RECOMMENDATION SYSTEM USING PATTERN MINING FOR A STORE

Ankush Gajanan Arudkar
The University of Adelaide
a1845627@student.adelaide.edu.au

Ka Hing Chan
The University of Adelaide
a1867200@student.adelaide.edu.au

Manish Adhikari
The University of Adelaide
a1876371@student.adelaide.edu.au

Contents

1 Executive summary

Many items present in the store are often bought together owing to their inherent relations, and some customers prefer to buy certain items more than other customers. Using the buying preferences of customers we wanted to recommend items to individual customers that caters to their specific needs, which would directly translate to better customer satisfaction and increased sales.

Due to large number of products sold on the online website of store, explorability of an less popular but relevant products by a user becomes difficult, we wanted to reduce this gap by letting users find items that they want to buy, by means of personalized recommendations.

In this report, we propose a system that is capable of generating targeted recommendations for individual users based on their prior buying preferences and the contents in their cart. The system is not only capable of accommodating current transaction load but can be efficiently scaled to a million transactions, with little to no overhead in generating recommendations. Using the proposed method, we were able to predict nearly 23% of the items that were indeed bought by the customers, showing that the system could accurately identify almost a quarter of the items that customers needed and were willing to purchase.

2 Introduction

With a broad range of products available for the customers to choose from, the discovery of less popular items that are relevant to customers becomes increasingly challenging. Various data mining techniques such as association rules mining, content-based recommender systems (Pazzani and Billsus 2007), and collaborative filtering are widely used for recommending items targeted towards individual users. Content-based recommender systems rely on properties of users and items to match compatible items to users which requires detailed and accurate knowledge about both users and items, making them difficult to train and maintain for items with wide ranging attributes. Unlike content-based recommenders, the collaborative filters rely only on interactions between users and items irrespective of the properties of either but perform well when more data about the interactions is available. Moreover, association rules are capable of providing generic recommendations based on the items present in the user basket but lack the personalized touch to them. In this report, we propose a hybrid system that utilizes clustered association rules with user-based collaborative filtering and compare it to conventional collaborative filtering methods.

The objective of our system is to utilize data on past transactions by customers, to predict a rating score, which is indicative of their likelihood of buying the item, for every missing user-item pair and then use these scores to recommend most preferred items to the users. To test our system, we will observe how well our recommendations align with the items that were actually purchased by the customers in a transaction.

We investigated methods to alleviate the sparsity of user-item interactions using association rules, but relying solely on generic association rules would introduce popularity bias in the data. To reduce the popularity bias, we decided to cluster the users into groups and learn relevant association rules for each group, making the associations rules more personalized to the groups. We then proceeded to use the rules learnt for each group to impute the user-item interaction matrix, also known as the utility matrix, making it less sparse and

therefore using patterns in data to meaningfully augment it. We observed that using this method, we were able to generate a model that gave results that improved the precision by nearly 8% than the conventional models with a confidence level of 90%.

3 Exploratory analysis

The given dataset contains observations about items purchased by a customer on a given date. Each record in the data consists of the membership ID of the customer, the item that was purchased, and variables describing the date of purchase. The dataset was split into training and testing datasets such that items from each basket were split into train and test sets. Hence, given the user history and partial information about the current basket, we will build a recommender system that can reliably predict the items that were present in the same basket for a customer.

3.1 Long tail characteristics

The training data consisted of 3872 unique customers who bought items from a set of 167 unique items. To analyze if a recommender system would be useful in the store, we identified that half of the purchases were made for 149 items that are less popular as shown in the long tail in the item vs frequency chart in Figure 1. Therefore, it indicates that we can make use of a recommender system since a huge portion of profit is accounted for by less popular products in the store.

