planet, 2021). Given the advancement in techniques used to develop recommender systems, there are more avenues to tap on unstructured data such as text and image data which could potentially improve the quality of the recommendation and hence increase user engagement.

As such, the prime motivation of this project is to develop a recommender system with an algorithm which incorporates unstructured data (i.e., synopsis and anime cover images) with the assumption that viewers tend to watch animes with similar artwork styles or storylines.

3. Dataset

Three key datasets were identified for this study: (i) Reviews data; (ii) Anime Metadata; and (iii) Images of Animes. Details of each dataset are provided in the following sub-sections.

3.1 Reviews

Review data were extracted from Kaggle MyAnimeList Dataset (MyAnimeList Dataset, 2020). The creator of the dataset scrapped the data from MAL (MyAnimeList, 2021). The data was kept in 3 separate tables namely: (i) Profile Dataset – this table contains information about users who watched the various animes. It comprises 81,727 observations with 5 variables; (ii) Reviews Dataset – this table contains the ratings given by users for an anime. It comprises 192,112 observations with 7 variables; and (iii) Animes Dataset – this table contains metadata of each

anime. It comprises 19,311 observations with 12 variables. Figure 1 details the data variables and the corresponding data type found in each of the abovementioned dataset.

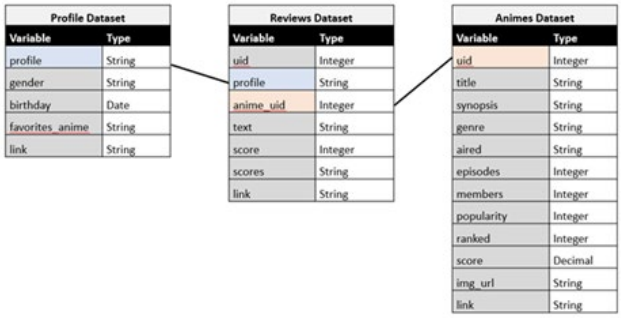


Figure 1 – Data Variables and Data Types for Profile, Reviews and Animes Datasets

3.2 Anime Metadata

Using a combination of Selenium and Beautiful Soup packages to scrape the MAL website (MyAnimeList, 2021), a total of 39 metadata fields were obtained. Table 1 below details the metadata fields and Table 2 shows the count of animes belonging to the top 10 genres.

Table 1. Anime Metadata Scrapped from MAL Website

Variable	Type	Variable	Type
Title	String	Popularity	String
URL	String	Members	Integer
English	String	Favorites	Integer
Synonyms	String	Started	Date
Japanese	String	Ended	Date
Type	String	Voters	Integer
Episodes	Integer	Adaptation	String
Status	String	Alternative version	String
Aired	String	Side story	String
Premiered	String	Spin-off	String
Broadcast	String	Synopsis	String
Producers	String	Prequel	String
Licensors	String	Alternative setting	String
Studios	String	Sequel	String
Source	String	Other	String
Genres	String	Summary	String
Duration	Integer	Character	String
Rating	String	Parent story	String
Score	Decimal	Full story	String
Ranked	String		

Table 2. Anime Metadata Scrapped from MAL Website

Genre	No. of Animes
Comedy	6,009
Action	3,865
Fantasy	3,258
Adventure	2,950
Kids	2,662
Drama	2,616,
Sci-Fi	2,575
Music	2,230
Shounen	1,995
Slice of Life	1,898

3.3 Images

Image data of the corresponding anime titles were scraped from MAL website (MyAnimeList, 2021) by utilising the URL information extracted in Section 3.2. Beautiful Soup package in Python was used to fetch a total of 17,335 images in jpg format. The sizes of the images are typically between 200 and 250 pixels wide and between 300 to 400 pixels long. Both vertical and horizontal resolutions are at 96 DPI. These images usually represent the cover pages/poster view of the anime. Figure 2 below illustrates some of the images extracted.

