Myung Sub Cho, Justin Tahara, Zhanpeng (Danny) Zhang

Professor McAuley

CSE 258 Web Mining and Recommender Systems

23 November, 2021

Assignment 2 Report

**Dataset:**

The dataset that we chose is the Amazon Review Data and more specifically the rating data for Video Games that was linked in Professor McAuley’s set of datasets. We also used the metadata dataset for the Video Games in order to use the ASIN values in order to search for the titles of the products we were interacting with in the dataset. The rating dataset for Video Games from Amazon contained 2,565,349 entries containing the User ID, ASIN, Rating, and Unix Time Value of when the review was submitted. Using this data we were able to see that there were 71,982 unique Video Games and 1,540,618 unique users who submitted reviews. The global average of the Video Game ratings that were provided was 4.022 out of 5 stars, and the global standard deviation is 1.406. The distribution of ratings is shown in Table 1. The distribution is highly left-skewed, indicating that most ratings given by users are either 4.0 or 5.0. And the most popular video-games-related items are shown in Table 2. Note that even the most popular items have only low counts, meaning that the data are very sparse in this dataset. We also have to take into consideration that the items that are most popular may not be games as well as we can see that some of the items in the list are products that are used in the Video Game.

| Rating | Frequency |
| --- | --- |
| 1.0 | 0.121578 |
| 2.0 | 0.055093 |
| 3.0 | 0.082775 |
| 4.0 | 0.160763 |
| 5.0 | 0.579791 |

Table 1

| Count | Title |
| --- | --- |
| 7630 | Diablo III |
| 6462 | Redragon M601 Wired Gaming Mouse, Ergonomic, Programmable 6 Buttons, 3200 DPI with Red LED Mouse for Windows PC Games - Black |
| 5135 | Playstation Plus: 3 Month Membership [Digital Code] |
| 4359 | HAVIT HV-MS672 3200DPI Wired Mouse, 4 Adjustable DPI Levels, 800/1200/2400/3200DPI, 7 Circular &amp; Breathing LED Light, 6 Buttons (Black)(Updated Version) |
| 3962 | HyperX Cloud Gaming Headset for PC, Xbox One&sup1;, PS4, PS4 PRO, Xbox One S&sup1;, Nintendo Switch (KHX-H3CL/WR) - Black |
| 3960 | Xbox 360 Wireless Controller - Glossy Black |
| 3930 | StarCraft II: Wings of Liberty |
| 3634 | Wii Fit Game with Balance Board |
| 3520 | Logitech G602 Lag-Free Wireless Gaming Mouse &ndash; 11 Programmable Buttons, Up to 2500 DPI |
| 3345 | Xbox 360 LIVE 1600 Points |

Table 2

**Predictive Task:**

For the predictive task we are trying to take a look at the video games dataset and predict the accuracy of recommending new items to the user. We begin by creating a negative dataset of video games the user has not purchased or reviewed and trying to see what item we can recommend to the user that they have not purchased. We first shuffled the dataset and partitioned it into a training set with 750,000 rows, a validation set with 250,000, and a test set with 1,565,349 rows. After creating the negative sets, the training set has 1,500,000 rows, the validation set has 500,000 rows, and the test set has 3,130,698 rows.

The baseline model we used makes use of popular items. First we sort all of the unique items in the order of the number of interactions between the item and all of the users. And we define that the top 50% of the items in terms of number of interactions as popular items. If an item is popular, the model predicts positively no matter who the user is, otherwise negatively. The way we assess the performance of a model is that, we calculate the accuracy of the predictions of the model, and we also take runtime into account. The baseline model has an accuracy of 70.578% and a runtime of approximately 0.37 seconds. We trained two other models using the training set and validated the performances of the models on the validation set. They utilize similar popularity thresholds along with Jaccard Similarity Scores, , which takes 1 or 0 based on whether A and B overlap each other, to make recommendations. Lastly, we selected the model that performs the best on the validation set and used it to predict the test set.