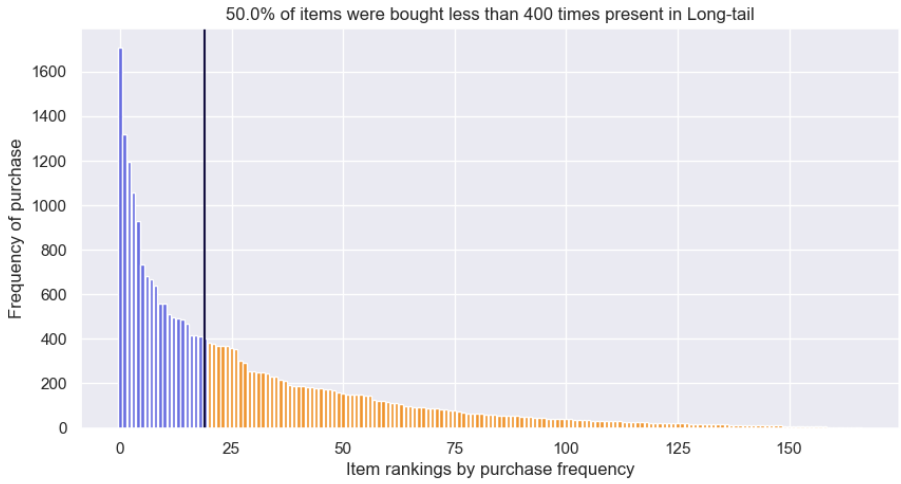

Figure 1: The Long tail of items based on purchase frequency

3.2 Clustering

Having established the necessity of a recommender system for the store, we then wanted to uncover any groupings in users or items to pick a suitable similarity measure to group them. On conducting a Principal Component Analysis for both users and items as shown in Figure 2, we found that users clustered together much better compared to items. To observe if the

clusters are maintained in higher dimensions as well, we performed K-Means clustering with three groups, which were identified during PCA as shown in Figure 2, and plotted them using Uniform Manifold Approximation and Projection (UMAP (Dorrity et al. 2020)) as shown in Figure 3. UMAP is a visualization technique suitable for high dimensional sparse datasets, which makes a better choice for given data compared to T-SNE due to the known sparsity in training data.

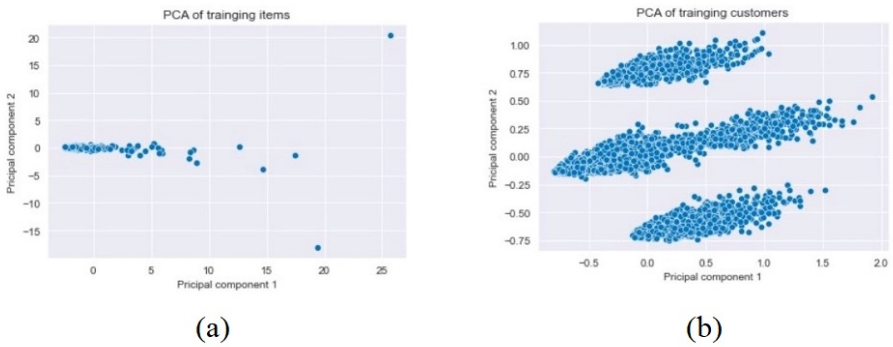

Figure 2: The first two principal components of training using: (a) Items; (b) Customers