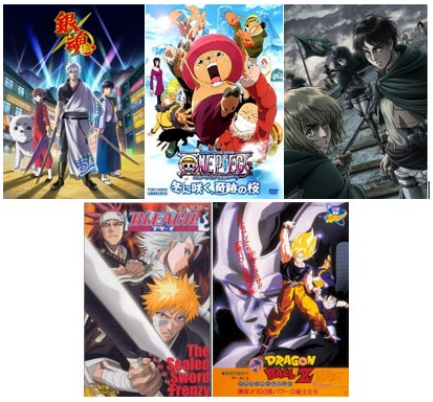


Figure 2 – Sample of Image Data obtained from MAL website

4. Data Pre-processing

4.1 Removal of duplicates

A total of 61,593 duplicate uid, which represents the rating made by a user for an anime, were removed from the

Reviews Dataset, while a total of 11 duplicate animes were removed from the Animes Dataset.

4.2 Scope of analysis

A subset of the Anime Dataset was obtained where only the animes which were shown on television were considered for analysis. Animes with other types, such as "Movies" and "Music" were excluded. In addition, only animes which had valid synopsis data (i.e., non-null fields) were selected. This seeks to ensure that the animes used in the content-based filtering analysis are consistent. Apart from limiting the user rating data from the Reviews Dataset based on the animes which were within scope, users who had only rated one anime were excluded. The exclusion seeks to ensure that the performance of the recommender algorithm on every user can be evaluated in the test dataset.

5. Machine Learning Models

The techniques used to build a recommender system could be classified into two broad categories: (i) Content-based Filtering – a method which makes recommendations by identifying the common characteristics of items that have been well received by a user and recommending new items which share similar characteristics to the user (Ricci et al., 2011). An example of characteristic can be the genre class in the context of anime; (ii) Collaborative Filtering – a method which makes recommendation through gathering similar users' historical preference on a set of items. The main idea is that the rating of a user is likely to be similar to another user if both of them have rated other items in a similar manner (Ricci et al., 2011).

A total of 5 models were experimented in this project. Models 1, 2a and 2b were content-based filtering models which were trained using features either derived from the synopsis of animes or the images to measure the similarity of animes. Models 3 and 4 were collaborative filtering models which were trained using user ratings.

5.1 Model 1: Content Based Filtering with Synopsis Similarity

In model 1, text preprocessing was done on the synopsis data to remove noise and retain key words as features before the computation of cosine similarity scores for the animes. The text preprocessing steps consisted of adding titles to the synopsis if it only contains phrases such as "second season", converting text to lower case and removing punctuation; removing digits and whitespace; removing English stopwords; removing words with 2 characters or less; and lemmatization of the words. With the processed synopsis, features were generated using Term Frequency Inverse Document Frequency ('Tfidf') vectorizer (Jain, 2020), with n_gram ranging from 1 to 5 to capture context and semantics. With the vectorized text data, the cosine similarity scores were calculated for every pair of animes using sklearn's metrics.pairwise.linear_kernel function. Thereafter, the predicted ratings of user *u* for anime *i* was obtained by taking the sum of the product of similarity scores and user u rating over the sum of all similarity scores using the formula:

$$\hat{p}_{u,i} = \frac{\sum_{m \in I}(r_{u,m} * s_{i,m})}{\sum_{m \in I} s_{i,m}}$$

where

$m \in I$ represents an anime in the set of all animes within scope
$r_{u,m}$ represents the user ratings for anime *m*
$s_{i,m}$ represents the similarity score between the anime *m* rated by the user and anime *i* of interest
$\hat{p}_{u,i}$ represents the predicted rating for user *u* for anime *i*

5.2 Model 2a: Content Based Filtering with Image Similarity (Transfer learning only)

Keras provides access to several top-performing pre-trained models such as ResNet-50, VGG19 and InceptionV3. Given the availability of these top-performing pre-trained models and in view that our image dataset is relatively small, we applied CNN transfer learning to extract the latent features from the images of each anime. Specifically, model weights from a pretrained ResNet-50 model, trained using images from ImageNet, were used. The fully connected output layer was excluded by specifying the "include top" argument to "False" since the model was intended as a feature extractor rather than a classifier. Additionally, a global average pooling layer was added to summarize the activation for their use as a feature vector representation of the input images. As ResNet-50 requires the image inputs to be of target size 224 x 224, the cover images of the animes were resized to this dimension. The application of transfer learning thus converts an image from an input of 224 x 224 x 3 dimensions to an output of 2048 dimensions.

