Hybrid Steam Video Game Recommendation

CMPE 256 - Large Scale Analytics -SPRING 2020

By:

Maahi Chatterjee[013721552]

Ranjana Ajayakumar [012528217]

Shivani Shivanand Suryawanshi[014490944]

Chapter 1. Introduction

This project is a Hybrid Video Games Recommendation System trained using the data from the "Steam Video Game Bundle Data". More information about the data sets have been described in the subsequent sections.

The system returns top 10 recommended video games based on user profiles. The user profile includes the ratings they gave to the video games they played and the item profile includes metadata like *genres, release date* and *tags*. The main objective of the project is to understand how video game data can be used to develop a user profile based recommendation system and maximize the use of information provided in the dataset. In the project, we are taking a user and an item they have watched to recommend similar items for that user. The item similarity is calculated using cosine similarity and the ratings are predicted using SVD and Deep AutoEncoder.

Since explicit ratings were not given for the video games played by any user, we performed *Sentiment Analysis* on the textual reviews provided in the database and classified them into ratings in the range of 0-5. These ratings are used to train our Latent Factor Models.

Chapter 2. System Design & Implementation

2.1 Algorithms considered and selected, system design and concepts implemented

The *Steam Video Game* dataset is a fairly large dataset intended to include video game bundles as a part of its dataset domain. More information about the dataset is mentioned in section 3.1.

The dataset did not contain range specific ratings for items (video games in our case). Therefore, we generate ratings by performing Sentiment Analysis on the items reviewed by users. Since not every user that played a game reviewed it, we only considered user-item pairs on the basis of games that were reviewed by one or the other user. In ideal situations, we can expect explicit numerical ratings for items by users.

2.1.1 Sentiment Analysis

Since our database did not contain user ratings, and our project objective was to explore the possibilities of implementing different concepts and understanding the ways in which a game recommendation system can be implemented, we chose to perform *Sentiment Analysis* on the user reviews to generate our ratings for our dataset.

The subsequent paragraphs explain the steps performed in Sentiment Analysis:

- ***Removing the stop-words:***

  Stop-words are irrelevant words like a ,an, and are as but they occur very frequently in every sentence. One of the main step is to clean the sentence from such words. NLTK library has a list of these stop words and we can import this package for this purpose. We can import from nltk by using the following code

  import nltk

  from nltk.corpus import stopwords.

- ***Removing symbols from words:***

  Most of the text includes images of heart in their reviews. In order for the algorithm to understand such words it is important to convert such symbols to words. In our algorithm, we have replaced this heart symbol from the entire dataset to the word 'Love'. The following figure illustrates this objective

```
In [18]: df.iloc[16].review

Out[18]: '♥♥♥♥ me where to start..-Made Mad Base-Got Red Moon and butt ♥♥♥♥ed11/10 would be butt ♥♥♥♥ed again..christ.Loving t
         his game so far..'
```

```
In [12]: # After replacing the symbols
         df.iloc[16].review

Out[12]: 'LOVE LOVE LOVE LOVE  me where to start..-Made Mad Base-Got Red Moon and butt LOVE LOVE LOVE LOVE ed11/10 would be bu
         tt LOVE LOVE LOVE LOVE ed again..christ.Loving this game so far..'
```

- ***Removing Null values:***

  The null values are identified and dropped from the dataframe using the pandas dropna() , notnull() and isnull() feature.

- ***Removing negation from sentences:***

The user reviews contain words like hasn't ,doesn't, isn't etc. In order to maintain a uniform structure we need to convert those words to standard texts.

- ***Tokenizing and lemmatizing:***

Function to tokenize is designed in this system which essentially breaks down a sentence or a paragraph into smaller words. Similar words among these are grouped together to form aunque word which is similar in meaning. This process is called lemmatizing.

