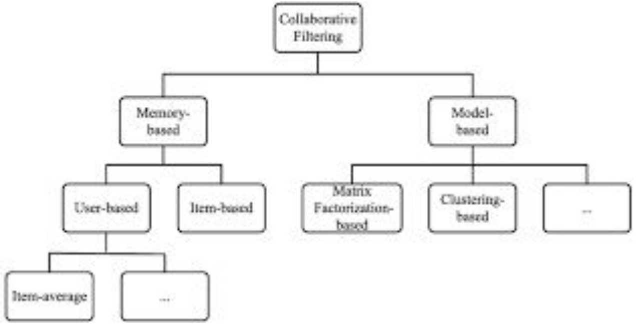
Anime Recommendation System

By: Liam Cannon,Yusra Suhail, Linda Tran

Collaborative Filtering



Collaborative filtering is a method used by recommendation systems to predict the interests of a user by collecting preferences from many users

Memory-Based: User-Based versus Item-Based

User-Based Collaborative Filtering:

- **Goal:** Recommend anime to a user similar to other selected items
- **Process:**
    - Calculate the similarity between users based on their ratings of various anime
    - Find users who are most similar to the target user
    - Recommend anime that these similar users rated highly but the target user hasn't seen yet

Item-Based Collaborative Filtering:

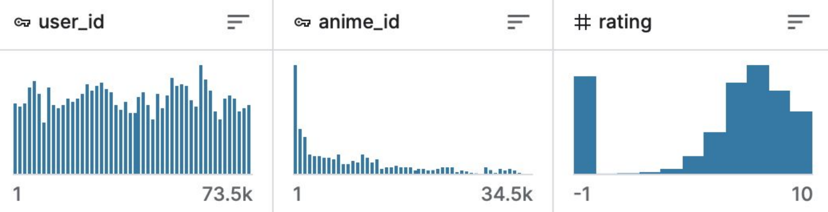
- **Goal:** Recommend anime similar to those the user already likes
- **Process:**
    - Calculate the similarity between anime titles based on how they've been rated by users
    - For given anime that a users likes, find other anime with high similarity scores
    - Recommend these similar anime to user

Dataset: rating.csv

Key Features on Dataset:

- ‘**user_id**’ - non identifiable randomly generated user id.
- ‘**anime_id**’: the anime that this user has rated
- ‘**rating**’: rating out of 10 this user has assigned (-1 if the user watched it but didn't assign a rating)

Anime data



user_id	anime_id	rating
1	20	-1
1	24	-1
1	79	-1

Dataset: anime.csv

Key Features on Dataset:

- ‘anime_id’ - myanimelist.net's unique id identifying an anime.
- ‘name’ - full name of anime.
- ‘genre’ - comma separated list of genres for this anime.
- ‘type’ - movie, TV, OVA, etc.
- ‘episodes’ - how many episodes in this show. (1 if movie).
- ‘rating’ - average rating out of 10 for this anime.
- ‘members’ - number of community members that are in this anime's "Group".

Detail Compact Column 7 of 7 columns

anime_id	name	genre	type	episodes	rating	members
32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630