In terms of the features that we used, the Rating Dataset that was provided to use neatly organized the User ID, ASIN, and ratings for the reviews which we then used a DataFrame in order to organize and use throughout our code. We also use other data structures to hold information such as Dictionary that maps users to the items they reviewed and also items reviewed by which users. For the metadata, we created a dictionary that held the ASIN as the key and the Title of the item as the value. This allowed us to do quick lookups of items when we were showing lists of recommendations to the user. In terms of creating the negative dataset, we needed to organize the data to track which user item combinations we had seen in our dataset and then create new user item entries that we had not seen in the past. This allowed us to create our negative dataset which we then used to strengthen our model.

**Model:**

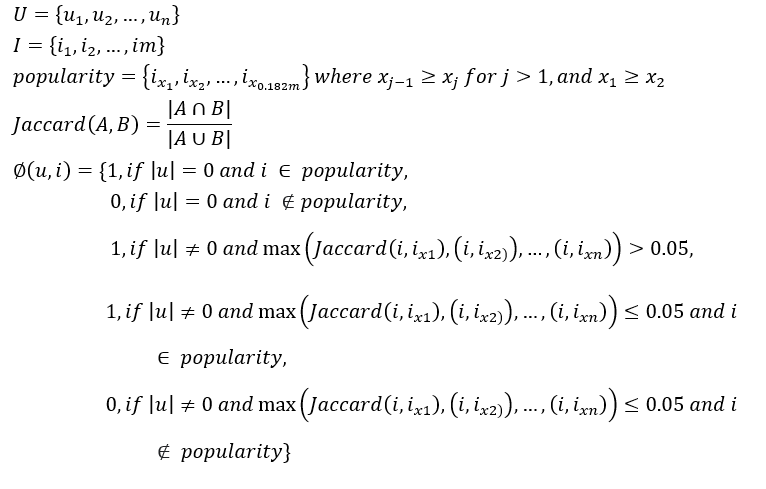
(i)

The first model is an optimized version of the baseline model, and we just aim to find a better threshold than 50%. We brute-forced the threshold for being considered popular and found that the top 17.241% items gives us the best performance on the validation set. This approach gives 264,544 negative predictions, 235,456 positive predictions, and 402,156 out of the predictions are correct. In total, the model has an accuracy of 80.431% on the validation set. And the runtime of this model is approximately 0.31 seconds, which is extremely fast.

(ii)

The second model combines the first model with Jaccard Similarity. On top of model 1, this model predicts positive if an item has a high Jaccard similarity with another item that that user has already purchased with before. If a user has never interacted with any item, we treat it as a cold-start problem, and predict positively if the item is popular (same threshold as we used in model 1). Otherwise, we calculate the Jaccard similarity of all of the items that that user has interacted with and keep the maximum. If this maximum is larger than the threshold, the model predicts positively. If the maximum doesn’t pass the threshold, the model makes its predictions again using the same approach it used in the first step. We optimized the thresholds and found that when the threshold for defining popular items is the top 18.182% items, and the threshold for Jaccard similarity is 0.05, the model has the best performance on the validation set. This model returns 262,757 negative predictions, 237,243 positive predictions, and 403,441 out of the predictions are correct. In total, the model has an accuracy of 80.688% on the validation set. The runtime of this model is approximately 10.5 seconds.

Since model 2 has a better performance on the validation set, and the runtime is not terribly bad, we use it on the test set. We train the model with both the training set and the validation set, and use the same parameters optimized by the validation set. In return, The model gives 1,566,426 negative predictions, 1,564,272 positive predictions, and 402,156 out of the predictions are correct. In total, the model has an accuracy of 84.598% on the test set. To optimize the model that we had chosen, we would most likely need to ensure the thresholds we set are maximized allowing us to get the best performance out of our model. Since the dataset is quite large, it did take much longer to process all of the data compared to our past homeworks. For this we needed to let the code run for quite some time due to the fact that we were scaling up by quite a lot, especially since we added more entries through the negative dataset. Below shows the mathematical equation for the model we ended up using.