The review text from the user is extracted and with the help of Vader Analyzer and we are classifying the sentiment to positive, neutral and negative review. The resulting value from the vader sentiment analyser is on a scale between 0 and 1. In our algorithm we first identify which class does the review belong to and based on this vader analyzer score we are assigning a 'SentimentScore' which ranges from 0-5. This revised dataframe is later used for recommendation purposes.

i7]:

	User ID	Item ID	Item Name	RecommendItem	Reviews	Total Item Playtime	SentimentScore	Emoji
4621	crazyclerkm	730	Counter-Strike: Global Offensive	True	k	142917967	0	
2267	76561197995338574	730	Counter-Strike: Global Offensive	True	yer (buy game, much fun)	142917967	3	

2.1.2 Latent Factor Models for Collaborative Filtering

In this approach, user_item matrix is decomposed into a smaller matrix which is called Matrix Factorization. The objective of these models is to find latent factors and reduce the dimensions. The rating value Rij is the data associated with each user i and item j. These ratings are collected in matrix called utility matrix Rij, in which row i represents the list of items for user i while each column j lists all the users who rated item j. The value of We used following latent factor models for rating prediction:

1. SVD - The singular value decomposition (SVD) provides a way to factorize a matrix,into singular vectors and singular values. The SVD allows us to discover some of the same kind of information as the eigen decomposition.
2. Deep AutoEncoder - It is composed of two symmetrical deep networks. First half of the network represents encoding, and the second half represents decoding. Each half is composed of four or five shallow layers.

2.1.3 Content Based Recommendation System

A Content-Based Filtering System leverages the content of items in the dataset to recommend similar items. For our project, we have created a *Hybrid Recommendation System* that first finds similarity between all pairs of items. The item similarity matrix is generated by finding *cosine similarity* between pairs of items. We use cosine similarity because our metadata dump is represented by word counts. An article mentioned in [1] was an excellent resource for us to understand similarity better. Our hybrid system then uses the top 100 similar items and a user's already rated items, to generate user specific recommendations (SVD as described above).

The content used to compute similarity included the item *genres, release date and tags* associated with the item. We deliberated using *publisher* as a feature as well, but due to the low repetition frequency of 'publisher' values, we decided to drop it from our metadata dump. In a richer video game dataset, *publishers* would constitute an important feature for similarity calculation.

The similarity derived from the item metadata is calculated using a representation of every item in the form of a ***TF-IDF vector*** *(Term Frequency-Inverse Document Frequency)*. This vector helps to evaluate the importance of different words in a document.

2.2 Visualization screenshots

Visualization is very important to understand the data especially in situations which involves textual data. The following (Fig 1)is a visualization displaying the most frequent words in user reviews that contributed in our Sentiment Analysis algorithm. Word cloud is a feature which represents the text data visually. The most frequent data is highlighted in bold and large size. The chart in Fig 2 shows the binary distribution of user reviews that show whether a user would recommend an item or not.

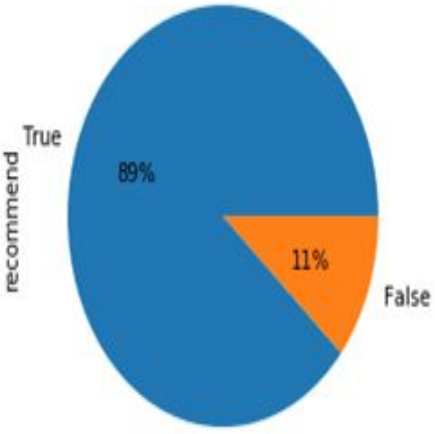

Fig 1 **Fig 2**

The frequency distribution of top 20 words in item reviews is plotted in the bar graph given below:

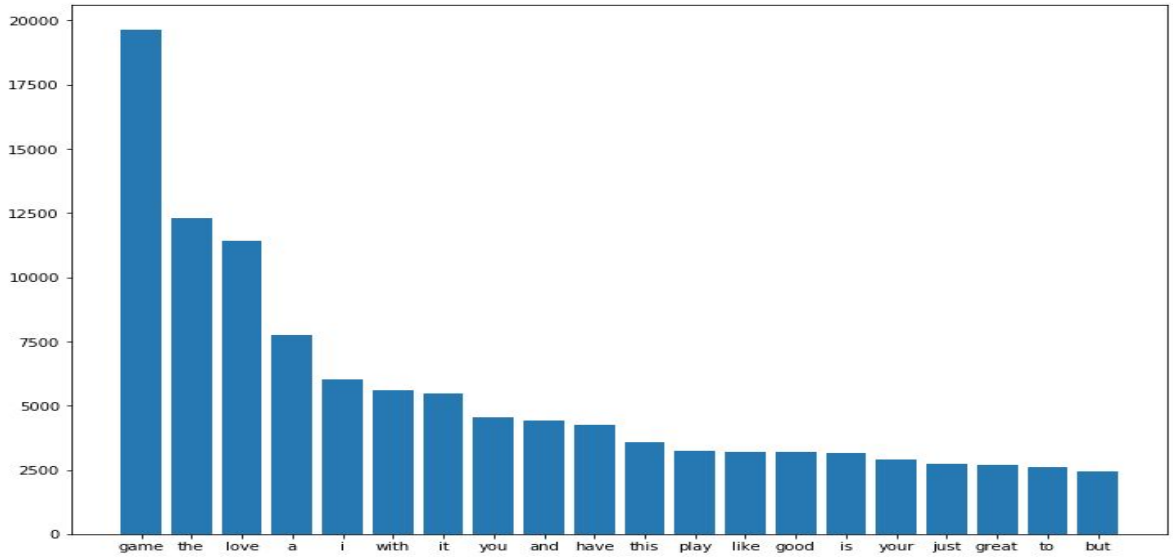

Fig 3

Fig 4 shows the Word cloud representation of the words in the Item Metadata dataframe and Fig 5 shows the rating distribution generated from Sentiment Analysis and used to train the SVD model.

Fig 4

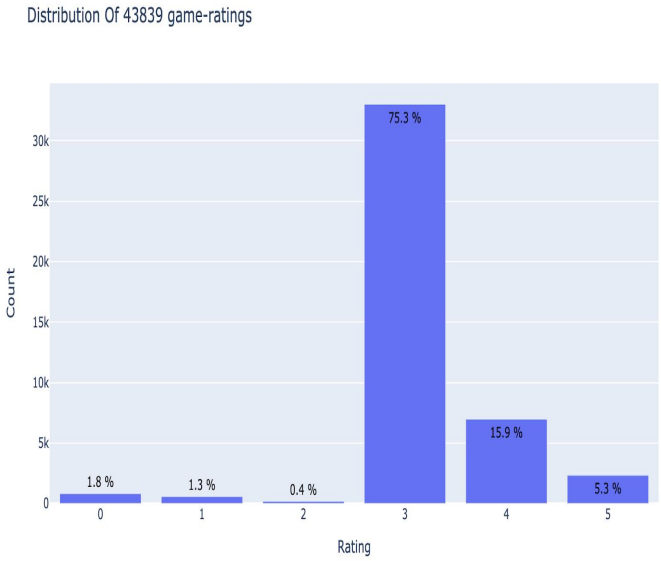

Fig 5

Chapter 3. Experiments and Proof of concept evaluation

3.1 Dataset

The steam video game and bundle data were obtained from the recommendation dataset provided by the UCSD research lab.

Link: https://cseweb.ucsd.edu/~jmcauley/datasets.html

The dataset contains 4 files with the following keys (keys in green are included in our recommendation system):

- Australian_users_items.json : *user_id, items_count, steam_id, user_url, item_id, item_name, playtime_forever, playtime_2weeks.* (*playtime_forever* is used to calculate the total time that any game has been played by all users)
- Steam_games.json: *publisher, generes, app_name, title, url, release_date, tags, discounted_price, reviews_url, price, early_access, id, developer*
- Australian_user_reviews.json: user_id, user_url, reviews_funny, reviews_posted, reviews_lastEdited, reviews_item_id, reviews_helpful, recommend, review
- Bundle_data.json: This file contains data about pre-made bundles. Since we used the dataset to recommend game items, we did not use this file. We deliberated on creating our own bundles to recommend as enough data was not available for the pre-made bundles. However,

we concluded that recommending 3 bundles of 5 items each is equivalent to recommending top 15 items.