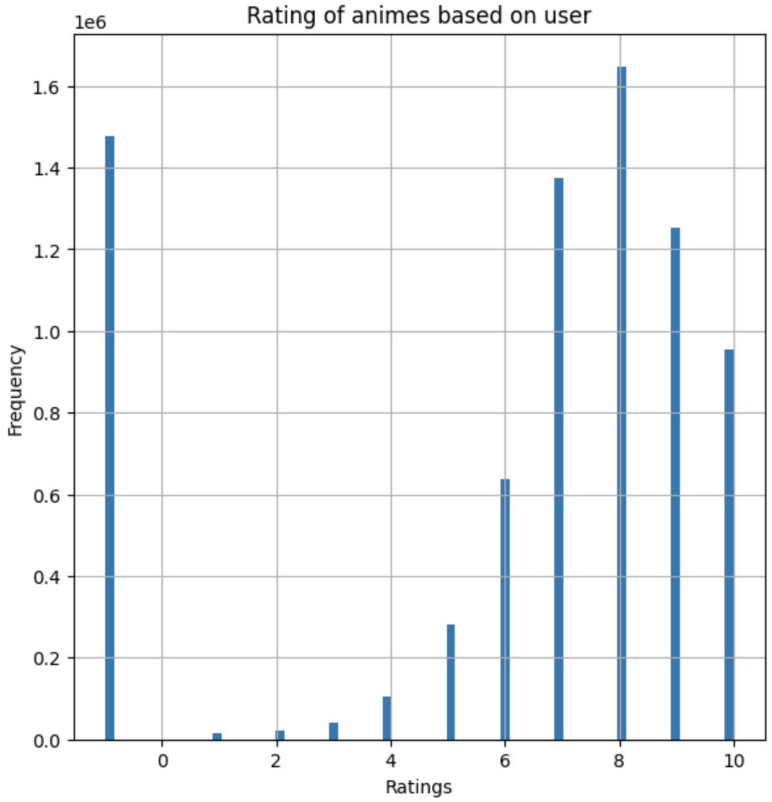
Challenges

- **Scalability:** Managing a large dataset with many users and items is computationally expensive
  - Problem: Google Collab crashes, runtime randomly disconnects with many users
  - Hard to run exhaustive hyperparameter searches because of how long runs can take
  - Ex: we would like to have experimented with more values for the cutoff for item-based similarity, but we had to settle for manually approximating the ideal cutoff
- **Data Sparsity:** Occurs in systems with many users and items but limited interactions
  - Example: Users may watch or rate only handful of anime series, leaving many titles with titles with few or no rating
  - Problem: Causes difficulty to predict user preferences due to limited data
  - Even users with many rated anime often have no neighbors with a high similarity score
  - Setting threshold of ratings for our selected users decreases generality
- Our problem, while falling under Nearest Neighbors, did not cleanly integrate with the prebuilt models we tried. So the training/ prediction is done manually (more prone to errors)

Data Preprocessing

1. Our dataset has -1's where the user has watched the show but hasn't rated it
    a. Poses a problem: how do we know what they think about the show?
    b. Solution: give the show the mean rating of the user , assuming they have a neutral opinion (assumption)
2. Mean center data
    a. This avoids bias of "tough/easy" graders, also makes comparisons between users more meaningful
3. Thresholding:
    a. Due to computational limits, and to remedy the problem of new users / low rating users, we set a limit to users who have rated 1000 anime (165 left) : any lower threshold becomes very time-consuming
    b. But ideally we can test with much lower thresholds

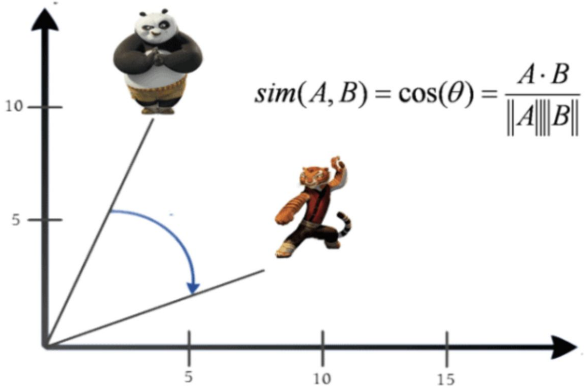
Rating frequencies among users



Cosine Similarity Matrix - User Based vs Item Based

Cosine Similarity: Computes the similarity between users based on their anime ratings and helps us get similar users based on their preferences

- To calculate this , we have to make a table that relates every user to every item, which is not how our data is natively stored (pivot table)
- Similar process for item based searching : this calculates the similarity between items based on how users interact with them
  - The important consequence here: can get similar items without having close neighbors
- The crux of our implementation: if there are no close neighbors , do item-based recommendation. If there are sufficient neighbors, user-based recommendation is preferable



Methodology

- isWatched? : function is crucial so we don't give recs the user has already seen
- User based functions: Get similar users (query the cosine similarity for this)
    - Predict rating : Given user's similar neighbors , predict their scores using the weighted ratings of users
        - Weight ⇔ distance

```python
def predict_rating(anime_id,user, top_sim_user_sim_scores ):
  rating_list = np.array([])
  weight_list = np.array([])
  top_users = np.array([])
  # if and only if the current user had top users then continue else switch to item_based.
  if  top_sim_user_sim_scores != None:
    for i, (sim_user, similarity_score) in enumerate(top_sim_user_sim_scores):
        # for each top_user who has rated the particular anime, get their ratings, and only use it for prediction if rating is not nan
        rating_for_anime = user_ratings_matrix_nans_present.loc[sim_user, anime_id]
        #this decreases the available neighbors by a lot . This is a known problem of sparse datasets in recommendation problems.
        if np.isnan(rating_for_anime):
             continue
        elif not np.isnan(rating_for_anime):
            # Populate rating list, top_user list and weight list where similarity score is the weight that will be added to the ratings
            top_users = np.append(top_users, sim_user)
            rating_list = np.append(rating_list, rating_for_anime)
            weight_list = np.append(weight_list, similarity_score)
    # calculate avg of weight so that the total sum of all weights is 1, and we can see the total contribution of each user in the ratings.
    weight_list = weight_list / weight_list.sum()
    # multiply ratings with new weights, to get weighted average of each of the ratings
    weighted_avg_rating = rating_list * weight_list
    weighted_avg_rating_df =  pd.DataFrame({"User": top_users, "rating list": rating_list})
    return weighted_avg_rating.sum()
  else:
    return item_based(user)
```

