Recommender system for e-commerce

Jawahar Pinnelli
Rutgers University
jp2141@scarletmail.rutgers.edu

Viswajith Menon
Rutgers University
vm623@scarletmail.rutgers.edu

Sai Pradyumn Shrivastava
Rutgers University
ss4369@scarletmail.rutgers.edu

ABSTRACT

Recommender systems provide significant business value alongside personal value to users. Usage of users in many platforms where they access items presents patterns of sequential dynamics. These resolve to long-term recurrence of usage of similar items. This data affords itself to many data mining approaches through which meaningful inferences and recommendations can be made to users. Several approaches have proliferated over the years in accomplishing these recommendation tasks. In this work, we implement and compare two different but compatible strategies to make item-list recommendations. One employs rating prediction models to recommend most relevant items to users. Another mines sequential patterns in item sequences to recommend items that best fit recent context.

KEYWORDS

Collaborative filtering, Sequential recommendation, Rating prediction

1 INTRODUCTION

Recommender systems employ algorithms to analyze data recorded in users' interaction with items to suggest relevant items to users. As a function of the data in consideration, the task of recommendation could be formulated mathematically in various ways. Some task formulations like rating prediction are particularly useful as models trained on that task can be applied to multiple use-cases in a recommender system. For example, applying rating prediction models on a set of items that a user hasn't interacted with and selecting items with the best predicted ratings lets the system present a list of most relevant items to the user.

Over the last two decades, several shallow and deep learning methods have been studied and implemented in academia and industry to recommendation tasks like these in various domains. They provide a lot of value to users in reducing a large item pool to a small list of relevant items to consider. In doing so, recommender systems have enabled several new business models which use these techniques to market a large catalog to users.

Conference'17, Washington, DC, USA

DOI: 10.1145/nnnnnnn.nnnnnnn

These algorithms can be classified into two broad categories based on the data they use to identify relevant items: collaborative filtering and content-based filtering. Collaborative filtering techniques primarily use user-item interaction and preference data to mine patterns in preferences. They operate on the assumption that data about users' preference levels to items can be compared against data of other users to make predictions about whether a user would have high preference for unknown items. Content-based filtering algorithms, on the other hand, consider data describing the items alongside user's preference for those items, to model which features of items best describe users' preferences. This modeling is used to predict preference for unknown items and make recommendations. It is easy to see how collaborative and content-based techniques differ in their strategies of identifying the most relevant items to users.

In this work, we implement models to perform two recommendation tasks–rating prediction and item recommendation–in e-commerce domain using rating and review data from Amazon. We first implement three models for the rating prediction task: SVD, SVD++, and Co-clustering. We then take a linear combination of their predicted ratings, weighed by the inverse of their error rates, as a unified prediction that incorporates capabilities of all three models. This rating prediction model is used to perform item recommendation, by making a list of unknown items with high predicted ratings. We also implement a transformer-based model Self-Attentive Sequential Recommendation (SASRec) to make an alternative list for the item-recommendation task. While both techniques could be considered as collaborative filtering, their approaches surface different items to users as they are solving different related problems. While rating prediction models focus on general relevance to the user in making item recommendations, sequential transformer-based models focus on identifying the most likely items to continue a user's recent purchases. The latter uses collaborative-filtering to model item-item correlations to suggest items that best continue recent context. We analyze the performance of both the models to compare which strategy is better suited for this recommendation task and domain.

2 RELATED WORK

Recommender systems have been studied extensively over the last few decades. Several papers have been very helpful references. Recommender systems model compatibility between users and items based on historical feedback such as clicks, purchases, and likes. Collaborative filtering methods are summarized in [5]. Matrix factorization [4] has been widely used in many works since the success of Netflix prize. TimeSVD++ [1] presents a time aware modelling of user item relations and

Table 1: Dataset statistics

Range	Mean	Median	Variance	Standard Deviation
0-5	4.22	5.0	1.22	1.23

account for drift in preferences. Sequential recommendation has been studied in [2]. Transformers have been used for sequential recommendation in [3], which served as a reference for our implementation.

3 DATASET DESCRIPTION

For this project we have chosen the 'Cellphones and Accessories' product subsection from the Amazon E-commerce Dataset (https://nijianmo.github.io/amazon/index.html). The dataset includes over 1.2 million reviews, covering products from various brands such as Apple, Samsung, LG, Motorola, and more. The reviews span from May 1996 to July 2018. Each review in the dataset contains information such as the reviewer's ID, the product ID, the review text, the overall rating, and other metadata such as helpfulness votes and review date. Over the years, this dataset has been used for various natural language processing and machine learning tasks, such as sentiment analysis, text classification, and recommendation systems, specifically for the cell phone and accessories domain.