Once the latent features of the anime cover images were extracted, the cosine similarity scores between the images were calculated using the sklearn.metrics.pairwise cosine similarity function. Figure 3 shows an example of an image with its top-3 most similar images.

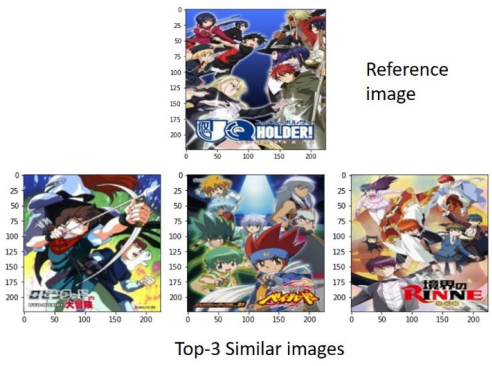


Figure 3 - Example of an image with its top-3 similar images

5.3 Model 2b: Content Based Filtering with Image Similarity (Transfer learning and classification)

Model 2b also used image latent features for its similarity matrix. However, instead of just extracting the features directly from the global average pooling layer via transfer learning from a pre-trained ResNet-50 model, fine-tuning was applied. A dense layer with a 'relu' activation function was added to compress the dimensions even further to 1024 dimensions, and a softmax layer was added to model it as a multilabel classification task. During training, the backbone ResNet50 model weights were frozen. Only the dense and softmax layers' weights were updated. After training the classification model, the output of the dense layer (i.e., 1024 dimensions) was used to compute the cosine similarity scores of the animes. The objective of such modification seeks to ensure that the features to be extracted from the images are conditioned on the genres of the animes, thereby minimizing the occurrence where two animes with drastically differing genres are given high similarity scores solely due to the similarity in the design of the images (Wrg, 2020). Figure 4 shows an example of an image with its top-3 most similar images. One observation is that the ordering is slightly different from Figure 3 because of the fine-tuning done on the additional layer based on the classification task.

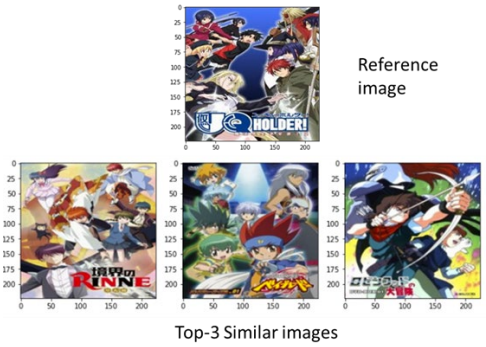


Figure 4 – Example of an image with its top-3 similar images

For the classification task, instead of using the 42 genres extracted from MAL as the target values, a total of 5 topics derived from the synopsis were used as the target classes. These topics were obtained through Latent Dirichlet Allocation (LDA) on the synopsis of the animes. Before LDA was conducted, the following text preprocessing steps were taken: conversion to lower case; removal of whitespace; removal of non-alphanumeric characters; conversion of blank synopsis to empty string; removal of english stopwords; lemmatization where bigrams were created and nouns, adjectives, verbs and adverbs were retained. Subsequently, the Mallet library was used for Topic Modelling and it was preferred over the Gensim library's LDA function as it had been shown to produce better quality topics (McCallum, 2002). The Mallet model provided Coherence Values which were a measure of the degree of semantic similarity between high scoring words n the topics. The selection of optimal number of topics was based on the elbow-method where the number of topics with highest Coherence Values before flattening out was chosen. As shown in Table 3, the optimal number of topics appears to be 5. From the Mallet model, the 5 topics inferred from the keywords that were linked to the image features include: Animes about war or battles on earth; Animes about membership in a club or group; Animes about love story set in school; Animes featuring songs, films or videos; and Animes about family life.

