Data Mining Project

Mick Neupart
Email: mine@itu.dk

Paul Mihai Stolniceanu
Email: pmis@itu.dk

Yumer Adem Yumer
Email: yuyu@itu.dk

***Abstract*—The aim of this project is to investigate the differences in performance of content-based and collaborative filtering recommendation systems, in the context of animes watched by users on the myanimelist.net portal. The findings from the experiments run on a subsample of the presented dataset, with the two different implementations, suggest that collaborative filtering yields better results in this setup. The generated user profiles seem to better summarize the users' behavior in relation to other similar users, rather then being a direct indicator of the users' preferences towards the content.**

Keywords*—Recommendation systems, anime, collaborative filtering

I. Introduction and Motivation

User behavior data are a valuable resource for any company as they can be a reliable predictor for the next actions the users are about to take. Therefore, recommendation systems are used with the aim of getting the users to consume more content as they are now presented with more relevant and enjoyable content. These recommendation systems can roughly be grouped into two families: content-based and collaborative filtering[1]. The first one relies on the features of the content previously consumed by the user, while the second one relies on the similarity relationships between the users when trying to identify items to recommend.

For this project, an implementations of the two approaches will be compared on a dataset of users' rated animes. The motivation for this is to explore the inner workings of recommendation systems as their use is widespread and applicable to many different topics.

II. Data overview

The dataset used comes from Kaggle, an online community for data mining and machine learning. The data in the set is mined from the myanimelist.net and consists of two sets. One containing a list of animes and the other a collection of user data.

A. Data structure

The two datasets used are *ratings.csv* (user data) and *anime.csv* (anime data). The rating file has three columns as listed below.

- **user_id**, id for the user
- **anime_id**, id for anime, which can be found in the anime file
- **rating**, the user's rating form 1-10 given to the anime, if no rating is given -1 will be listed

The anime file has the following attributes

- **anime_id**, id for the anime
- **name**, name of the anime
- **genre**, a list of genres for the anime
- **type**, the kind of anime, TV, series, music video, etc.
- **episodes**, the number of episodes in the anime. Movie types has a episode count of 1
- **rating**, the average rating
- **members**, the number of community members on myanimelist.net

A sample of the *ratings.csv* file as well as the file structure can be seen in Table I. The same can be observed for the *anime.csv* in Table II.

TABLE I. Sample of the ratings.csv file

user_id	anime_id	rating
2	11771	10
2	12189	-1
2	16417	-1

TABLE II. Sample of the anime.csv file

anime_id	name	genre	type	episodes	rating	members
32281	Kimi	Drama, School	Movie	1	9.37	200630
25835	Shirobako	Comedy, Drama	TV	24	8.49	146895

B. Exploratory data analysis

This section offers an overview of the diversity and distribution of the data with regards to which cleaning methods to utilise. It is used to improve our knowledge to better reason about the feature selection for the recommendation systems.

Listed below is an overview of the descriptive statistics for rating and members attribute in the anime file consisting of $12,294$ different animes and also an overview of the anime watched per user aggregated from the ratings file with a total of $73,515$ unique users.

TABLE III. Descriptive statistics

attribute	mean	median	mode	std	min	max
rating	6.47	6.57	6.00	1.03	1.67	10.0
members	18,071.34	1,550.00	nan	54,820.68	5.00	1,013,917.00
anime pr user	106.29	57.00	1	153.09	1	10,227

TABLE IV. Descriptive statistics - quartiles

attribute	1 qrt	2 qrt	3 qrt
rating	5.88	6.57	7.18
members	225.00	1,550.00	9,437.00
anime pr user	18.0	57.0	136.0

Looking at the rating attribute it can be inferred that it closely resembles a normal distribution given close mean, median and mode scores. The member attribute however shows that our distribution is very much skewed to the right since the mean is very large compared to the median. Also the min and max is so far from each and further investigation into the distribution could be interesting. The last attribute anime per user paints a similar picture as the members meaning that the distribution is skewed to the right, as a large amount of users seem to have seen very few animes. Another interesting aspect is that the mode is only 1.0 which could tell us that these users only signed up for the page once to register an anime and then left or that they are fairly new users to the platform. The fact that the maximum number of animes watched by a user is 10,227 is a strong indicator of an outlier, as that implies the user has seen more than 80 % of the entire animes' catalog.