4 THE PROPOSED MODEL

We are implementing models to aid two tasks: Rating prediction and Item recommendation. For the first task–rating prediction–we implement three models with different internal considerations. SVD does dimensionality reduction through latent factor analysis; SVD++ augments the recombination of latent factors with implicit interaction information; Co-clustering builds clusters of users and items to build rating predictions informed by those groupings. In the item recommendation task, a model is needed to present a set of 10 recommended items ordered by relevance to the user. We build models to solve this task using two strategies. First uses the rating prediction models to sort items by predicted relevance to the user and present the top ones. The second, SASRec, performs item prediction directly from an input sequence describing user's recent purchase history. Both approaches use very different strategies to make the item lists and therefore could provide both relevant and contextual recommendations when put together.

4.1 Rating Prediction

We implement three rating prediction algorithms which make different considerations to perform the task, and hybridize their results.

4.1.1 SVD. The Singular Value Decomposition (SVD) algorithm is a popular method for building recommendation systems. SVD is a matrix factorization technique that factors a large matrix into three smaller matrices. This is done in order to uncover the underlying latent factors that explain the observed ratings in the original matrix. Latent factors refer to underlying, unobserved variables or attributes that influence a user's preference for an item.

In a recommendation system, the input matrix is a sparse user-item matrix where each row corresponds to a user and each column corresponds to an item, and the cells represent the user-item interactions. The SVD algorithm factorizes the user-item matrix into three matrices, one representing the users, one representing the product and one that contains singular values that represent the importance of the latent factors in the user-item matrix. The user matrix represents how much each user likes each underlying factor, and the product matrix represents how much each product exhibits each underlying latent factor. The weight matrix contains the singular values that indicate the relative importance of each factor. By multiplying these three matrices, we reconstruct the original sparse user-item matrix and predict missing values; see figure 1.

To make recommendations, we can use the reconstructed matrix to find the top-rated items for a given user or to find similar items based on their factor weights.

$$\hat{r_{ui}} = \mu + b_i + b_u + (q_i)^T p_u$$

$\hat{r}_{ui}$: predicted rating

μ : average rating of user u

b_i : bias in rating of item i

b_u : bias in rating of user u

p_u : user factors

q_i : item factors

A : rating matrix

Figure 1: SVD

4.1.2 SVD++. The SVD++ algorithm is an extension of the Singular Value Decomposition (SVD) algorithm for building recommendation systems. SVD++ takes into account additional information beyond the user-item interactions. The difference here is that it considers the implicit ratings, as described in figure 2.

Like SVD, SVD++ factorizes a sparse user-item matrix into three smaller matrices, representing the user matrix, the item matrix, and the weight matrix. However, SVD++ also includes a fourth matrix. To incorporate implicit feedback data, the algorithm introduces an additional matrix Y that represents the implicit feedback. Each row in the Y matrix represents a user's interactions with the implicit feedback sources. Here, an implicit rating describes the fact that a user 'u' rated an item 'j', regardless of the rating value.

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \left(p_u + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j \right)$$

r_{ui} : predicted rating

μ : average rating of user *u*

b_i : bias in rating of item *i*

b_u : bias in rating of user *u*

Y_j : The new set of item factors that capture implicit ratings.

Figure 2: SVD++

4.1.3 Co-clustering. When a user rates a particular product, there are two entities involved, the product and the user. The co-clustering algorithm tries to group similar users and similar items from the user-item matrix based on their pairwise interactions, as described in figure 3. We have used the built-in co-clustering algorithm provided in the python Surprise package. This algorithm in the Surprise Python package works by simultaneously clustering the rows and columns of the user-item rating matrix using the Expectation-Maximization (EM) algorithm. The algorithm minimizes the squared error between the observed ratings and the predicted ratings.

$$\hat{r}_{ui} = \overline{C_{ui}} + (\mu_u - \overline{C_u}) + (\mu_i - \overline{C_i}),$$

Figure 3: Co-Clustering

4.2 SASRec

The Self-Attentive Sequential Recommendation model (SASRec) is a transformer that models sequential dynamics in users' purchase histories to capture the context of users' recent purchase activities. The attention mechanism enables the model to capture long-term semantics using only relatively few past actions. Essentially, it assigns weights (in the form of an attention vector) to each of the past items, to make a prediction of the next item. It can learn strategies to assign these weights by mining item correlation patterns across the purchase history of all users.

The Amazon review data is structured as tuples of user id, item id, rating, and review time. For each user, the data is transformed into a list of item ids of the items they've purchased, ordered by review time. This gives a sequence of items $S^u = (S_1^u, S_2^u, ..., S_{|S^u|}^u)$, for each user u. The input of the model is $(S_1^u, S_2^u, ..., S_{|S^u|-1}^u)$. The expected output it is trained to is $(S_2^u, S_3^u, ..., S_{|S^u|}^u)$. An additional transformation is to truncate long sequences and pad shorter ones to transform the data of all users to fixed-length sequences.