Of course, we didn’t get the correct models at first. Initially, we wanted to set ratings as our predictive task, and used gridsearch along with the surprise model. However, since the dataset is large, and the surprise model doesn’t run very fast, it took the machine more than 10 hours to run just a single code block. So we ended up not predicting ratings. When we were using the current models, we also tried brute-forcing with the wrong combinations of thresholds. For example, I first tried the range from 70% to 30% for popular items, but it didn’t give us the global maximum since the range is far from the correct answer.

In hindsight, we could’ve used grid search rather than brute force, which returns the result much more efficiently. We can also consider adding more features, for example “unix\_time” to our chosen model, to potentially give better predictions.

**Literature:**

For our project we decided to use the Amazon rating dataset specifically pertaining to Video Games. This dataset is publicly available through the Professor’s own website which has amassed many different types of datasets. There are multiple categories of data but we went with the Video Game category and looked at the Rating Only dataset along with the Metadata dataset in order to simplify our model but also be able to retrieve the necessary Metadata such as the Title. This allowed us to focus on the recommendation model and once we were content we could retrieve more specific information utilizing the ASIN associated with each product. The Professor also talks about it in his own research papers as well.

We began by taking a look at the literature that the Professor provided on his dataset website of the papers that the Professor was the author of. *Ups and Downs: Modeling the Visual Evolution of Fashion* written by the Professor focuses on the recommendation for fashion which changes with the seasons and time. This makes recommendations for fashion much more difficult as the recommendations do not stay constant. By utilizing deep Convolutional Neural Networks and temporal dynamics, the authors were able to achieve a recommender system that was capable of adjusting it’s recommendations based on current fashion trends as well as other factors that may come into play. I*mage-based Recommendations on Styles and Substitutes* is another paper written by the Professor which takes a look at recommendations for fashion based off of existing outfits. This is another paper related to fashion in which recommending other articles of clothing can either ruin the outfit or add to it. The authors were able to use data from Amazon in order to retrieve similarly categorized clothing in order to train their model. What they found was that recommendations were able to complement what was already in use which apparently was a first in this field of research. Along with the papers the Professor had provided we also took a look at some other research papers pertaining to the prediction of different items based off of ratings collected in the past.

The first paper we took a look at was *Amazon.com Recommendations: Item-to-Item Collaborative Filtering* which showed a similar approach to our model which is called “Item-to-Item Collaborative Filtering”. The paper begins by talking about other methods that were used in the past including, traditional collaborative filtering, cluster models, and search-based methods. These methods relied on other techniques such as mathematical equations, partitioning of datasets into smaller subsets, or even using algorithms that would explore similar items to what the user had already interacted with. With the item-to-item collaborative filtering, it connects each of the user’s purchased items which they have also rated, then links those similar items they have not purchased and appends them into a recommendation list. Based on the purchase history of other customers, the model can suggest recommendations that correlate to the data that Amazon had. With this technique, Amazon was able to scale their recommender system to their platform and with more data being collected from customers on a daily basis, the model would become significantly stronger with its predictions. The paper is quite dated but the techniques that are mentioned are still relevant which is why we incorporated the techniques into our Assignment 2.

On the other hand, *Learning a Joint Search and Recommendation Model from User-Item Interactions* shows not just user behaviors to predict future user-item interactions but also includes natural language analysis for further retrieval purposes. In other words, the Joint Search and Recommendation framework (JSR) model optimizes simultaneously both user-item recommendation objective and item text reconstruction objective. This can explain and retrieve another interaction that existing user-item or item-item recommendation models could not. To fulfill the item text reconstruction objective, the model learns the item description and maps it into a natural language space, using those descriptions for item retrieval. In this process, the model uses the unigram language model for representing items and the softmax function for estimating. This paper also considers item representation that our work has not been able to look at, by using an ngram approach as well.