Most of the data in the dataset were either url links or unpopular keys. So we decided not to use boolean values (as most of our computations rely on text analysis and boolean values don't contribute much). The different files were merged and processed in multiple ways to make it useful for our system. Kindly refer to the appendix attached at the end of the report which contains screenshots of the various data frames created and used.

This dataset contains 7,793,069 user reviews, 2,567,538 Users and 15,474 items which represents the type of games. In this project we have focused on Australian user reviews Australian user items and Bundle data for the recommendation system. In the Australian user review dataset around 17939 users have given reviews and in the Australian user item dataset contains the game name and playtime of 85,876 users. After processing the data and removing incomplete data, our final algorithms were run on 43839 user-item ratings (SVD and Sentiment Analysis), 22857 items (Content Based Filtering using cosine similarity) and 2789 games that have been rated.

3.2 Data preprocessing decisions

The data processing step mostly included data merging and cleaning rather than any kind of computational preprocessessing. Since our feature set was distributed across different files, our first step was to fetch and collect the right features in the respective algorithm required data frames. We have deliminated the steps in the subsequent paragraphs. This can also be considered as a step wise walk through of the data cleaning steps as performed in the submitted notebook.

- We extracted *user_id, tem_id, item_name* and *playtime_forever* from the australian_users_items file (*dataframeUserItem)* . This file contains information about games users played. *Playtime_forever* is an attribute showing the duration a user played the game for. Item wise summation on the basis of this feature was used to calculate the popularity of items. However, since this feature information is only available for some games and not all games were played by users, our dataset was becoming very small. Our intention behind this step was to include only popular items for recommendation. But we concluded that other metrics are needed to calculate item popularity properly which were not provided in the dataset.
- We extracted *user_id*, *item_id*, *recommend* (boolean value) and reviews from australian_users_reviews file (*dataframeUserItemReviews*). Then we merged the 2 dataframes described on the basis of *item_id* and *user_id* to only keep those user-item pairs which have been reviewed (check description of dataframe

dataframeUserItemReviewMerged in the Appendix). A copy of this merged dataframe (*dataframeUserItemReviewMerged1)* is used to perform Sentiment Analysis.

- As a next step, we calculated item based summation of feature *playtime_forever* to calculate the total play time for the games in the *dataframeUserItemReviewMerged* dataframe and created dataframe *dataframePlaytime.* After this step, we are left with 2789 unique games which have been both played as well as reviewed by users.
- Then we created 1 dataframe just containing the *item_id* and *item_name* features of the games that have been both played and reviewed by the user. This is stored in the dataframe *dataframeItemOnly.*
- Next we extract *item_id, title, publisher, genres, release_date, price* and *tags* from the file steam_bundles and store it in dataframe *dataframeItem.* We clean the data to convert *genres* and *tags* to string from array so that it becomes compatible to be fed into the tf-idf vectorizer for computing the item similarity. We also extract only the year from our release_date to use it as a similarity measure. This dataframe is used to return item details for the items recommended by the algorithm.
- We dump all the metadata (*genres, release_year* and *tags)* together as a string in a column of the dataframe *dataframeMetadata*, which is the dataframe that holds our item profile. Therefore, finally this dataframe contains the following columns: *Item ID, Item Name* and *Metadata Description.*
- We merged *dataframeMetadata* with *dataframeItemOnly* (which holds total playtime of games) to keep only the games which have a playtime greater than 50. These values are stored in dataframe *dataframeItemMergedMetadata* but is not used for any further computation.
- Finally, we drop duplicates and empty values from our *dataframeUserItemReviewMerged* dataframe and make a copy of this dataframe to perform Sentiment Analysis which results is the dataframe *dataframeSVD*.
- *dataframeSVD* contains user-item pairs and the sentiment score generated from Sentiment Analysis. This dataframe is the training set for performing SVD.
- Next we run the tf-idf vectorizer on the *Metadata Description* column of dataframe *dataframeMetadata*, to get a matrix representation of all the item-item pairs which is then used to compute cosine similarity between the item pairs.
- *dataframeAutoEncoder.csv* file is used as input for autoencoders. Each userID and itemID values are replaced by unique integers to form a utility matrix (num_users x num_items).