1) Number of animes / rating: As expected, the rating seems to follow a normal distribution as seen in figure 1. If we take the quartiles into account for the ratings, seen in table reftbl:quartile, we can infer that when users rate an anime with 5 or less it is below average since it is lower than the first quartile. The same can be said for user ratings of 8 or higher. Those represent "good" animes, since the rating is larger than the third quartile.

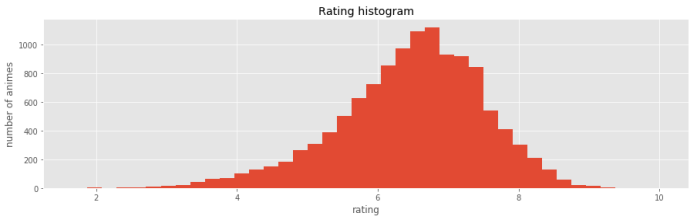

Fig. 1. Distribution of the rating attribute

2) Number of users / number of watched animes: From figure 2 we are able to see how many animes the users have watched. The reason for the long tail is the far outlier with $10,227$ animes seen. Calculating the 99 quantile we get ≈ 707 meaning that 1% of the population is creating this tail. We will have to consider ways to deal with these outliers once we start the data preprocessing section (III-E).

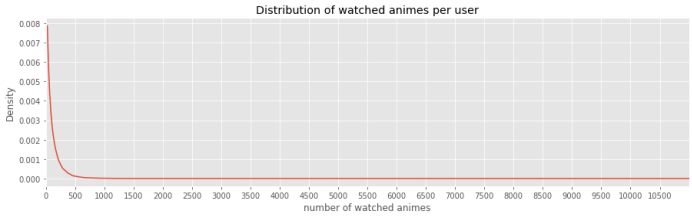

Fig. 2. Density of users and how many animes they have watched

3) Number of animes / number of genres: As mentioned before, each anime has a number of genre attached to it, describing its content. In figure 3 we see the most common genre for animes and their distribution (i.e. how many animes are tagged with that genre). The figure clearly shows that some genres are more common than others. Animes with *Comedy* as a genre makes up approximately $1/3$ of the entire catalog making it the most prominent genre. There are 43 genres in total.

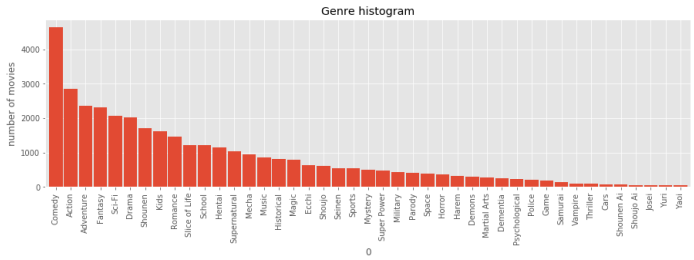

Fig. 3. Most common genres

4) Genres distribution / number of animes: Only knowing that there is a total of 43 genres paints an incomplete picture, since genres can be mixed. Figure 4 shows us the distribution of the genre count for animes. Interestingly animes can have up to 13 genres, but more importantly, approximately 80% of animes have more than one genre. This means that animes can be more specialised when trying to look for similar animes.

