COMS 4995 - Applied Machine Learning
Recommendation System for Amazon (Team 29)

Ke Li(kl3352), Parth Gupta(pg2677), Senqi Zhang(sz3016), Vishweshwar Tyagi(vt2353), Xuanhao Wu (xw2744)

***Abstract* – Recommendation Systems have become a crucial factor in driving revenues of tech giants such as Amazon and Netflix. They help reduce the cost of finding and selecting items in an online shopping environment by accurately predicting whether a particular user would prefer an item or not. In this project, we set forward to build a recommendation system of our own using Content-based filtering and Collaborative filtering. Moreover, we calculate the performance of our model using machine learning models.**

I. INTRODUCTION

We use the video games category under the Amazon Review Data (2018), which has over 2.5 million reviews and 84,819 observations in the metadata dataset. It consists of two datasets, the review dataset consisting of information from each review and the metadata dataset consisting of information about each product. These two are linked together by 'asin', which denotes the product's unique ID. We tried several approaches based on content-based filtering and collaborative filtering to recommend video games to users.

II. DATA EXPLORATION

A. Initial Data Cleaning and Processing

After initial exploration, we found out that there were duplicates in the meta dataset. So, we removed them. Some product IDs in the review dataset were absent from the meta dataset. We got rid of rows from the review dataset containing such product IDs. We also merged the review and meta dataset using a left join on 'asin'. Doing so, we lost product IDs present in the meta dataset that were absent from the review dataset. It is reasonable since no one reviewed these and they won't add too much information to our system. In order to draw meaningful insights and overcome computational limitations, we obtained a 10-core subset from the merged data. A k-core subset ensures that each product is reviewed at least k times as well as each reviewer has provided at least k reviews, which avoids the data sparsity and reduces the number of reviews from 2,565,349 to 126,703.

B. Missing Values

Table 1. Table of Missing Values

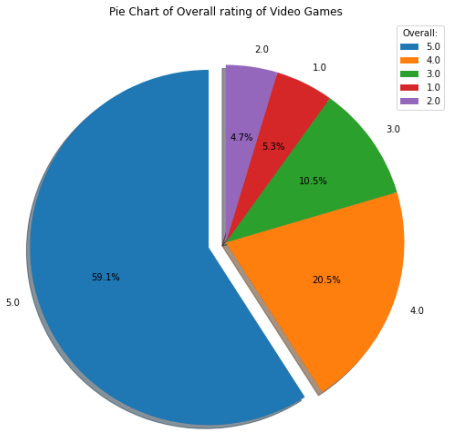
Feature Names	Proportion(%)
overall	0.0000
verified	0.0000
reviewTime	0.0000
reviewerID	0.0000
asin	0.0000
main_cat	0.0000
title	0.0000
summary	0.0174
reviewText	0.0268
reviewerName	0.0331
brand	0.0331
category	0.2352
rank	1.0489
description	1.1389
also_view	1.6156
also_view	4.9415
feature	6.2950
style	32.1366
vote	73.7433
price	90.2867
date	97.3600
image	99.3725
details	99.6859
similar_item	99.9108

All features that have over 40% of missing data will be dropped, except for "vote" in which we will replace the NaN values with 0. Other features such as "imageURL" and "unixReviewTime" have a

very low number of missing values, but we also drop them as we believe they will not contribute to the model.

C. Distribution of Overall Ratings

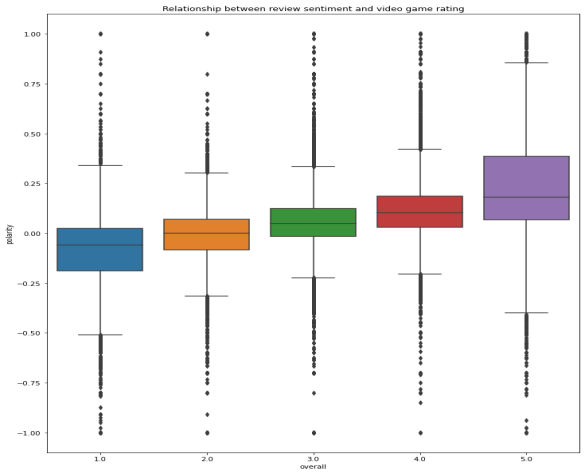
Fig 1. Pie Chart of Overall Ratings



We explored the distribution of overall ratings for the products. Most of the products have ratings of 4 or 5, and very few have ratings less than 3.

D. Sentiment Analysis

Fig 2. Boxplots of Sentiment Score



We also perform sentiment analysis for each review in our dataset, and there is an apparent increase in sentiment score as the rating increases. Though we see some outliers that do not make much sense, we believe that the sentiment score will add value to our recommendation. For the content-based model, we performed many NLP techniques to standardize the text data, and it will be elaborated more in later sections.

III. MODEL APPROACHES

We used two approaches that are commonly used in recommender systems: collaborative filtering and content-based filtering.

