Recommender Systems Final Project: User Interface Images

HAILEE KIESECKER*, Boise State University, USA

(user story)As a creative User Interface (UI) developer, I can put a User Interface image into a UI recommender and have it return similar recommended UI images.

CCS Concepts: • **Computer systems organization → Recommendation systems**; *Algorithms*; • **Networks** → Network reliability.

Additional Key Words and Phrases: datasets, neural networks, recommendation systems

ACM Reference Format:
Hailee Kiesecker. 2021. Recommender Systems Final Project: User Interface Images. *ACM Trans. Graph.* 37, 4, Article 111 (August 2021), 6 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

The goal of this recommender is to assist developers in interface layout design, by taking developer User Interface (UI) layouts that they like and returning similar images that the user might find helpful in development. Developers may want to use this recommender to give them new ideas on a layout that they really want to implement but something isn't quite there. The domain of this project is specifically mobile application UIs. Developers will input into the system a UI mobile application screenshot when they run the python script. After, a few suggested images will output to them to help with their design process. Expected characteristics of returned images have a basis of similarity between them and the input image. For example, if a user inputs an image of a login screen, then the recommended UI's return should be images of login screens based on a metric of distance and a type of page rank that takes into account how popular the output image is to other login screens. This measure of popularity can be calculated based on how many similar images that image has.

Figure 1 is a graph structure of the current dataset representing connected nodes based on a range of similarity. The closer the node is to the center, the more connected node is, and by extension the nodes page rank is higher.

2 PROJECT OVERVIEW

In the following sections we discuss the methodology's used to create this project.

Author's address: Hailee Kiesecker, haileekiesecker@u.boisestate.edu, Boise State University, 777 W Main St, Boise, Idaho, USA, 83702-0000.

0730-0301/2021/8-ART111 $15.00
https://doi.org/10.1145/1122445.1122456

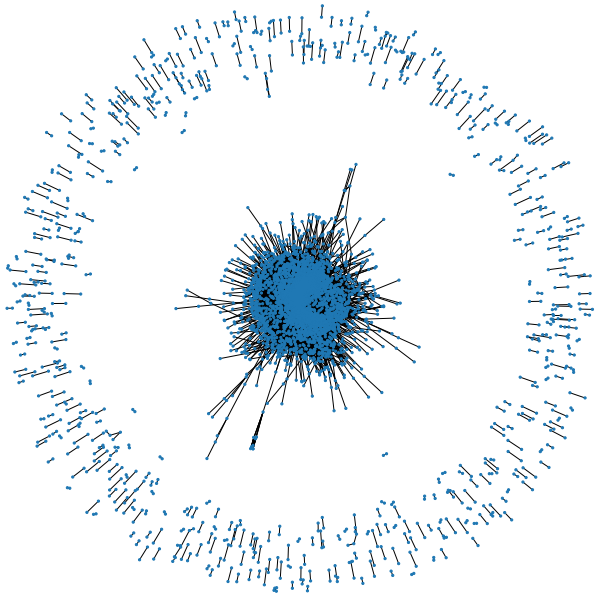

Fig. 1. pageRank formation where each node is an image and connections are within a degree of similarity. Closer to the center the node is the more similar the image is to others.

2.1 The Data

RICO is a public data-set that has around 66k unique UI screens each containing element details of the screen, and a 64-dimensional vector representation of the image. The vector representation of the UI can be used to help with image clustering to retrieve similar UI's from new information sources. For this project we took around 3.6K images from RICO and created our own vector representations for them each vector is of shape (1024, 1). In creating our own basic vector representations of the images instead of using the already supplied vectors, we allowed ourselves to replicate the process to add future images to the data for recommendations.

Initially in trying to retrieve the images from RICO it was planned to use the full dataset. However, Complications occurred in the downloading process and image processing speed so a sample dataset of the 3.6K+ was used instead. An ideal dataset if this would to actually be put into production would be using the full RICO data set along with its integrated vectors since they are derived from an autoencoder manipulating the image instead of a basic formula. As users started to interact with the recommender it would begin to grow itself as new images are added, and store new data in the same manner as the old.