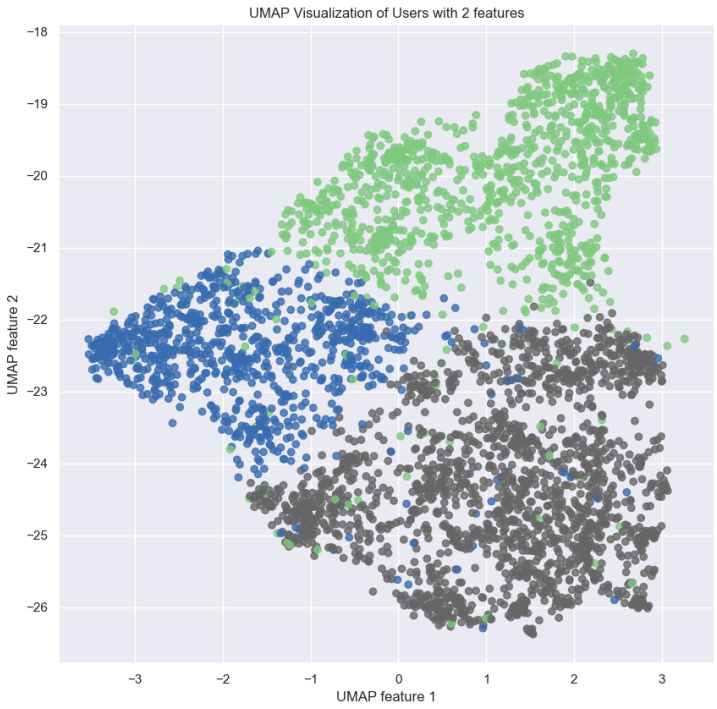

Figure 3: The UMAP visualisation of users with 2 features

3.3 Sparsity of data

A major challenge for building a collaborative filter is the sparsity of the data, we only had about 4% of the user-item interactions in our dataset. To alleviate this we propose usage of association rules to impute the data. Using the global association rules would introduce popularity bias in the data. To reduce the impact of popularity bias we intend to use association rules learnt from each cluster of users to populate their user-item interaction as shown in Figure 4.

3.4 Proposed recommendation system

The diagram for our proposed system is as follows, the detail explanation is in session 7.

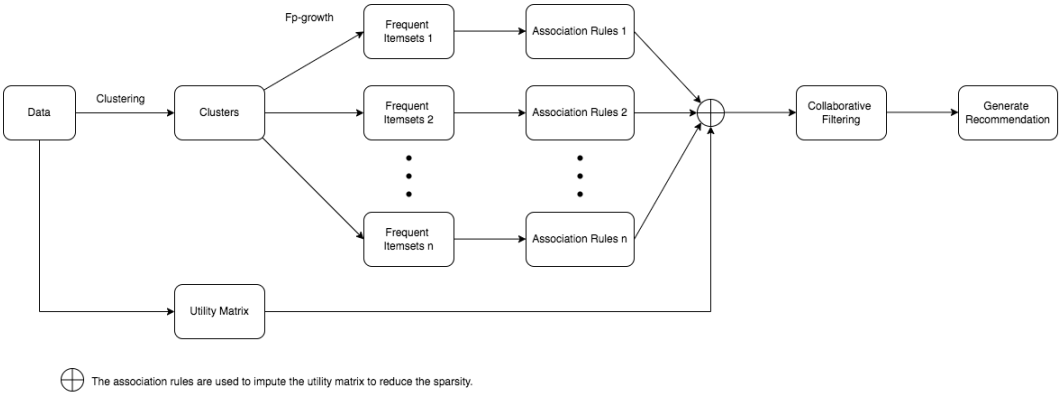

Figure 4: The flow of the recommendation system

4 Frequent pattern mining

The goal of mining frequent patterns is to be able to identify items that are bought together by adequately many customers. A group of items appearing together in a basket with a frequency more than a support threshold is regarded as a frequent item set. Additionally, we can derive association rules of the form $I \to j$, where I is a set of items and j is an item, implying if a basket contains I then j is likely to appear in it.

4.1 A-Priori algorithm

The A-Priori algorithm generates candidate itemsets of increasing length and removes those that do not meet the minimum support threshold, utilizing the property of monotonicity of itmesets; which states that is a set of items I is frequent, then so will be every subset of I. It iteratively scans the database to find frequent itemsets of increasing length with every scan, storing all potential sets in main memory with every iteration. Due to its iterative computation and memory requirements, it can be computationally expensive and even infeasible for large datasets.

4.2 FP-growth

To find frequent itemsets, the FP-Growth algorithm works by creating a compressed representation of the dataset called FP-tree (Borgelt 2005). The FP-tree reduces the memory storage and increases the efficiency of frequent itemsets mining. The algorithm recursively traverses the tree in a depth-first manner and further constructs the conditional FP-trees, which avoids iteratively scanning the database. The conditional FP-trees are then used to extract the frequent itemsets based on the minimum support threshold.

4.3 Comparison between FP-growth and A-Priori

As our recommender system has to be scaled up to millions of users, we need an efficient frequent itemsets mining algorithm. Table 1 shows the comparison between FP-growth and A-Priori algorithm. Figure 5 shows the execution time of FP-growth and the Apriori algorithm. The FP-growth algorithm is more efficient than A-Priori Algorithm in terms of execution time and memory. Therefore, the FP-growth algorithm is a better choice.

Table 1: The comparison between FP-growth and A-Priori algorithm (Zoumanan n.d.)