We create the item embedding matrix $\mathbf{M} \in \mathbb{R}^{|\mathcal{I}| \times d}$, representing each item with d latent factors. Each input sequence is transformed into an embedding matrix $\mathbf{E} \in \mathbb{R}^{n \times d}$, where n is the length of the fixed-length sequences. A positional embedding, $\mathbf{P} \in \mathbb{R}^{n \times d}$, is added to the sequence embedding **E**.

The model employs two self-attention blocks to learn complex item transitions. ReLU is used in aggregation between layers for non-linear patterns in the interactions between different latent dimensions. To avoid overfitting normalization is done on the output of each layer, and dropout is done during training only. The binary cross entropy loss is used as the objective function to optimize using Adam optimizer.

	RMSE	MAE	Precision	Recall	F-Score
SVD	1.1453	0.8608	0.8079	**0.9680**	0.8808
SVD++	1.1433	**0.8454**	0.8181	0.9506	0.8794
Co-clustering	1.2344	0.8545	**0.8605**	0.8451	0.8527
Hybrid	**1.1341**	0.8515	0.8107	0.9666	**0.8818**

Table 2: Rating prediction results

5 EXPERIMENTS

The following section presents the results for all the models discussed earlier, and their accuracy is evaluated using metrics such as RMSE and MAE for rating prediction. Additionally, the accuracy of item recommendation to each user is evaluated using metrics like Precision, Recall, and F-Score for SVD, SVD++ and Co-Clustering while we use NDCG and Hit rate for SASRec.

Table 2 summarizes the results of the experiment conducted on the models (SVD, SVD++, Co-Clustering and hybrid model). Where we could see that Hybrid model has the least error rate (RMSE: 1.1341), leading us to infer that hybrid model has better performance that the rest three in case of rating prediction. While if we consider the three individual rating prediction algorithms, SVD++ has a comparatively better performance. The table above shows that the precision, recall, and F-score ranges are similar across all models. As is often the case in data science, there is a trade-off between precision and recall. Improving one of these metrics typically comes at the expense of the other. The choice of whether to prioritize precision or recall depends on the specific use case. For cases where optimal values are needed, we can use the F-score, which combines both metrics. The F-score increases when the two values are similar. Therefore, in order to obtain a balanced model, we aim to maximize the F-score. See table 3 for prediction samples.

SASRec's performance on the item recommendation task is: NDCG@10: **0.356**, Hit rate@10: **0.555**. It is able to achieve this performance without considering ratings at all. It's consideration of item correlations across different users, and inter-item dependencies within each purchase sequence, and weighing recent context appropriately were enough to surface 55% of relevant items within the next ten items predicted.

6 CONCLUSIONS AND FUTURE WORK

In this project, we used for experimenting and testing a 5-star rating from Amazon – Cellphones and Accessories dataset with 1128437 ratings, 48186 unique products and 157212 unique users; we built the models and methods to make it work with other data sets and can scale for more massive data sets. Experimenting on various models before building the recommendation system was essential for us to learn more about the models and on the other side to have scientific numbers (RMSE and MAE) that convince us which model for recommending the items to the users. Predicting ratings using these different models: SVD, SVD++, Co-Clustering, Hybrid models and SASRec gave us a noticeable conclusion that the hybrid model (SVD, SVD++, and Co-Clustering) has the lowest error. Combining three predicting models made us benefit from their complementary advantages; as a result, it achieved the highest precision for the rating prediction. We considered an alternative item recommendation strategy through Self Attention based Sequential Recommendation model. Refining this model to make it faster and work on the larger counterparts of the data set is one of our main future improvements for this recommendation system as we see promising results on the scarce data and computing resources at our disposal. The SASRec has an advantage over the other methods discussed and we hope to consider this as an addition to the hybrid approach pushing its boundaries even further.

ACKNOWLEDGEMENT

We thank Professor Yongfeng Zhang for the insights provided into data mining at scale in the course, and for the valuable feedback on the project. We also thank the Rutgers University CS department for the compute infrastructure.

REFERENCES

[1] Koren Y Collaborative. 2010. filtering with temporal dynamics [J]. *Commun. ACM* 53 (2010), 89–97.
[2] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).
[3] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 197–206.
[4] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.
[5] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. 2007. Collaborative filtering recommender systems. *The adaptive web: methods and strategies of web personalization* (2007), 291–324.

	SVD	SVD++	Co-Clustering
User ID :A5AL9MYTWU5R9 ProductID:B015FLFC56	True Value: 5 Predicted:4.50	True Value: 5 Predicted:4.52	True Value: 5 Predicted:5
User ID :AWRKVZQD2T9V5 ProductID:B0146G1M9Q	True Value: 5 Predicted:4.50	True Value: 5 Predicted:4.19	True Value: 5 Predicted: 3.43
User ID :A241FL95ZO6DQY ProductID:B00XIJRCOM	True Value: 4 Predicted:4.04	True Value: 4 Predicted:4.18	True Value: 4 Predicted: 3.59

Table 3: Sample predictions