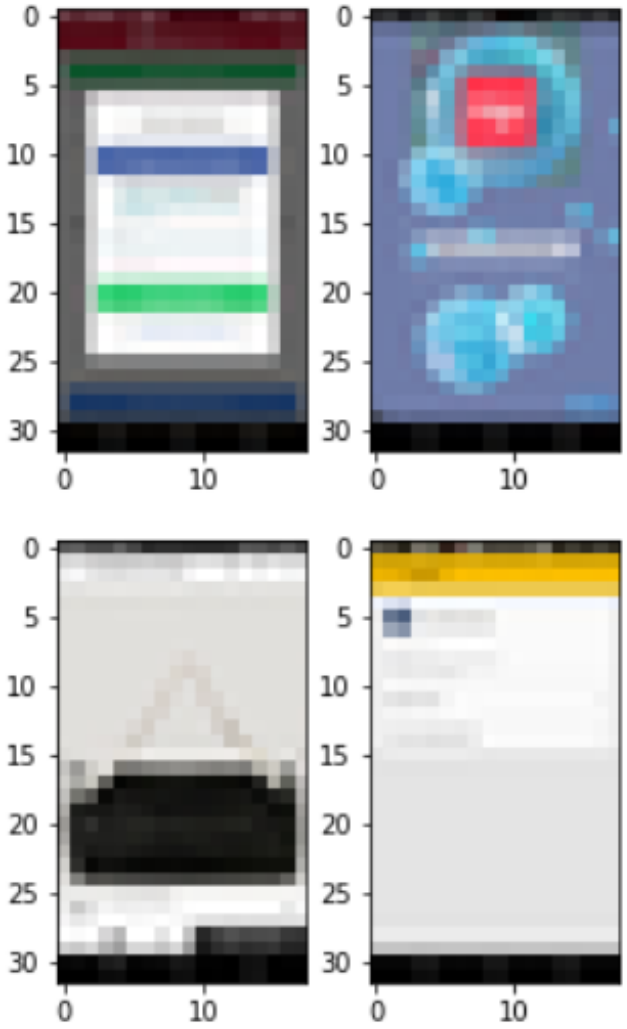

Fig. 2. Example of RICO dataset transformed to a 32x32 image for data processing.

2.2 Measurement of Distance

The main goal of this project is to return image results to developers with images that are relevant to what the user provided. To do so a measurement of distance is needed to see how similar or dissimilar images are to each other in the dataset. There are many different kinds of distance similarities that could be used. Since this is only a toy implementation of the overall recommender we ended up implemented a Euclidean Distance of similarity view Equation 1. Different types of measures of distance could of also been used, such as; Cosign index, is a distance measure that is able to compute the angle between two vectors by following the formula $sim(A, B) = cos(\theta) = \frac{A*B}{||A||B||}$. or, Minkowski distance, which is the generalized form of Euclidean ($p = 2$) and Manhattan ($p = 1$) distance. The distance between x and y can be computed as $\sqrt[p]{(x1 - y1)^p + (x2 - y2)^p + ... + (xN - yN)^p}$. Euclidean

Fig. 3. Example of RICO dataset image original version as those referenced in Figure 2.

distance was used because to the developer it is a basic form of distance measurement that has a lot of documentation and they are familiar with it.

$$d(x, y) = d(y, x) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2}, \quad (1)$$

Equation 1 can be described as two points in Euclidean n-space x and y where x_i and y_i are Euclidean vectors starting from a space of origin in n space. This formulation was chosen as a base point of implementation. Since its computation time is relatively quick it posed as an optimal start to the creation of this project. As further development occurs on this project different distance of measurements could be used and compared to see which ones operate the best for recommendations.

3 DATA PREPARATION

To get the RICO data set to a point of usability for the recommendation system, we compressed each image to 32x32, calculated its N-Dimensional vector representation. Then for each image in the dataset we calculated its distance between each other effectively

making a distance matrix. This could have been optimized but it ended up getting the job done, this part of the data preparation process currently runs in $O(n^2)$ time and takes around 30 minutes. After the distance from each image to each other is calculated, a range of distance scores can be used to be associated with each image. Images with more distance scores associated with them (or in other words more images associated with them) have a higher pageRank score and vice versa. A similar application of this is called visualRank which is practically equivalent to a google search web page recommendation algorithm however done with relevant images searches.