A-Priori	**FP-Growth**
scans the dataset in each of its steps, more time-consuming	only one scan of the dataset, consumes less time
candidate itemsets are generated, more memory required	no use of candidate itemsets, less memory required
breadth-first search	depth-first search

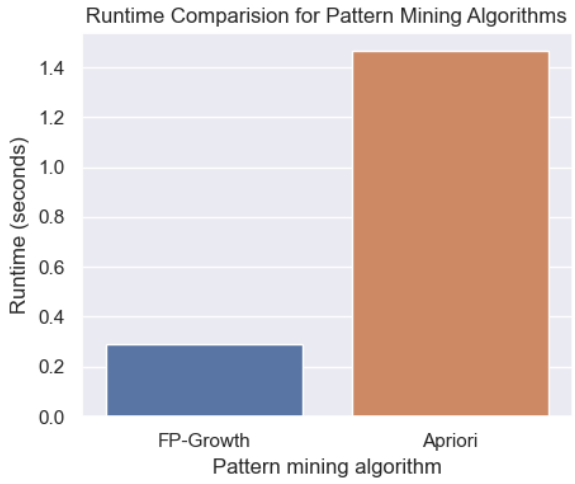

Figure 5: Execution time of FP-growth and A-Priori algorithm

4.4 Hyperparameters for frequent items mining

An itemset is said to be frequent when it has a higher support than the chosen threshold. If two items have no association, then they will appear together at random in any basket

with a probability p, as given in equation 1.

$$p = \binom{m}{2} \frac{1}{k^2} \tag{1}$$

where k is the number of unique items in the dataset, and m is the number of items in the basket. By analyzing the training dataset and consulting the management, we found that 4 is the median number of items present in a customer's basket. Approximating that every customer has $m = 4$ items in their basket we can model the distribution of number of times two items randomly appear together in a basket using a binomial distribution as given by equation 2

$$P(X = x) = \binom{n}{x} p^x (1-p)^{n-x} \tag{2}$$

where,
n is the total number of transactions,
p is probability of success as defined by equation 1,
x is the number of transactions where the two items appeared together

Using the probability density function of a binomial distribution, we concluded that two items will not appear more than eight times if they do no have any association with a confidence level of 99%. Therefore, we found that a frequent itemset with more than minimum support threshold given by equation 3 can be confidently said to be associated.

$$\text{minimum support threshold} = \frac{8}{n} = \frac{8}{13901} \approx 0.0006 \tag{3}$$

After finding the frequent itemsets based on a threshold greater than or equal to the minimum found above, we filter the generated association rules in two steps. First, we select the rule with lift greater than or equal to one, ensuring that the antecedent and consequent are occurring more often together than independently. Then, based on the business profit margins we pick rules with a confidence of more than 0.01, since it would make up a profitably large portion of given million transactions.

4.5 Evaluation of Association Rules

Association rules were generated for both the training and testing dataset. We evaluated the quality of association rules found by measuring the number of test association rules that were identified in training association rule as given in equation 4. We observed a precision of 28.57%

$$\text{Association rules precision} = \frac{|\{Test_{I \to j}\} \cap \{Train_{I \to j}\}|}{|\{Test_{I \to j}\}|} \tag{4}$$

5 Collaborative filtering

The collaborating filtering systems recommend items to customers based on similarity of user and/or items. We evaluated different recommender methods as described in this section to select a suitable approach for the given dataset.

5.1 Baseline model

We will use a random baseline model to act as a reference for more complex recommender methods. This model randomly generates a rating $\hat{r}_{ui}$ for a user u and item i pair from a normal distribution of $\mathcal{N}(\hat{\mu}, \hat{\sigma}^2)$ where $\hat{\mu}$ and $\hat{\sigma}$ are estimated from training data as given in equation

$$\hat{\mu} = \frac{1}{|D_{train}|} \sum_{r_{ui} \in D_{train}} r_{ui}, \quad \hat{\sigma} = \sqrt{\sum_{r_{ui} \in D_{train}} \frac{(r_{ui} - \hat{\mu})^2}{|D_{train}|}} \tag{5}$$

where D_{train} is the training data.