We have included the data visualizations as well as recommendation functions to use just our *User Based Collaborative Filtering* and *Content Based Filtering* algorithms as comment right after the algorithm definitions are over in the notebook. To get a better understanding of the dataframes created, kindly check the appendix.

3.3 Evaluation methodology followed

A lot of discussion was put into the steps to be taken to make the best use of the data that was available to us. For predicting the user ratings, we first evaluated the possibility of using Logistic Regression to predict the ratings. The dataset was split into training and test sets and the model was trained to predict the sentiment score. We achieved 86% accuracy, and the breakdown is represented below in the heatmap in Fig 6:

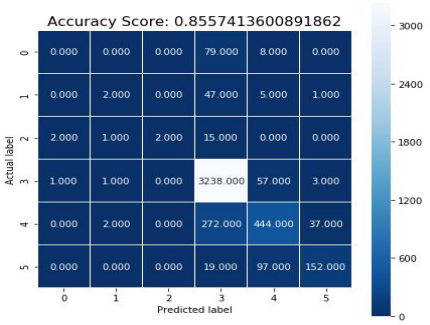

Fig 6

We however decided to go for Latent Factor Models. First we selected SVD because of 2 reasons:

- Our user-item-rating dataset was a sparse matrix and SVD yields better results in such scenarios.
- This accuracy was achieved by splitting data from the dataset into training and testing dataset and the ratings were calculated using Sentiment Analysis and were not implicit ratings.

We also evaluated our dataset using gridsearch across several other algorithms. The results are displayed in Fig 7.

Algorithm	test_rmse	fit_time	test_time
SVD	0.746156	9.491659	0.593278
SVDpp	0.746263	14.720017	5.986150
CoClustering	0.837498	7.275580	0.294337
SlopeOne	0.872218	1.262877	0.549246
NormalPredictor	1.053870	0.235737	0.396866

Fig 7

Later we decided to implement Deep AutoEncoder. We decide to implement deep learning autoencoder due to the following reasons:

- Dataset is very huge and the deep learning algorithm performs very well on huge datasets.
- Autoencoders are widely used in Collaborative Filtering for recommendation systems. On large datasets, deep autoencoders perform better than SVD.
- As per our expectation, loss of the autoencoder model is less than the SVD. It achieved highest accuracy than any other algorithm. Fig 8 represents the RMSE over train and validation data for 200 epochs

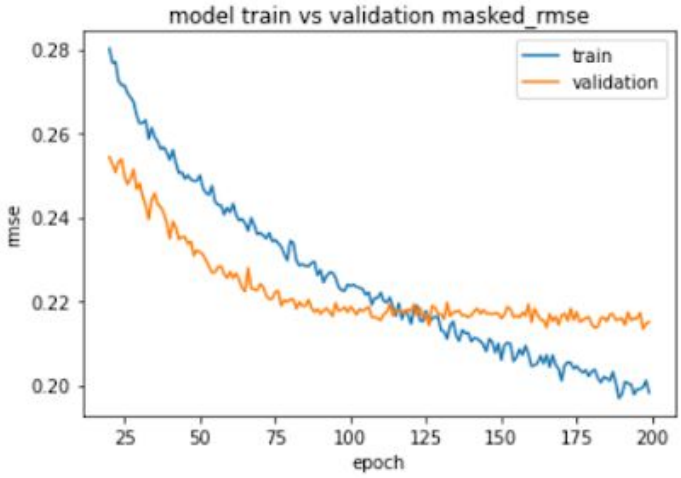

Fig 8

Chapter 4. Discussion & Conclusions

4.1 Decisions made/Things that worked and Difficulties faced

The thought process behind our decisions are as follows:

- We had 4 data files in our dataset - 3 related to individual items and 1 related to bundles. So one of our major decisions was to build an item recommendation system instead of a bundle recommendation system as based on the information available, we concluded that we did not have enough information to build an effective bundle recommendation system.
- The second step was to determine the information that can be used to maximize the data available in the data files. We ended up using the *recommend* and *reviews* keys of the user_item_review data file to perform sentiment analysis and *genres, tags and release year* (extracted from release date) as the metadata to find item similarity.