After the data is all collected and processed it is free to run different experiments on, outlined in the next section. The data we have at this point can be seen in Figure 4.

	image_name	simple_vector	similarImages	rank
0	33302	[[128]\n [187]\n [181]\n ...\n [0]\n [0]\n...	[64037, 57581, 66744, 53119, 56977, 52743, 648...	0.000138
1	33305	[[56]\n [63]\n [67]\n ...\n [0]\n [0]\n [0]]	[65297, 47508, 51054, 58030, 53323, 35104, 333...	0.000094
2	33312	[[61]\n [170]\n [245]\n ...\n [0]\n [0]\n...	[36889, 57581, 56977, 64831, 59339, 61276, 690...	0.000056
3	33328	[[36]\n [36]\n [36]\n ...\n [0]\n [0]\n [0]]	[33328]	0.000135

Fig. 4. Portion of the recommender systems end data after preparation and transformations.

At this point basic recommenders were run on the data. It was during this process of very low NDCG scores for baseline algorithms that we discovered that our preparations were not done. View figure 5.

After this discovery we transformed our data set to each image being a user, and each users closest image is an item. The rating assigned to each user item pair is $distance * itempagerank$. yielding the following data setup, see figure 6.

The reasoning for this is before we were assuming that the recommender algorithm could properly sort through similar images and use the image/user page rank as a rating. However, this was a very naive perception. The previous data setup is needed to create the new data set, which is now at a size of 734k+ user item pairs. As users add more images to the dataset by using the recommender the same process of converting the image to a vector, comparing that vectors distance with every image vector in the dataset needs to take place. Along with filtering down and storing images that have the shortest distance away from that image to calculate the page rank of the new image. once all that is done we now have a feasible data set to explode for user item recommendations for that new image.

Using basic recommendation algorithms on the new exploded data Figure 7 shows the ndcg scores for each one. Notice BPR strongly out positioning the other recommendations.

4 EXPERIMENTS

Retrieving the scores outlined in Figure 7, we separated the data into 3 partitions with each training set containing a vast majority of the data, mainly due to the fact that splitting it more evenly was