Methodology (Cont) : Giving Recs/Error Calculation

- For each item the user has watched, calculate a predicted rating for it .
- To calculate the user’s error for one item, their actual rating is subtracted by their predicted.
- The Average error of all predictions: mean error of each user, where each user’s error is the mean of error over all watched shows.

1 to 25 of 165 entries Filter

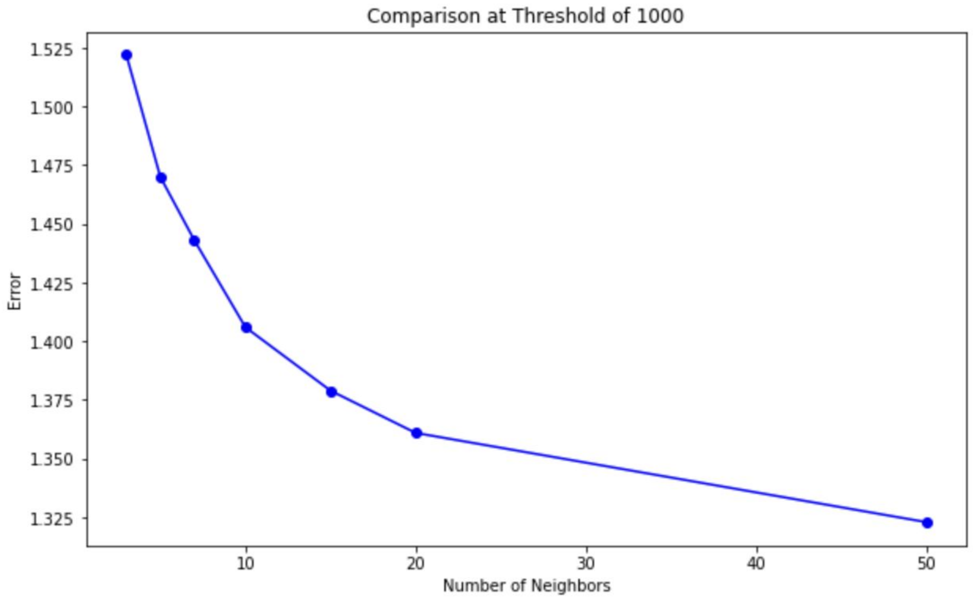
index	User	Anime List	ApproachType	RMSE Error
0	1497	Clannad: After Story,Steins;Gate,Fate/Zero 2nd Season,Clannad,Fullmetal Alchemist: Brotherhood	user-based	1.5247056211262666
1	1530	Eyeshield 21,Hachimitsu to Clover,Hungry Heart: Wild Striker,Initial D Fourth Stage,Monster	item-based	N/A for item-based
2	2951	Witch Hunter Robin,Beet the Vandel Buster,Eyeshield 21,Hachimitsu to Clover,Hungry Heart: Wild Striker	item-based	N/A for item-based
3	3569	Fullmetal Alchemist: Brotherhood,Death Note,Clannad: After Story,Suzumiya Haruhi no Shoushitsu,Steins;Gate	user-based	1.1501827695693692
4	5310	Code Geass: Hangyaku no Lelouch R2,Fullmetal Alchemist: Brotherhood,Clannad: After Story,Death Note,Code Geass: Hangyaku no Lelouch	user-based	1.3815038897348466
5	5908	Beet the Vandel Buster,Hungry Heart: Wild Striker,Initial D Fourth Stage,Monster,Prince of Tennis	item-based	N/A for item-based
6	6384	Eyeshield 21,Hungry Heart: Wild Striker,Initial D Fourth Stage,Naruto,Prince of Tennis	item-based	N/A for item-based
7	6569	Clannad: After Story,Fullmetal Alchemist: Brotherhood,Steins;Gate,Death Note,Fate/Zero 2nd Season	user-based	0.8800353155196015
8	6969	Clannad: After Story,Fate/Zero 2nd Season,Code Geass: Hangyaku no Lelouch R2,Fullmetal Alchemist: Brotherhood,Angel Beats!	user-based	1.1438410888684507
9	7081	Witch Hunter Robin,Beet the Vandel Buster,Eyeshield 21,Hungry Heart: Wild Striker,Initial D Fourth Stage	item-based	N/A for item-based
10	7114	Clannad: After Story,Fullmetal Alchemist: Brotherhood,Steins;Gate,Death Note,Fate/Zero 2nd Season	user-based	1.2881163010111605
11	7247	Fullmetal Alchemist: Brotherhood,Death Note,Code Geass: Hangyaku no Lelouch,Code Geass: Hangyaku no Lelouch R2,Clannad: After Story	user-based	1.5087586515795306
12	7249	Clannad: After Story,Sen to Chihiro no Kamikakushi,Cowboy Bebop,Monster,Death Note	user-based	1.3994229892124543
13	7345	Monster,Sen to Chihiro no Kamikakushi,Death Note,Clannad: After Story,NHK ni Youkoso!	user-based	1.1152341639375642
14	7366	Witch Hunter Robin,Beet the Vandel Buster,Eyeshield 21,Hachimitsu to Clover,Hungry Heart: Wild Striker	item-based	N/A for item-based
15	7511	Fullmetal Alchemist: Brotherhood,Steins;Gate,Death Note,Code Geass: Hangyaku no Lelouch,Code Geass: Hangyaku no Lelouch R2	user-based	1.2737224747657245
16	8006	Death Note,Clannad: After Story,Monster,Fate/Zero 2nd Season,Fullmetal Alchemist: Brotherhood	user-based	1.512886827791668
17	8094	Sen to Chihiro no Kamikakushi,Cowboy Bebop,Neon Genesis Evangelion,Tengen Toppa Gurren Lagann,Clannad: After Story	user-based	1.5802733011146015
18	8115	Clannad: After Story,Fullmetal Alchemist: Brotherhood,Steins;Gate,Monster,Tengen Toppa Gurren Lagann	user-based	1.2959865355010818
19	8122	Code Geass: Hangyaku no Lelouch R2,Code Geass: Hangyaku no Lelouch,Tengen Toppa Gurren Lagann,Summer Wars,Steins;Gate	user-based	1.2819701219787025
20	8149	Steins;Gate,Fullmetal Alchemist: Brotherhood,Tengen Toppa Gurren Lagann,Clannad: After Story,Monster	user-based	3.748898865148284
21	8217	Suzumiya Haruhi no Yuuutsu,Fate/Zero 2nd Season,Death Note,Clannad: After Story,Fate/Zero	user-based	0.986548682475263
22	8820	Cowboy Bebop: Tengoku no Tobira,Trigun,Beet the Vandel Buster,Eyeshield 21,Hachimitsu to Clover	item-based	N/A for item-based
23	9032	Clannad: After Story,Fate/Zero 2nd Season,Clannad,Code Geass: Hangyaku no Lelouch R2,Fullmetal Alchemist: Brotherhood	user-based	1.2483533408624499
24	10419	Initial D Fourth Stage,Monster,Ring ni Kakero 1,Sunabouzu,Texhnolyze	item-based	N/A for item-based