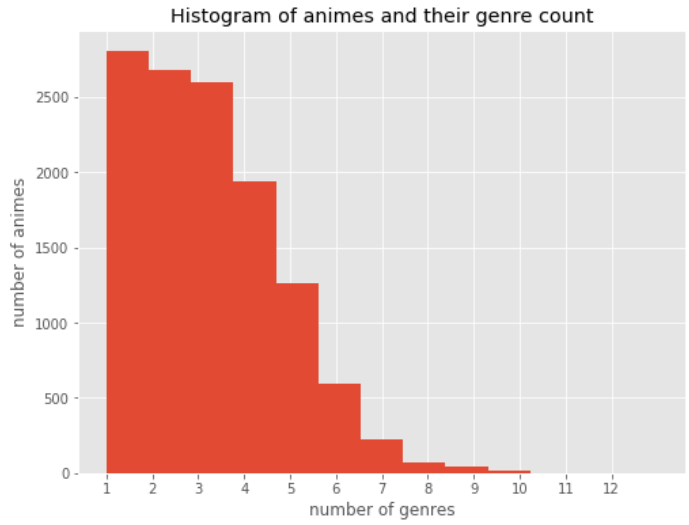

Fig. 4. Distribution of the genre count

5) Member count distribution: The data set also contains a member count or a popularity measure of a particular anime. The larger the community fan base the more popular it is. Only looking at table III and IV it is clear that the diversity in the member count is very large and only a handful of them are of any meaningful size. Looking at figure 5 and 6 the same pattern can be observed. Therefore, using the member count attribute as a feature for the recommendation system should be properly weighted, not to bias the recommendations towards suggesting popular shows, but rather highly rated animes tailored to the user.

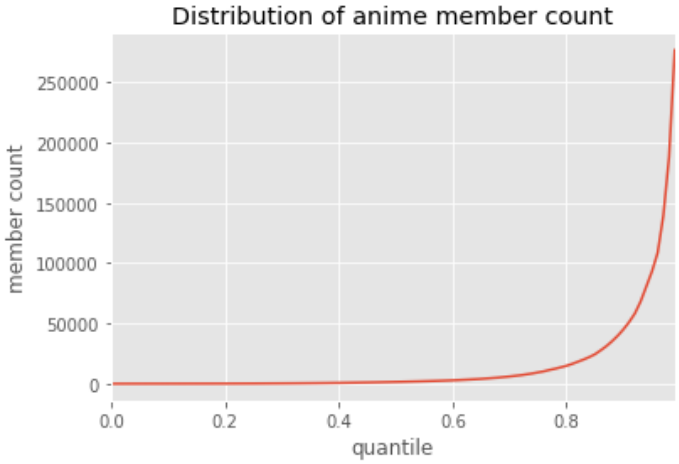

Fig. 5. Member count at the different quantiles

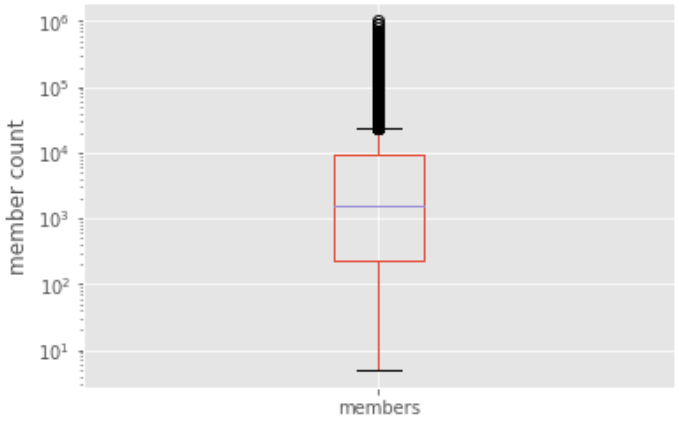

Fig. 6. Box plot of the member count

III. Methods

When creating recommendation systems, two general approaches are often applied, namely content-based and collaborative filtering. This section will explain our approach in implementing them and how features were selected for both approaches.