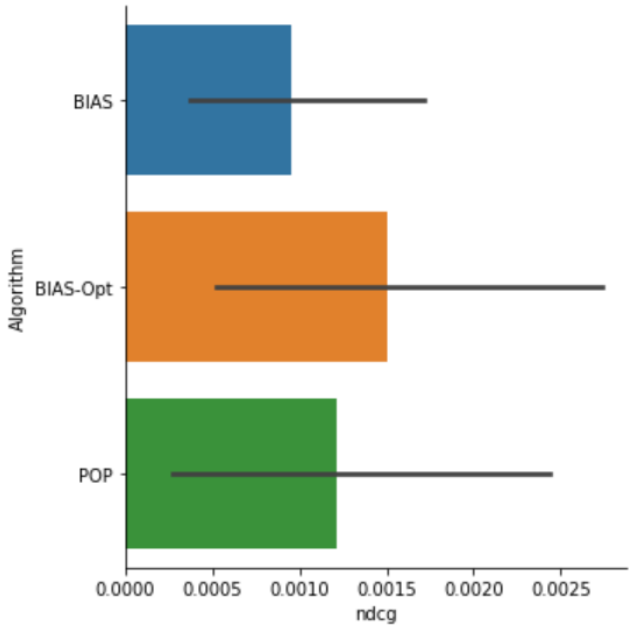

Fig. 5. Using our prepared dataset yeilds horrible ndcg scores

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 736968 entries, 0 to 736967
Data columns (total 3 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   user    736968 non-null  int64
 1   item    736968 non-null  int64
 2   rating  736968 non-null  float64
dtypes: float64(1), int64(2)
memory usage: 16.9 MB
```

Fig. 6. New dataset for recommender after explosion and rating calculation

causing errors in computation. The testing sets ended up containing around 2k data entries, issues with this include over fitting however, modifications to the train/test partitions is on the back burner of this project.

Running individual recommendations was done in a standard loop iterating over each available recommendation algorithm. Starting with converting the *algo* to a recommender and then $fitting$ each $trainingpartition$ on the data. After which iterating over each $testingpartition$ on the trained models was done and the $NDCGscore$ was measured. We chose the NDCG score as a basic means of measurement due to its ability to predict a sorted list of relevant images and compare it with truly relevant images. While our NDCG scores

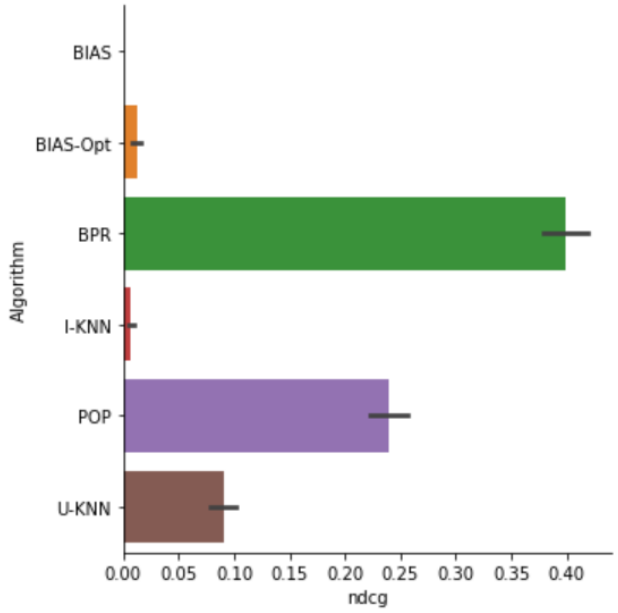

Fig. 7. New NDCG scores for recommender algorithms after explosion and rating calculation

only reached 40 percent using the BPR recommender, we believe this mostly occurred due to over fitting the model.

In the following subsections we are returning top 3 recommendations giving the same starting image using our top 3 algorithm NDCG scores; those being BPR, Popular, and User KNN. You will also note no modifications to these recommender algorithms were made. This is because during the data processing we have already taken into account our unique dataset and modified the data ranking based on the page rank of each image, making each basic algorithm already unique.

4.1 BPR Algorithm

The starting image chosen for this test was *image* : 33633, its top 3 recommended images were [*recImage*01 : 54813, *score* : 2.290359], [*recImage*02 : 36017, *score* : 2.288895], [*recImage*03 : 71619, *score* : 2.245636], view Figure 8.

You can notice similarities between the core structure of the image similarities. Such as the top bar area, and a portion of the top of the page being used with either a list or a form in comparison to location temperatures in a grid format.