Modern state-of-the-art methods utilize other techniques and technology in order to improve on past implementations in order to increase the accuracy of the predictions that they generate. In the paper *Customer Reviews Analysis With Deep Neural Networks for E-Commerce Recommender Systems* the authors talk about utilizing deep neural networks in order to incorporate the text that can be found in the review data of customers when reviewing products found on Amazon. This required the team to understand Natural Language processing and create a model that utilized Latent Dirichlet Allocation to extract attributes related to each product category. This would only create a sparse matrix that the team could use, so in order to make the matrix more dense they utilized the deep neural networks. Lastly in order to predict the ratings, they used matrix factorization to extract the values. This methodology proved to increase the accuracy of Hidden Topics and Factors (HTF) and Ratings Meet Review (RMR) models. The HTF model looks at the review text with the stochastic topic distribution modeling which can be applied either on users or items while the RMR model is a hybrid model constituted of content-based filtering and collaborative filtering.

The conclusions that we found are similar to what other researchers have stated in the past. Many of the algorithms that were used in these papers follow a similar nature of utilizing user and item data with other aspects that can be found from reviews or ratings. This showcases that the simple data that we used is capable of providing a strong recommendation system that has been used in the real world as well. Rather than overcomplicating certain processes, we can observe that simple techniques of utilizing Jaccard or Popularity Thresholds still result in accuracy that are somewhat reasonable.

**Results:**

Our final model that we decided to go with that used both the Popularity Threshold and the Jaccard similarity together performed quite well resulting in an accuracy of 84.598% on the test set. In order to achieve this accuracy we set the Popularity Threshold to the top 18.182%, and the threshold for Jaccard similarity to 0.05. Compared to the baseline model, that only used the Popularity threshold of top 50%, and the 1st model we tried, the second model we created performed the best. This shows that adding the Jaccard similarity score into the features did improve the overall performance of recommending new items from the Video Game category to users who have not purchased or reviewed the items. When using just the Popularity Threshold we were able to make some improvement but not a significant change in performance. We believe that our model was more successful due to the fact that we tuned the hyperparameters for both the Popularity and Jaccard thresholds which were used when deciding whether to recommend or not recommend a specific item. Overall, our implementation was not sophisticated in the fact that it did not use Machine Learning in the implementation but we were able to improve our accuracy and run our code in very reasonable time, since machine learning models can be extremely time-consuming, which might be infeasible if the data are large.

Works Cited

He, Ruining, and Julian McAuley. “Ups and Downs.” *Proceedings of the 25th International Conference on World Wide Web*, 2016, <https://doi.org/10.1145/2872427.2883037>.

Linden, G., et al. “Amazon.com Recommendations: Item-to-Item Collaborative Filtering.” *IEEE Internet Computing*, vol. 7, no. 1, 2003, pp. 76–80., <https://doi.org/10.1109/mic.2003.1167344>.

Maleki Shoja, Babak, and Nasseh Tabrizi. “Customer Reviews Analysis with Deep Neural Networks for e-Commerce Recommender Systems.” *IEEE Access*, vol. 7, 2019, pp. 119121–119130., <https://doi.org/10.1109/access.2019.2937518>.

McAuley, Julian, et al. “Image-Based Recommendations on Styles and Substitutes.” *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2015, <https://doi.org/10.1145/2766462.2767755>.

Ni, Jianmo, et al. “Justifying Recommendations Using Distantly-Labeled Reviews and Fine-Grained Aspects.” *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, <https://doi.org/10.18653/v1/d19-1018>.

Zamani, Hamed, and W. Bruce Croft. “Learning a Joint Search and Recommendation Model from User-Item Interactions.” *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, <https://doi.org/10.1145/3336191.3371818>.