Table 3: Number of Topics and Coherence Values

No of Topics	Coherence Value
2	0.428
3	0.536
4	0.581
5	0.587
6	0.600
7	0.608
8	0.600

5.4 Model 3: Collaborative Filtering with Singular Value Decomposition for User Interactions

In Model 3, Singular Vector Decomposition (SVD) based on Matrix Factorization (MF) is applied to better address the sparsity issue observed in the reviews data. Under this approach, MF learns the latent preferences of users and the latent attributes of items from known ratings (learn features that describe the characteristics of ratings) which could then be used to predict the unknown ratings through the dot product of the latent features of users and items (Cambridge Spark, 20202).

To apply SVD modelling, the Surprise library was used. As the Surprise library (Hug, 2015) also offered other prediction algorithms, to examine if SVD is the best prediction algorithm as measured by the lowest RMSE attained, an initial test was run using SVD and 6 other prediction algorithms (i.e. SVDpp, KNN Baseline, KNN with Means, KNN Basic, SlopeOne and CoClustering) with default parameters and 3-fold cross validation. The initial test revealed that the SVD prediction algorithm achieved the lowest RMSE as shown in Table 4.

Table 4: RMSE of the 7 Prediction Algorithms Tested

Prediction Algorithm	RMSE
SVD	1.872
SVDpp	1.898
KNN Baseline	1.996
KNN Basic	2.182
KNN with Means	2.174
SlopeOne	2.224

CoClustering	2.096

With the SVD model identified as the best prediction algorithm, a Grid Search with 5-fold cross validation was conducted to further identify the best hyperparameters with RMSE and MAE being the evaluation metrics. The best hyperparameters found which yielded a RMSE of 1.83 and MAE of 1.40 are:

i. 'n_factors': 5

ii. 'n_epochs': 5

iii. 'lr_all': 0.02

iv. 'reg_all': 0.05

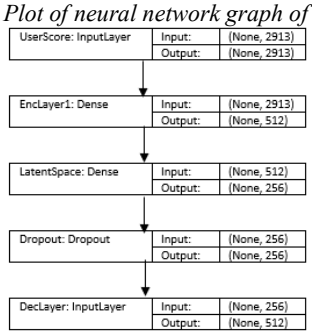
5.5 Model 4: Collaborative Filtering with Autoencoder for User Interaction

In model 4, an autoencoder model was explored (Rosebrock, 2020). To apply this model, a user-item sparse matrix first had to be created using both the user rated animes as well as animes that were not rated by users. For animes which were not rated by users, they were assigned a value of 0 in the sparse matrix. However, this introduced an issue as the minimum rating was 0. Thus, to distinguish between animes which have not been rated vis-a-vis animes which were poorly rated, we adjusted the minimum rating from 0 to a value of 0.1.

The autoencoder model (shown in Figure 5) was built with the following components:

i. The 'selu' activation function was applied to the encoder layer, latent space, and decoder layer. The 'selu' activation function was used to achieve output that follows a normal distribution, which would reduce the chances of vanishing or exploding gradient problem.
ii. The "linear" activation was chosen for the output layer. "Linear" activation was selected as the output is a continuous score.
iii. Optimisation of the model was done by using the Adam optimizer with learning rate of 0.00001. Adam is fast and has relatively low memory requirements.
iv. Mean Squared Error (MSE) was chosen as the loss function as this is not a classification problem where binary cross-entropy can be used.

Figure 5 – Plot of neural network graph of Model 4



A trial-and-error approach was used to perform hyperparameters tuning to determine the parameters that gave the lowest validation loss. The number of epochs was set at 30 with batch size being 64. Deeper models were considered by adding more dense layers to both the encoder and decoder to check if the results could be improved. It was observed that building a deeper model did not improve the validation loss. Table 5 shows the validation loss for the different sets of hyperparameters tested. The best fit autoencoder model was subsequently used to generate a new set of matrix of user-item interactions where predictions were made for unknown ratings.