A. Feature extraction

After getting an overview of the structure and distribution of both datasets, it seemed natural to select the genres as a starting point for the different features, as they should be a good indicator of of the animes' content. This assumption is taken into account when considering the distance measure. Considering the fact that the genres are categorical (i.e. non numerical), the content aspect will be represented by 43 binary dimensions - one for each genre. Looking at the remaining possible attributes (i.e. type, episodes, rating and number of members), we chose to disregard the episodes and anime type as to allow for cross recommendations between the different types. This consideration is based on the idea that similar content (e.g. belonging to the same genres) should be considered the same, regardless of the distribution (e.g. TV-serie or Movie). The members attribute was also left out after much consideration, with the reasoning that the member count indicates the popularity of the show rather than the likelihood a user would like it.

Nevertheless, the popularity dimension is kept in a diluted form in the average rating feature for each anime. This attribute has been regarded as relevant as it combines both popularity of the anime with content quality. Based on these considerations the animes dataset has been transformed and 45-featured vectors were created for each of the animes, containing 1 or 0 for each genre representing the belonging to the genre, as well as the average rating for the anime. To cater for the variety of genres, the count of genre for each anime is also registered as a feature. At last the 45d-vector is normalized.

B. User profile

After settling on the features for the animes the problem of fitting users into the same space were faced. To overcome this each *anime_id* in the rating data set was mapped to the corresponding *anime_id* in the anime file. This allowed us to extract the genres and the count of genre for each anime a user had watched. All these attributes would then be summed for each user and the genre count would be divided by the amount of animes to get the average genres per anime. The rating feature would be a fixed 10 for all users to favor higher rated animes.

TABLE V. Sample a user profile

action	adventure	...	yaoi	yuri	genre_count	rating
56.0	0.0	...	4.0	0.0	3.23	10.0
22.0	2.0	...	32.0	48.0	4.11	10.0

At last all the counts for the genres are squared to emphasize higher counts for genres before the profile is normalized according to its own values. A sample profile before squaring and normalizing can be seen in table V.

C. Content-based filtering

The content-based filtering is a method to recommend items based on a comparison between the items and a user profile. As described in section III-A and III-B both animes and user profiles now have similar attributes and can thus be compared.

For comparison the k-nearest neighbours (k-NN) algorithm was used. k-NN is often used for classification or/and regression; however in our case we are only interested in the nearest neighbors measured by euclidean distance as we are searching for animes to recommend. The algorithm itself computes the distance to all the data points in the given samples and returns k neighbours with the lowest distance to the input point.

In our case k-NN was initialized with the anime vectors and $k = 10$, as we wanted to retrieve 10 recommendations for a user. For the input, user profiles were given; k-NN would then return 10 animes, that were closest to the user profile.

D. Collaborative filtering

In this method, we search for user profiles close to the user we want to make recommendations for. We use the k-NN

(k-nearest neighbors) algorithm in the same fashion as in the first method, only this time to find the closest 10 users to the user profile. Afterwards, the apriori algorithm is used to find strong association rules for the animes to recommend to the target user, from the pool of all the animes watched by its neighbors.

In general, the apriori algorithm generates frequent itemsets from transactions. In our case the transactions consist of all the animes a user has watched. An itemset is considered frequent if it is present at least n times in the transactions list, where n is the support threshold. The algorithm uses an approach, where frequent sets are extended with one item at a time, to generate potential candidates, making use of the property that if a m-sized itemset is infrequent, all the m'-sized itemsets containing it will be infrequent, with $m' >= m$. After this step, only those itemsets that meet the support threshold are taken into account. Further, association rules are generated, by considering only the rules that have a confidence level higher or equal than the confidence threshold.

As mentioned in the previous section, each user is represented with a feature vector as in the content based filtering. Unlike the previous method, the vector doesn't include average rating and genre count, because those dimensions were considered more content specific. Using the k-NN method we find the 10 closest users to the target user, limiting therefore the number of potential candidates as input to the apriori algorithm.