4.2 Popular Algorithm

The starting image chosen for this test was *image* : 33633, its top 3 recommended images were [*recImage*01 : 52291, *score* : 1646.0], [*recImage*02 : 39031, *score* : 1621.0], [*recImage*03 : 51683, *score* : 1618.0], view Figure 9.

For this example returned images are completely blank which is likely why they are rated so high on similarity between other images, nothing to compare anything to might be causing it to be similar more often than not.

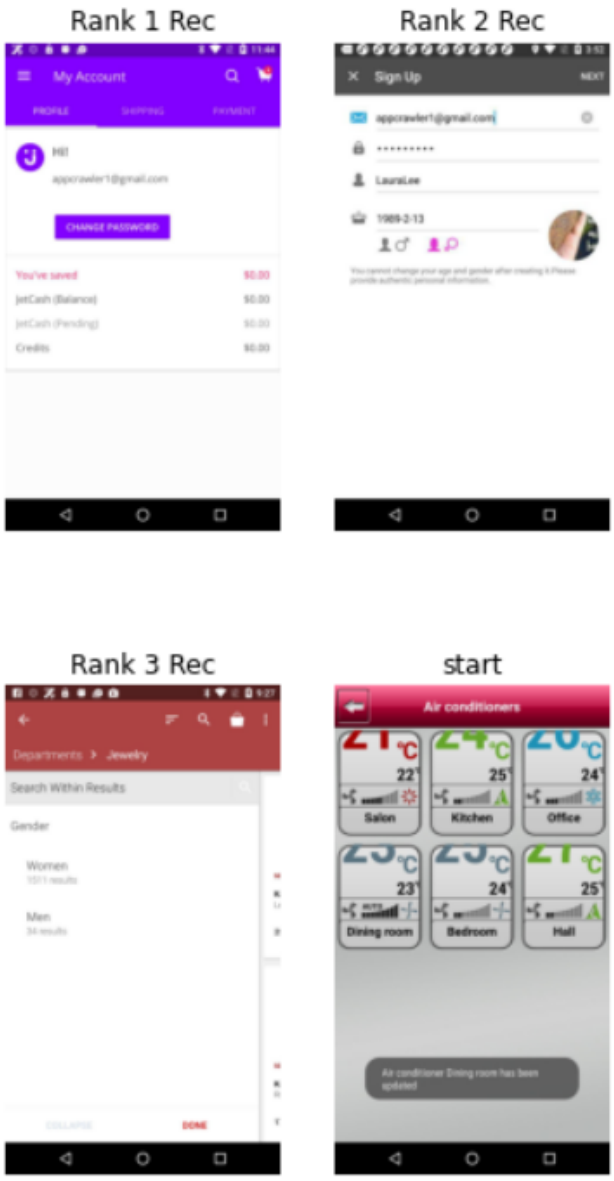

Fig. 8. Top 3 recommendations using BPR

4.3 User KNN algorithm

The starting image chosen for this test was *image* : 33633, its top 3 recommended images were [*recImage*01 : 60863, *score* : 0.460670], [*recImage*02 : 68143, *score* : 0.456066], [*recImage*03 : 60799, *score* : 0.434411], view Figure 10.

While mainly empty space was also returned like Popular, User KNN returned impressing results. Each recommended image has a top bar just like the start image. Additionally the similarities between the images seem to be interpreting the temperature grid as a list object, both recommendation 1 and 3 have a list object in their main field. Then finally the most impressive in our opinion,

Fig. 9. Top 3 recommendations using Popular

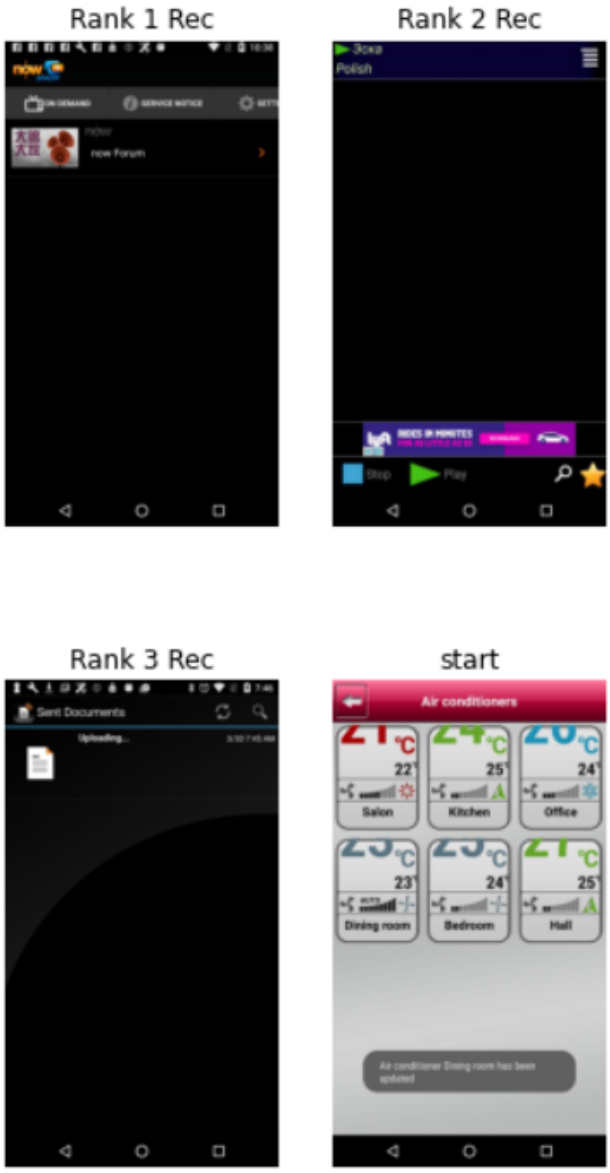

Fig. 10. Top 3 recommendations using User KNN

recommendation 2. This image has a add object in the same general area as the starting images air conditioner notification.

4.4 Further Experimentation

Based on the results in the previous subsections, additional visuals were ran on both BPR and User-KNN. What ended up happening is while User-KNN continued to give decent image recommendations such as in Figure 11. The BPR recommendations started to become the same top 3 images for any random image input. Even after debugging to see if it was user error, the same issue was occurring. Now it might still be user error, but for this reasoning User-KNN seems to be the best recommendation algorithm for our functionality, which makes sense. The user nearest neighbor recommendation algorithm calculates the similarity between a target user (in our case image) to all other images, it then selects the top most similar images based on the weighted average of ratings from these similar users, see Equation 2. i.e. if an user:image01 is similar with another user:image02, and user:image02 is associated with item:image03, then it is likely that user:image01 will also be associated with item:image03.