Table 5: Validation loss for different sets of hyperparameters

EncLayer1: Dense	LatentSpace: Dense	DecLayer1: Dense	Validation Loss
512	256	512	0.0262
256	128	256	0.0272
128	64	128	0.0279
64	32	64	0.0283
32	16	32	0.0285
512	2	512	0.0287

6. Results

6.1 Leave-one-out method for train-test split

To evaluate the performance of each recommender model, the Reviews Dataset was split into train dataset and test dataset where the train dataset was used for model training while the test dataset was used only for evaluation of models. Unlike other machine learning applications, a random train-test split strategy cannot be applied as it would result in the train dataset having access to recent reviews made by users while the test dataset contains their older reviews. This would result in data leakage and the model will incur a look-ahead bias, thereby unable to generalise well to unseen data. To avoid the above issue, the leave-one-out methodology was used across all 5 models during the train-test split where for each user, the most recent review was used as the test set. The most recent review for each user was indicated by the largest uid.

6.2 Evaluation of models using Hit Rate @ 10

To provide recommendations for each user, instead of making predictions for all animes which have not been rated by the users which is computationally intensive, a simulation was carried out on a smaller set of 100 animes. Specifically, for each user, the set of 100 animes was derived by randomly selecting 99 animes from the list of animes which the users had not rated before and combining it with the most recent anime that the user had rated as captured in the test dataset. Using each of the 5 models,

rating predictions for each of these 100 animes for each user were derived.

To compare the performance of the recommendations made across the models, Hit Rate @ 10 evaluation metric was applied (He et al., 2017). Under this evaluation criteria, for each model, the top 10 animes with the highest predicted ratings were extracted for each user. The top 10 animes extracted were matched against the test data to examine if the most recent rated anime for every user was captured in the top 10. A hit score *(h)* of "1" was generated if a match occurs for each user, otherwise "0". Thereafter, the hit rate of each model was derived by taking the sum of the hit scores across all users divided by the total number of users, as shown below:

$$\boldsymbol{Hit\ Rate_x} = \frac{\sum_{u \in U} \boldsymbol{h_{u,x}}}{\boldsymbol{Total\ number\ of\ users}}$$

where

$\boldsymbol{x}$ refers to one of the 5 models

$\boldsymbol{u}$ refers to a user

$\boldsymbol{U}$ refers to all users

$\boldsymbol{h_{u,x}}$ refers to the hit score for user u in model x

To evaluate robustness of each model, the above process of deriving a list of 100 animes for each user (by randomly sampling 99 animes and combining it with the most recent anime rated in the test dataset) and calculating the hit score of each user was simulated 5 times. The overall hit rate for each simulation was also derived. Given the 5 simulations, an average hit rate was derived for each model as shown below:

$$\boldsymbol{Average\ Hit\ Rate_x} = \frac{\sum_{j=1}^{5} \sum_{x=1}^{5} \boldsymbol{H_{j,x}}}{\boldsymbol{J}}$$

where

$\boldsymbol{x}$ refers to one of the 5 models

$\boldsymbol{j}$ refers to a simulation and ranges from 1 to 5

$\boldsymbol{H_{j,x}}$ refers to the hit score for simulation j and model x

$\boldsymbol{J}$ refers to the total number of simulations

Table 6 shows the hit rates across the 5 simulations as well as the average overall hit rate for each model. A baseline model 0 was added as a benchmark. The baseline model was derived by applying a random selection of 10 animes from the list of 100 animes to be recommended to users. As every anime in the list of 100 animes has a 10% chance to be selected in the top 10 animes to be recommended, the average overall hit rate for the most recent anime to be selected for each user is 10%. Based on 5 runs of simulation, the hit rate of the baseline model 0 was 9.61%. It was observed that all of the recommender models outperform the baseline model, with the Autoencoder model for user interaction (Model 4) yielding the highest overall average hit rate of 42.97%, meaning that for 42.97% of the users, their most recent rated anime appeared as one of the recommended top 10 animes. It was also interesting to note that the average overall hit rate for the collaborative filtering models (i.e., models that were trained on the user ratings) were higher as compared to the content-based filtering models (i.e., models trained on the synopsis data or the images).