- We first included the *publisher* feature as a metadata entry as well, however, on the basis of popularity information we researched on the internet, we were not getting appropriate results. We attributed this to the quality of the dataset and included the features that were giving us better recommendations. However, it is important to note that *publisher* is an important aspect for video game recommendation.
- As a part of data cleaning, we only wanted to consider video games which had a total playtime of greater than 1000. However, since data for the time a video game was played wasn't available for every user who played a game, our dataset was becoming too small. So we included the computation in the notebook (available in the git repository) but did not use the data frame to find similarity (Kindly check the comments and heading in the submitted notebook).
- Ratings were generated using Sentiment Analysis and therefore might not reflect the actual rating that a user would have given to a game. Our rating scale was from 0-5 and most of the ratings were concentrated around 3. With a better dataset, we expect to achieve better quality of results.
- We also did not calculate weighted ratings for items by users to remove the bias in the ratings. This was again because explicit ratings were not for user-item pairs and the actual distribution of the ratings could be way different.

4.2 Conclusions

- Our attempt at understanding how a *Hybrid Video Game Recommendation System* was successful as we were able to gain an understanding about how video game data is collected and can be leveraged for creating recommendations.
- We were successfully able to combine and understand 3 important concepts namely, *Sentiment Analysis*, *Latent Factor Models for Collaborative Filtering* and *Metadata based Content Filtering* for item recommendation.
- In ideal situations, with higher computational hardware and data collected from activity tracking of users, this system can be refined to include other metadata like *voice actors* and *actors contributing to motion capture* as well as better segregated information when it comes to genres and tags.
- We were able to evaluate different algorithms for different steps of the system - Euclidean Distance for similarity, Logistic Regression, SVDpp, Co-Clusering etc (using Grid Search) for predicting ratings.

Chapter 5. Task Distribution

Our task distribution for the project was fairly simple. All our decisions including the decision to build an item recommendation system instead of a bundle recommendation system was taken

together. We discussed the data attributes that we included as our feature set and decided on all the preprocessing steps described above together. The next steps were divided as follows:

- Sentiment Analysis was the first step and was performed by Ranjana Ajayakumar.
- Latent Factor Models for Collaborative Filtering was performed by Shivani Shivanand Suryawanshi
- Metadata Content Based Recommendation was performed by Maahi Chatterjee

Chapter 6. Appendix and References

- dataframeUserItem: Dataframe containing user-item details

	User ID	Item ID	Item Name	Playtime
0	76561197970982479	10	Counter-Strike	6
1	76561197970982479	20	Team Fortress Classic	0
2	76561197970982479	30	Day of Defeat	7
3	76561197970982479	40	Deathmatch Classic	0
4	76561197970982479	50	Half-Life: Opposing Force	0

- dataframeUserItemReviews: Dataframe containing user-item reviews

	User ID	Item ID	RecommendItem	Reviews
0	76561197970982479	1250	True	Simple yet with great replayability. In my opi...
1	76561197970982479	22200	True	It's unique and worth a playthrough.
2	76561197970982479	43110	True	Great atmosphere. The gunplay can be a bit chu...
3	js41637	251610	True	I know what you think when you see this title ...
4	js41637	227300	True	For a simple (it's actually not all that simpl...

- dataframeUserItemReviewsMerged: Merged dataframe formed by merging *dataframeUserItem* and *dataframeUserItemReviews*. This dataframe is used for deducting ratings by performing *Sentiment Analysis*

	User ID	Item ID	Item Name	Playtime	RecommendItem	Reviews
6944	wolop	4000	Garry's Mod	642773	True	Like buiding well you might like this
15134	76561198039832932	4000	Garry's Mod	613411	True	man this game RULES THANKS GARRY
18153	tsunamitad	72200	Universe Sandbox	600068	True	What can I say?1 hour of gameplay to explore t...
18864	shinomegami	8500	EVE Online	530882	True	When a player quits EVE and goes to WoW, the a...
30714	ThisIsWhereIGetOff	4000	Garry's Mod	495058	True	Garry's mod, Best mod in this whole dumb world...