5.2 k-NN Based Collaborative Filters

This approach finds k nearest neighbours based on a similarity measure and estimates the rating for a user-item interaction based on rating from neighbours. We have explored two main approaches of k-NN based filters which are user-based and item-based by using cosine similarity. Since we found that user-based similarity is superior to item-similarities as seen in Exploratory analysis we will focus on user-based k-NN collaborative filters.

5.2.1 Similarity Measure

Since the given dataset does not have explicit ratings, we decided to use frequency of items bought by customers as their implicit rating. Unlike movie ratings, where a low score would mean a negative interaction, for our implicit ratings a low score would not mean the same since the customer bought a product. Therefore, we decided not to normalize the ratings as this would make missing interactions more important than less frequently bought products.

Furthermore, to compare the similarity between two users we decided to use cosine similarity as it would consider the absolute value of the implicit ratings and the similar rated products while producing a similarity score as given by equation 6 for users u and v. Methods such as Jaccard similarity would discount on the rating value and lose important information.

$$\text{cosine\_sim}(u, v) = \frac{\sum_{i \in I_{uv}} r_{ui} \cdot r_{vi}}{\sqrt{\sum_{i \in I_{uv}} r_{ui}^2} \cdot \sqrt{\sum_{i \in I_{uv}} r_{vi}^2}} \tag{6}$$

where r_{ui} is the calculated implicit rating for item i by user u.

5.2.2 User-based k-NN Collaborative Filter

User-based collaborative filtering or user-user collaborative filtering is a memory-based recommendation system. This recommendation system algorithm uses user similarity to make

product recommendations. It utilizes the user-item utility matrix generated based on the ratings provided by the user, which are implicitly calculated as mentioned earlier for given dataset. To predict a rating $\hat{r_{ui}}$, for a user u and item i this approach searches for k users that are closest to the user u based on the similarity measure, also referred to as k-nearest neighbours $N_i^k(u)$. The a weighted sum of the ratings by neighbours on item i is estimated to be the predicted $\hat{r_{ui}}$ as given by equation 7.

$$\hat{r}_{ui} = \frac{\sum\limits_{v \in N_i^k(u)} \text{cosine\_sim}(u, v) \cdot r_{vi}}{\sum\limits_{v \in N_i^k(u)} \text{consine\_sim}(u, v)} \tag{7}$$

where $N_i^k(u)$ is a set of k nearest neighbours of user u based on cosine similarity measure.

To find the optimal number of neighbors we used grid-search cross validation and used the best found hyper-parameter by identifying the lowest RMSE value.

5.3 Singular Value Decomposition

SVD is a dimensionality reduction technique which is effectively used to alleviate the sparsity of user-item utility matrix for collaborative filters(Ma 2008). Since our proposed system also intends mitigate the sparsity of utility matrix, we will compare the proposed method to this standard technique and compare their results. SVD is a matrix factorization technique that uses eigen vectors to decompose a matrix in three components as given by equation 8.

$$\mathbf{M} = \mathbf{U}\Sigma\mathbf{V}^T \tag{8}$$

where,

$\mathbf{U}$ is a left singular matrix containing eigen vectors of $\mathbf{M}\mathbf{M}^T$,

Σ is diagonal matrix containing eigen values,

$\mathbf{V}$ is a right singular matrix containing eigen vectors of $\mathbf{M}^T\mathbf{M}$

We tuned the number of latent factors used by the model using grid-search cross validation method by observing the lowest RMSE and used the optimal found parameters for training.

5.4 Evaluation metrics

To evaluate the models generated we will use two set of metrics. Firstly, we will evaluate the quality of ratings produced by the collaborative filter. Then, since the ratings were implicitly generated to improve model performance by increasing the information density, we do no have access to explicit ratings in test set and using similar way to generate rating would bias the results due to disproporationate distribution of data. Therefore, for a fair comparision to judge the quality of recommendation we will rely on 0/1 model of recommendations (Leskovec, Rajaraman, and Ullman 2014) which are described below

5.4.1 Collaborative Filter Metrics

Mean Absolute Error

MAE measures the mean of absolute value of distance between the predicted and actual rating for a user-item interaction as given by equation 9. It gives a point estimate of how far off the ratings were predicted from the original ratings.