Table 6: Results from Models

Model	**Average Overall Hit Rate @ 10**
0: Baseline for comparison	9.61%
1: Content Based Filtering with Synopsis Similarity	20.87%
2a: Content Based Filtering with Image Similarity	19.52%
2b: Content Based Filtering with Image Similarity (Transfer learning and classification)	18.92%
3: Collaborative Filtering: SVD based on User Ratings	29.13%
4: Collaborative Filtering: Autoencoders for User Interaction	42.97%

Figures 6 to 10 show the hit rate for each simulation of each model. Across all models, the results are robust. In Model 1 (content-based filtering with synopsis similarity), the hit rate ranges from 20.1% to 21.7%, with a standard deviation of 0.006. In Model 2a (content-based filtering with images similarity using transfer learning), the hit rate ranges from 18.4% to 20.9%, with a standard deviation of 0.009. In Model 2b (content-based filtering with images similarity using transfer learning and classification task), the hit rate ranges from 17.9% to 19.5%, with a standard deviation of 0.005. In Model 3 (collaborative filtering with SVD for user interactions), the hit rate ranges from 25.2% to 30.4%, with a standard deviation of 0.03. In Model 4 (collaborative filtering with autoencoder for user interactions), the hit rate ranges from 41.9% to 45.2%, with a standard deviation of 0.01. While a higher standard deviation was observed in

models 3 and 4 compared to models 1 and 2, the hit rate @ 10 across all simulations in models 3 and 4 were consistently higher than that of models 1 and 2. Overall, the results are robust and suggest that collaborative filtering models outperformed content-based filtering models.