Show 25 per page 1 2 3 4 5 6 7

Like what you see? Visit the data table notebook to learn more about interactive tables.

```
Average error across users:  1.3795227911854246
```

Evaluation: RMSE Values when using different number of neighbors (All at the threshold of 1000)



Conclusions

- At k = 15 (elbow point) , our average RMSE is 1.375 . Considering how our ratings can only have a 10 point spread (1-10), this is not perfect !
- At higher thresholds , we get lower RMSE's, but this is outweighed by the scarcity of users at those levels
- Improvements we could make :
    - Use a pipeline for estimating ratings threshold/item-based cutoff - means we need more processing power/ better algos
    - Use a combination of item based and user based instead of conditionally doing one or the other.
    - Calculate the cosine similarities differently : the act of imputing the mean for all the missing values can likely be done better / needs to be examined
    - Both csv files could be merged together so we can have all the data at one place
    - Shows can be recommended based on popularity too, not only on the basis of what they have seen

References

https://www.kaggle.com/datasets/CooperUnion/anime-recommendations-database?select=rating.csv

https://colab.research.google.com/drive/1HvNKhTfAMsJ3_FE3I7aEdxymJzo_KVuB?authuser=1#scrollTo=uvmZ7-Fa6tdh (Our Colab)

Source of Cosine similarity Image:
https://www.kaggle.com/code/benroshan/content-collaborative-anime-recommendation

Questions / Comments ?

Thank you !