$$MAE = \frac{1}{n} \sum_{\{u,i\} \in U} |\hat{r}_{ui} - r_{ui}| \tag{9}$$

where, U is all user-item rating pairs in the dataset, $\hat{r}_{ui}$ is the predicted rating of item i for user u, r_{ui} is the rating of item i given by user u

Root Mean Squared Error

RMSE measures the root of average squared distances between the predicted and actual ratings as given in euqation 10. It penalizes higher deviations more compared to Mean Absolute Error and is used popularly for recommender systems owing to the Netflix prize (*Netflix prize,* n.d.).

$$RMSE = \sqrt{\frac{1}{n} \sum_{\{u,i\} \in U} (\hat{r}_{ui} - r_{ui})^2} \tag{10}$$

where the symbols mean the same as mentioned for Mean Absolute Error.

5.4.2 Recommendation Metrics

Precision

It is defined as the ratio of true positives and all positive predictions, which for recommender systems translates to the formula given in equation 11. This metric provides information about how many of the recommendations are actually relevant to the users.

$$\textbf{Precision} = \frac{\text{number of correct recommendations}}{\text{number of total recommendations}} \tag{11}$$

Coverage

Coverage is calculated as the ratio of true positives and all positives, which becomes ratio of correct recommendations and all items that could have been recommended as given in equation 12. It gives an estimate for how many of the products that are were bought by the customers could be recommended using the system.

$$\textbf{Coverage} = \frac{\text{number of correct recommendations}}{\text{number of bought items}} \tag{12}$$

6 Recommendations only from Frequent Patterns

To generate recommendations from association rules we consider the current items in the user's basket and find association rules which have current items as antecedents. Then we

collect all consequents from the rules and pick top k consequents as the recommendation for the user as given by equation 13 using their support values.

$$\textbf{Recommendations(i)} = argmax^{1:k}_{support}\{j : i \in I \text{ for } A_{I \to j}\} \tag{13}$$

where, $A_{I \to j}$ is set of association rules, and i is set of items in current basket. The recommendations generated solely from association rules are generic for all users, and when the cart it empty no recommendations can be made using the rules which is when we recommend the most frequently bought items as a fallback to the customer.

7 Proposed approach

As it was observed in the exploratory analysis, the data obtained accounted only for about 4% of the possible user-item interaction and rest should be predicted by a recommender system. The sparsity of data is a major bottleneck when it comes to training the collaborative filters. We propose the use of clustered association rules to impute the user-item utility matrix which reduced the sparsity of training data by nearly 40%.

To impute some of the missing interactions we first cluster the users using k-means clustering, where the number of clusters were identified using Principal Component Analysis in Figure 2. Association rules were found for individual clusters which were used to impute the missing user-item ratings for each user based on their cluster association. We then train a user-based k-NN collaborative filter, where the number of neighbours were chosen based on cross validation with grid search. The propose method for imputing the entries to reduce sparsity is shown in Algorithm 1.

Algorithm 1: Algorithm to impute utility matrix using clustered association rules

```
Input: user id, utility matrix, clustered association rules
Output: imputed utility matrix

for user, item in utility matrix do
    Use trained K-Means to find user cluster association rules = cluster association
     rules[user cluster] for antecedents in association rules do
        if user has bought item in antecedents then
            for consequent in top association rules do
                utility matrix[user][consequent] += 1;
            end
        end
    end
end
```

By reducing sparsity of them matrix the collaborative filter will have more information to learn the missing ratings. Moreover, using clusters of users would avoid popularity bias in generating recommendations since different association rules were used for different users based on their cluster associations.

8 Discussion of results

We trained earlier mentioned models and compared their performance on their quality model fit and their ability to generate relevant recommendations using the test dataset.

8.1 Evaluating Model Fit

To compare the quality of model fit, we computed the RMSE and MAE for the models. We observed that all trained models had lower metrics compared to the baseline model as shown in Figure 6. As expected we found that user-based model performed better that item-based collaborative filter, since the users cluster well together as shown in the exploratory analysis. Lowest values for both metrics were observed for the proposed imputed solution on the test data, indicating that the recommendations made by it were more relevant than other models.