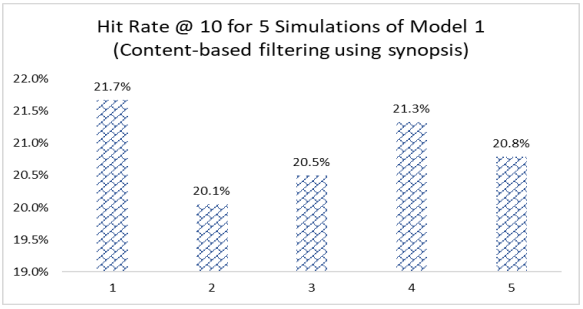


Figure 6 – Hit rate @ 10 for all simulations of Model 1

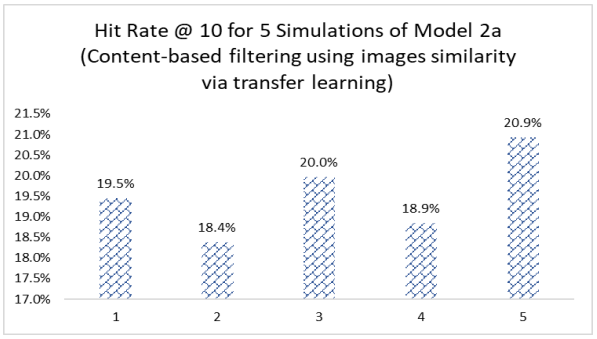


Figure 7 – Hit rate @ 10 for all simulations of Model 2a

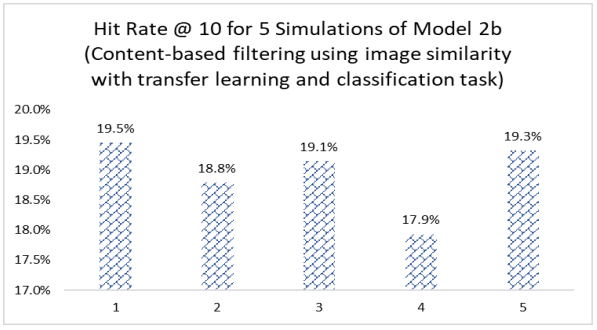


Figure 8 - Hit rate @ 10 for all simulations of Model 2b

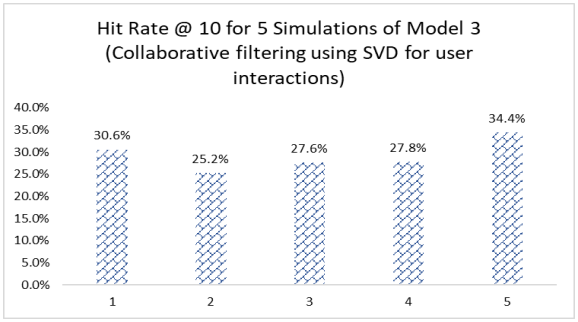


Figure 9 – Hit rate @ 10 for all simulations of Model 3

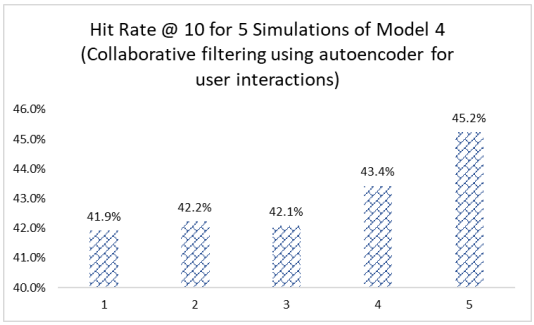


Figure 10 – Hit rate @ 10 for all simulations of Model 4

7. Insights gained while applying machine learning models

7.1 Feature generation: Pretrained model weights might lead to better performance when training data is lacking

For the content-based filtering with image similarity, an initial model utilised the autoencoder network structure as well as image augmentation to obtain latent features at the bottleneck layer. However, in terms of image similarity, the pre-trained ResNet50 model (Model 2a) led to better model performance as compared to using autoencoders for feature generation. The reason for this was because ResNet50 was trained with more than 1 million images from ImageNet whereas there were only around 17000 anime images for training with the autoencoder architecture. Therefore, Model 2a could extract better high-level features even though it did not use anime images for training.

7.2 Collaborative Filtering Models vs Content Based Filtering Models

As highlighted in the results, the collaborative filtering models performed better as compared to the content-based filtering models. One possible reason is because collaborative models utilize information based on other users to predict if a particular user would like the anime, while content-based models only make use of an individual's ratings and similarity between the animes based on the synopsis or images. As such, the content-based models are limited by the number of reviews that each user has provided.

In addition, under content-based filtering, it was observed that the cosine similarity score between each anime and its most similar anime derived using synopsis data was generally low, with the cosine similarity score at the 75th percentile being 0.068. Figure 11 shows that the distribution of the cosine similarity score is right skewed. The poor similarity scores could be a factor driving the lower average overall hit rate for model 1.

On the other hand, while the cosine similarity score derived using images are generally higher (Figure 12), the average

overall hit rate remains low. This suggests that the visual appeal of animes play a lesser role in users' decision of whether to watch an anime.

Beyond the higher hit rates, collaborative filtering approaches were found to be more efficient compared to content-based filtering approaches.

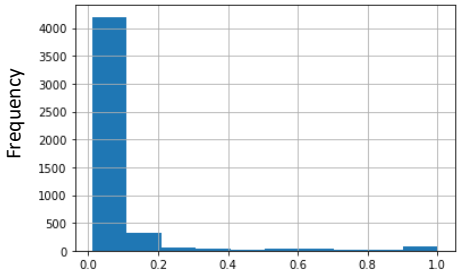


Figure 11 – Distribution of the Cosine Similarity Score of anime pairings derived using synopsis

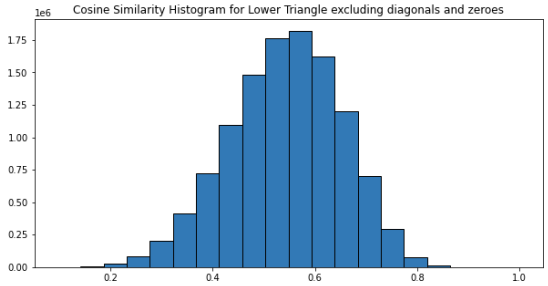


Figure 12 – Distribution of the Cosine Similarity Score of every anime pairings derived using images

7.3 Collaborative Filtering Models: Autoencoders performed better than SVD

SVD models are linear models which are unable to capture complex nonlinear and intricate relationships that can be predictive of users' preferences. An autoencoder is a neural network that learns to copy its input to its output to encode the inputs into a hidden (and usually low-dimensional) representation and it is proven to be capable of approximating any continuous function, making it suitable for addressing the limitation of matrix factorization and enhancing the expressiveness of matrix factorization. It is also widely used for its outstanding performance in data dimensionality reduction, noise cleaning, feature extraction, and data reconstruction. Thus, the autoencoder was able to perform better than SVD possibly due to its ability to learn the non-linear user-item relationship efficiently and to encode complex abstractions into data representations (Ferreira et al, 2020).