After finding the closest users, mining of association rules on the animes, watched by them, is executed. We aim to generate at least 10000 rules to be able to make good recommendations. For that we initially set both the support and the confidence to 1.0. We don't set fixed values for these two parameters; instead, we gradually decrease them until we generate 10000 association rules or until we hit the threshold for the support and confidence, both set to 0.55. We don't generate rules for support and confidence less than 0.55, because we would like to have support at least from the half of the users to make good enough recommendations.

The next step includes analysis of the association rules generated by the apriori algorithm. The chosen approach was to put a restriction that association rules with only one anime in the right side will be considered. No matter how many animes in the left side of the rule are watched by the user, we recommend the anime in the right side of the rule. Out of the recommended animes, we don't consider those that are already watched by the target user.

The result recommendation set later is sorted by the chain size (i.e. left part of the association rule), lift, support and then by the confidence value, since we consider a long chain, which is frequently met in the list of transactions, a strong indicator for the recommendation to be accurate. We take the first 10 animes after the sorting. If there are less than 10 recommended animes, we take all in the specified order.

E. Data preprocessing

Based on the observations made in the data exploration phase, the following data preprocessing decisions have been undertaken:

1) All the animes with an average rating lower than 5 have been removed from the dataset. The reasoning behind is that the recommendations should not contain badly rated animes - i.e. animes with a rating below the first quartile of the distribution.
2) Simple outlier detection applied on the users and the number of watched animes - 2% winsorizing. This involves removing the first and last 1% of the users from the distribution of animes watched per user.
3) As the purpose of this recommendation system's design is to recommend 10 animes to each user, we decided not to consider users with less than 10 watched animes.

The resulting dataset after all the pre processing steps were undertaken contains information about 60785 users and their watched animes.

IV. Experiments

As described in the previous section, the two approaches in generating recommendations for the users have been implemented and the comparison of the results of each of the method yields constitute to the findings discussed later in this paper.

The preprocessed dataset described in section III-E has been split into two subcategories, train and test subsamples, accounting for 80% of the ratings.csv file and 20%, respectively. The split has been done for each user, which means that all the users are present in both train and test subsamples, since the purpose of the recommendations is to individually suggest new content to the user, based on the their activity (i.e. watched animes). Once the split has been made, two csv files have been generated to serve as input to the next processing steps.

In both cases (content based filtering and collaborative filtering), the normalized user profiles have been generated out of the train data. We run two experiments on a subsample of these user profiles (recommendations were generated only for 955 user profiles), due to computational challenges.

The output of the two experiments are 20 recommendations for each of the user: 10 coming from the content based filtering and 10 generated by the collaborative filtering.

This output is afterwards matched with each user's test sample to identify how many recommended animes the user has actually seen. The results are evaluated first individually for each of the methods (collaborative and content based filtering) and afterwards the union of the two is checked against the test sample. The count of matches between union and test file are divided by the minimum among the size of test sample and the size of the union - so maximum 20 - in order to get a normalized estimate of the results' accuracy.

V. Findings

This section will present the findings from the experiment outlined in the previous section.

Taking a first look at the findings in table VI we immediately see that collaborative filtering is performing much better than the content-based filtering. The union column shows

TABLE VI. SUMMARY OF ACCURATELY RECOMMENDED ANIMES

	CB	CF	Union
$>= 1$	66	780	789
mean	0.07	2.12	0.14
std	0.27	1.76	0.11
25%	0.00	1.00	0.05
50%	0.00	2.00	0.12
75%	0.00	3.00	0.20
min	0.00	0.00	0.00
max	3.00	10.00	1.00

the score in percent when testing against the union of the content-based and collaborative filtering. CF is on average classifying 2.12 out of 10 animes correctly whereas CB is not even able to classify a single movie on average. CB is only able to accurately recommend a movie in $66/955 * 100 \approx 6.9\%$ of the test cases which is significantly less than CF $780/955 * 100 \approx 81.7\%$ which is only marginally worse than the union $789/955 * 100 \approx 82.6\%$.

The distribution of the collaborative filtering shown in figure 7.