A. Collaborative Filtering

In collaborative filtering, we create a user-item matrix and recommend items that similar users have already interacted with. Then estimate the user's ratings based on ratings of other similar users.

Here, we have created a rating matrix R where R[i][j] represents the overall rating that ith user gave to the jth item. R[i][j]. As for the recommendations, each user will have different products recommended to them as they are inferred based on the ratings provided by similar users. We have used different algorithms to predict the value of R[i][j] if the ith user has not interacted with the jth item before. For the implementation part, we have used the surprise library. We have used different algorithms like the KNN model, SVD model, and many more.

The KNN model uses cosine similarity or Pearson's correlation to find similar users to see the neighbors. In the case of SVD, each user will have different products recommended to them as they are inferred by filling out missing entries in the matrix during matrix factorization using SVD.

B. Content-based Filtering

Content-based filtering compares the similarity between two items based on specific metrics and recommends the most similar items to an item that the user previously liked. In our case, we recommend video games that have high similarity to the video games that one user has given a positive review. Unlike collaborative filtering, it creates the user profile by characterizing the user based on item features without comparing it to other users' preferences. In other words, it does not utilize any information about other users and hence does not have the cold start problem. The method of content-based filtering approach can be summarized into the following steps:

1. Item representation: The metadata and review information of each video game, including title, description, reviews, and summary of reviews, are extracted to represent the item.
2. User profile representation: User profiles are generated based on a user's review history. We only select video games that the user has

previously reviewed and given a high rating, as we do not want to recommend games similar to the ones that the user dislikes.

3. Recommendation: A list of video games is selected and recommended to the user based on the similarity between the games that a user likes and the games that a user is most likely interested in.

We implemented two content-based filtering recommender systems, one using only video games' metadata, including description, category, brand, and feature columns. The other used metadata and the review data, including reviews texts and their summary. We would like to see how the review data for each item is helping the system to recommend similar items to a user.

For each system, we preprocessed the text data by the following steps: (1) removing all punctuations, (2) removing all stopwords, (3) lemmatizing the text. Then we applied two vectorization techniques, bags of words and TF-IDF, to calculate the cosine similarity we used to represent the similarity between two games. For both bags of words and TF-IDF, we use unigram and bigram to also look at two words at a time. After calculating a pairwise cosine similarity matrix for all the video games in our dataset, we wrote a function to find the top 30 most similar video games based on the cosine similarity scores.

After that, we kept only reviews with a 5.0 rating and aggregated products by users. After constructing the list, we were able to recommend items that are similar to each user's preference.

IV. DISCUSSION AND RESULTS

In this section, we will discuss the main results and insights we gained from the models introduced in the previous section.

A. Collaborative Filtering

Table 2. Results of Collaborative Filtering

Algorithm	Test RMSE	Test Time
SVD	**0.9467**	0.2782
BaselineOnly	0.9768	0.2506
KNNBaseline	0.9957	2.9382
KNNWithZScore	0.9959	2.7282
CoClustering	1.0019	**0.2437**
NMF	1.0295	0.2499
KNNBasic	1.0952	2.3344

For example, for the user id A2MNJFQXCLMKT8, SVD Model recommends:

1) Tomb Raider: Definitive Edition - PlayStation 4
2) Assassin's Creed IV Black Flag - PlayStation 4
3) DualShock 4 Wireless Controller for PlayStation 4 - 20th Anniversary Edition4.
4) Resident Evil 5 - Standard Edition - PlayStation 4
5) Batman: Return to Arkham - PS4 Standard Edition

KNN model recommends:

1) Batman: Return to Arkham - PS4 Standard Edition
2) Ortz PS4 Vertical Stand with Cooling Fan [Dual Charger Ports] Premium Quality Controller
3) Resident Evil 5 - Standard Edition - PlayStation 4
4) Battlefield 4 - PlayStation 4
5) DualShock 4 Wireless Controller for PlayStation 4 - 20th Anniversary Edition

B. Content-based Filtering

We were able to give recommendations at both the item and user levels. For example, if a user likes "Legend of Zelda Box Set Prima Official Game Guide", our recommender system using TF-IDF with reviews method will return the following top 3 recommendations:

1) The Legend of Zelda: Ocarina of Time
2) The Legend of Zelda: Twilight Princess
3) The Legend of Zelda: Spirit Tracks

Even though it is mostly domain knowledge to determine the model's performance, we tried some methods to evaluate the performance roughly. We utilized "also_buy" and "also_view" features by calculating the overlap percentage between the recommendation set and those two sets, respectively. Then we tried to show the overall review sentiment for products we recommended, and they all presented to be positive. In the table below, it is clear to see that TF-IDF performs better than bag-of-words, and adding the review data significantly increases the overlap.

Table 3: Effect of Features

Vectorization	Features	Also Buy	Also View
Bag-of-words	metadata + reviewText	0.021	0.012
TF-IDF	metadata	0.132	0.141
TF-IDF	metadata + review summary	0.153	0.161
TF-IDF	metadata + reviewText	**0.164**	**0.170**