$$r_{ij} = \frac{\sum_k Similarities(u_i, u_k) r_{kj}}{numberRatings}, \tag{2}$$

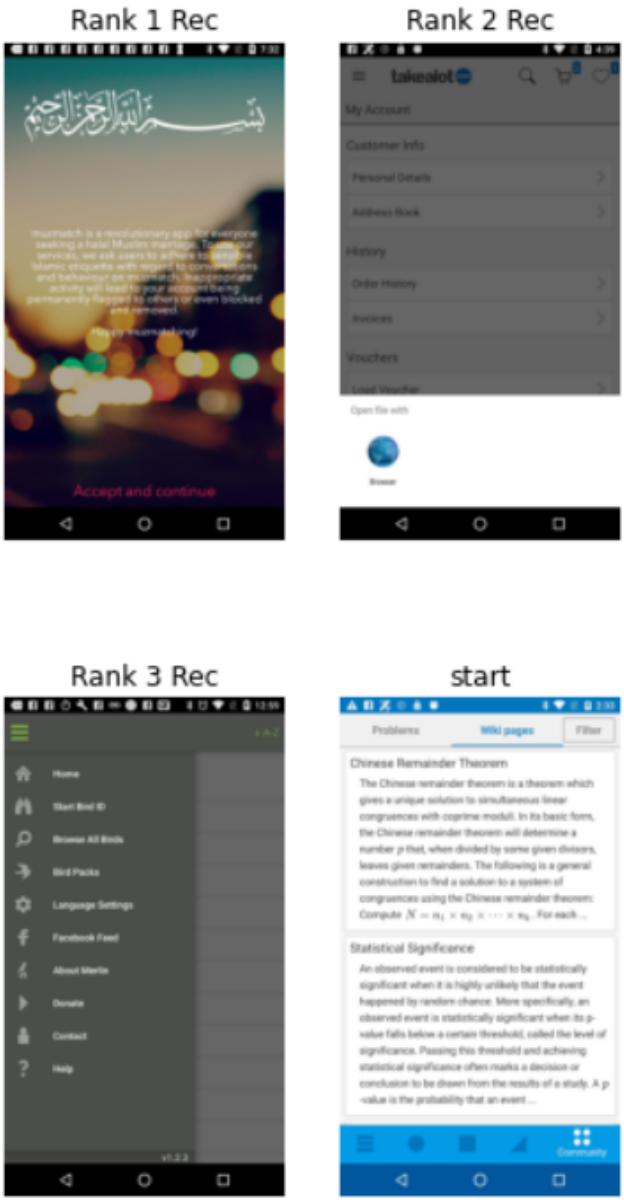

Fig. 11. User-KNN Image Similarity Extra Example

5 REFLECTION

For this recommender to be used in production new methods would need to be created so that the thousands of calculations does not need be ran when a user adds a new mobile UI to the dataset. Additionally from how the vector representation of each image was made, there is 2 distance measures between any two given node points. When creating the cleaned data we dropped the first and kept the second. But this quick fix is not exactly accurate. If done again another methodology would be used to calculate the distance between different nodes so that only half the calculations need to be made.

Throughout this project I learned more innovative ways of thinking involving data. I would start with a solution to a basic issue in my minds eye, but when it come to actually implementing the "simple" it became quite difficult just with how I was going about it. There was a time that I waited 5 hours for code to compile and timeout and when I came back to it later I realized all I was doing was a complicated version of an inner merge on two columns. Additionally I learned that how you process data has a huge effect on how good recommendations are. While this set up actually ended up preforming better than I had hoped, how images were initially transformed into one dimensional vectors for image similarity, likely has an effect on the overall quality that the recommendations returned are. The process used for vector transformation was entirely basic, and I am very curious to what the results would yield if we used an autoencoder to transform the images into vectors.

5.1 Course Reflection

Throughout this course I have learned how to make python scripts, evaluate data, and estimate best results for recommender. I learned about different types of recommender and how some recommender and measurements are not always ideal depending on the data and recommendation type. My favorite part about this course by far was the project. It allowed creativity and implementation of what I found useful and liked. In the future I think this course would be more enjoyable if the overall datasets were smaller so that students with low profile machines can have a good time and accomplish learning objectives.