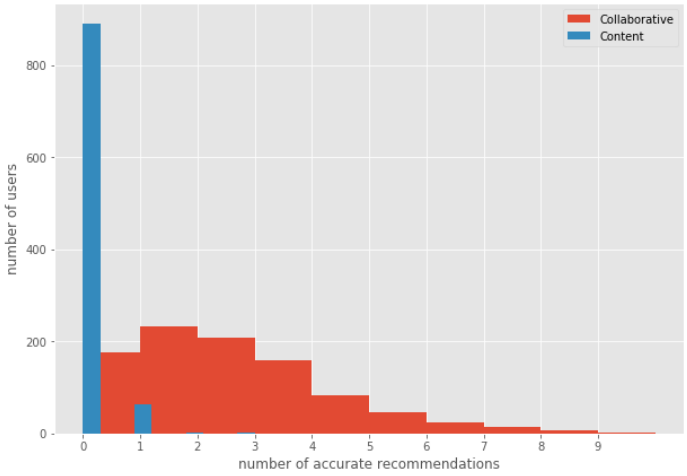

Fig. 7. Distribution of the amount of accurately recommended movies

VI. DISCUSSION

One interesting challenge the dataset and the task at hand have posed is the validation of the results. Generating user recommendations has not been approached as a standard classification or regression task, therefore measurements like misclassification error or accuracy did not entirely help in reaching meaningful conclusions.

Since there was no time dimension in the ratings dataset, signalizing the order in which the users saw the animes, it was not possible to discriminate between the animes the users watched and therefore the $80\% - 20\%$ split has been made in a arbitrary way. In this case the train data does not reflect the state of a user at specific point in time, to be able to predict the future actions and compare with the test sample, but rather an incomplete picture of its preferences.

In section IV - Experiments - we refer to computational challenges that restricted us from providing the results for the entire dataset, but only for a subsample of it. This is mainly due to the collaborative filtering method, which has high memory requirements in order to generate association rules for each target user. In trying to keep a balance between always recommending 10 animes to a user and restricting the number of generated association rules, for computation feasibility on personal machines (i.e. $8\ GB\ RAM$), the solution chosen was to display the results of a randomly selected subsample.

An illustrative example of this can be seen in figure 8, where the running time for generating the recommendations for user 108. Initially, as it can be seen, the algorithm generated $14.7\ mil$ rules to filter through.

```
Results for user  108
Closest neighbours identified in 0.453000068665 seconds!
We found 14702470 potential rules! Let's check them out!
Rules found in 805.512000084 seconds!
```

Fig. 8. Generating collaborative filtering recommendations for one user

While this has been partially alleviated by improving the implementation with a dynamic descending assignment of the confidence and support levels for generating association rules, as well as stopping right after exceeding the 10000 rules threshold, running collaborative filtering on the entire dataset required both more computational power and memory, and longer time.

We believe that the rather less than impressive results of the content-based filtering comes down to the way we generate the user profiles and compare them with animes. User profiles mainly consists of an aggregation of genres and do not exactly reflect what movies they like. A user might only see an equal amount of action and comedy animes however our generation of his user profile would say that he likes action, comedy animes which is not entirely true since those animes only consisted of 1 genre. For future references, instead of trying to fit a user profile within the anime space, other approaches that focus more on comparing the different animes' attributes directly, rather than an aggregation of them, might yield more accurate results.

VII. CONCLUSION

The experiments performed and presented in this paper highlight some of the differences between the two approaches in generating user recommendations - content based and collaborative filtering. We conclude that the primarily genre-based user profiles worked better at identifying similar users, rather than animes, and that applying the apriori algorithm on animes directly, to identify strong collaborative filtering association rules, yielded better results.

REFERENCES

[1] Rajaraman A. Leskovec J. and Ullman J. D. *Mining of massive datasets*. Cambridge University Press, 2014.

APPENDIX

Word count: XXXXXX

Character count: YYYYYY