7.4 Future Directions

Although the average overall scores remained low across all models (i.e., below 50% hit rate), the performance still exceeded the baseline model of random recommendation. This suggests that there is still value in each of the recommender algorithms developed. The lower than 50% hit rate is likely attributed to a relatively small dataset (75,000 reviews). With a larger dataset, the predictive power of the model is likely to improve.

Between the content-based filtering and collaborative filtering approach, although the models that utilized content-based filtering did not achieve the highest overall hit score, these models could probably be used to augment the autoencoder model for user ratings such that a hybrid recommender system could be implemented. This would allow a greater variety of data types to be used as inputs for the recommender system which in turn, would make it more robust. Specifically, a hybrid recommender system could prove to be useful in scenarios where users are not inclined to provide ratings, or where users' engagement with the platform lean towards reading of synopsis or browsing via images.

8. Conclusion

This project set out to develop a recommender system using collaborative and content-based filtering methods that incorporate unstructured data (i.e., synopsis and anime cover images) as well as structured data (i.e., user-item ratings). A total of five models were used to generate recommendations to users and it was found that the Autoencoder model for user interaction achieved the highest hit rate across the five models. In terms of the application of machine learning, several insights were also gained where: (i) Pretrained model weights could lead to better performance in feature generation if training data is lacking; (ii) Collaborative filtering models might perform better as content based models are limited by the amount of reviews available; (iii) Autoencoders could perform better than SVD as it is able to learn from non-linear user-item relationships and could encode complex abstractions into data representations.

The code implementation of this report can be found on this link:
https://github.com/WJIE08/Recommender_Algorithm_Japanese_Anime

References

Anime-Planet (2021) https://www.anime-planet.com/anime/recommendations/

Amazon. (2019). *The history of Amazon's recommendation algorithm*. https://www.amazon.science/the-history-of-amazons-recommendation-algorithm

Cambridge Spark. (2020). *Implementing your own recommender Systems in Python*.

https://blog.cambridgespark.com/nowadays-recommender-systems-are-used-to-personalize-your-experience-on-the-web-telling-you-what-120f39b89c3c

Ferreira, D., Sofia, S., Antonio, A., & Machaco, J. (2020). Recommendation System Using Autoencoders. *Applied Sciences, 10*, 5510.

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T-S. (2017). Neural Collaborative Filtering. In *Proceedings of WWW '17, Perth, Australia, April 03-07, 2017*.

Hug, N. (2015). *Surprise Documentation for Model Selection Package*. https://surprise.readthedocs.io/en/stable/model_selection.html

Jain, S. (2020). *Ultimate guide to deal with Text Data (using Python) – for Data Scientists and Engineers*. https://www.analyticsvidhya.com/blog/2018/02/the-different-methods-deal-text-data-predictive-python/

McCallum, A. (2002). *Mallet Documentation for Topic Modeling*. http://mallet.cs.umass.edu/topics.php

MyAnimeList (2021) https://myanimelist.net/

MyAnimeList Dataset. (2020). https://www.kaggle.com/marlesson/myanimelist-dataset-animes-profiles-reviews

Netflix. (2020) *Netflix Research.* https://research.netflix.com/research-area/recommendations

Ricci, F., Rokach, L., Shapira, B., & Kantor, P. B. editors. *Recommender Systems Handbook.* Springer, 2011

Rosebrock, A. (2020). *Autoencoders for Content-based Image Retrieval with Keras and TensoryFlow*. https://www.pyimagesearch.com/2020/03/30/autoencoders-for-content-based-image-retrieval-with-keras-and-tensorflow/

Wrg, A. (2020*). Image recommendation engine with Keras*. https://towardsdatascience.com/image-recommendation-engine-with-keras-d227b0996667