- dataframePlaytime: Dataframe containing I*tem ID*, *Item Name* and total *Playtime* for each item

Item ID	Playtime	Item Name
730	142917967	Counter-Strike: Global Offensive
4000	49464882	Garry's Mod
218620	12546425	PAYDAY 2
230410	12321959	Warframe
105600	10320781	Terraria

- dataframeUserItemReviewMergedFinal (User Profile): Contains all the relevant information that constitutes our user profile for our problem space

	User ID	Item ID	Item Name	RecommendItem	Reviews	Total Item Playtime
4621	crazyclerkm	730	Counter-Strike: Global Offensive	True	k	142917967
2267	76561197995338574	730	Counter-Strike: Global Offensive	True	yer (buy game, much fun)	142917967
2243	pirateness	730	Counter-Strike: Global Offensive	True	Formally comming from the community of TF2 i w...	142917967
2244	HolyJw	730	Counter-Strike: Global Offensive	False	after enjoying this game for 600hours i've bee...	142917967
2245	dsmob	730	Counter-Strike: Global Offensive	True	I recommend this game for anyone who has playe...	142917967

- dataframeItemOnly: Dataframe containing only *Item ID* and *Item Name* pairs

	Item ID	Item Name
4621	730	Counter-Strike: Global Offensive
0	4000	Garry's Mod
12467	218620	PAYDAY 2
5138	230410	Warframe
5984	105600	Terraria

- dataframeItem: Item wise mapping with other information about the items

	Item ID	Item Name	Publisher	Release Year	Genres	Tags	Item Price
0	761140	Lost Summoner Kitty	Kotoshiro	2018	, Action , Casual , Indie , Simulation , Stra...	, Strategy , Action , Indie , Casual , Simula...	4.99
1	643980	Ironbound	Making Fun, Inc.	2018	, Free to Play , Indie , RPG , Strategy	, Free to Play , Strategy , Indie , RPG , Car...	Free To Play
2	670290	Real Pool 3D - Poolians	Poolians.com	2017	, Casual , Free to Play , Indie , Simulation ...	, Free to Play , Simulation , Sports , Casual...	Free to Play
3	767400	弹炸人2222	彼岸领域	2017	, Action , Adventure , Casual	, Action , Adventure , Casual	0.99
4	772540	Battle Royale Trainer	Trickjump Games Ltd	2018	, Action , Adventure , Simulation	, Action , Adventure , Simulation , FPS , Sho...	3.99

- dataframeMetadata (Item Profile): Dataframe containing the Metadata dump

Item ID	Item Name	Metadata Description
10	Counter-Strike	, Action , , Action , FPS , Multiplayer , Sh...
1002	Rag Doll Kung Fu	, Indie , , Indie , Fighting , Multiplayer ,...
100400	Silo 2	, Animation & Modeling , , Animation & M...
10090	Call of Duty: World at War	, Action , , Zombies , World War II , FPS , ...
100980	3D-Coat V4.8	, Animation & Modeling , , Animation & M...

- dataframeSVD: Dataframe for SVD training

	User ID	Item ID	SentimentScore
4621	crazyclerkm	730	0
2267	76561197995338574	730	3
2243	pirateness	730	3
2244	HolyJw	730	3
2245	dsmob	730	4

- **References:**
  - [1] https://cmry.github.io/notes/euclidean-v-cosine
  - [2] https://monkeylearn.com/blog/sentiment-analysis-of-product-reviews/
  - [3]https://towardsdatascience.com/predicting-sentiment-of-amazon-product-reviews-6370f466fa73
  - [4]https://stackoverflow.com/questions/265960/best-way-to-strip-punctuation-from-a-string
  - [5]https://medium.com/@shrishekesh/deep-autoencoders-for-movie-recommendations-a-practical-approach-ae539f3473e4
  - [6]https://medium.com/@rabinpoudyal1995/latent-factor-based-method-in-collaborative-filtering-77756a02f675