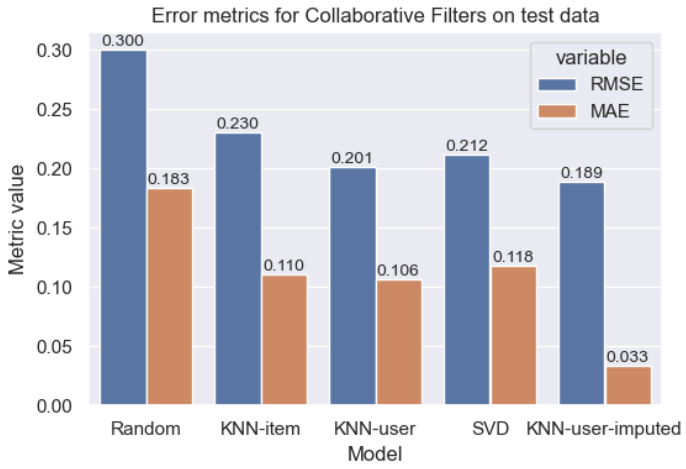

Figure 6: Error metrics of model fit for collaborative filters

8.2 Evaluating Recommendations

To evaluate the relevance of recommendations we used coverage and precision metrics. Similar to the model fit, all trained models performed better than the baseline model as shown in Figure 7. The sparisty reduction methods, namely SVD and proposed methods were able to achieved a higher value of precision and coverage for the testing data. Moreover, we observed that the proposed method was able to achieve higher coverage and precision for recommending items in test dataset, compared to the SVD method. To further check if the proposed system is actually better at recommending items compared to SVD, we performed a t-test and found that the performance improvement by using the proposed method in generating recommendations was statistically significant at 90% confidence level.

8.3 Model Predictions

Using the user-based collaborative filtering with clustered association rules, we were able to find meaningful patterns of items that were bought together, to evaluate the relevance of

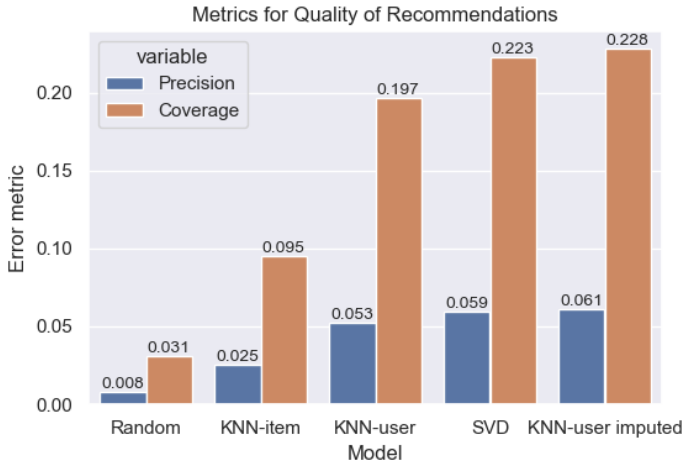

Figure 7: Metrics for quality of recommendations

association rules learnt during training we compared them to association rules generated by testing set, and found nearly 28% overlap among them. Some of the common association rules identified both in training and test set are shown in Table 2.

Table 2: Five examples of frequent patterns with their confidence and support on both training and test sets

Antecendents	Consequents	Training support	Testing support	Training confidence	Testing confidence
brown bread	pork	0.00093	0.00045	0.03163	0.0256
curd	sausage	0.00123	0.00068	0.04696	0.04109
frozen vegetables	root vegetables	0.00115	0.00103	0.05316	0.07563
white bread	sausage	0.00093	0.00046	0.05263	0.03539
semi-finished bread	whole milk	0.00094	0.00057	0.13541	0.10869

The proposed method generates recommendations by not only taking personal preferences of user into account but also the current items in their cart. A sample of recommendation generated for five random users based on two different current carts are shown in Table 3.

8.4 Scalability Estimation

To find the frequent itemset, we use FP-growth which takes linear time with respect to the number of transactions and the generated utility matrix can be imputed in linear time based on the number of user-item pairs in the system. Furthermore, training the user-based KNN model will require $O(M^2 * N * k)$ where M is number of users, N is number of items, and k is the number of neighbours for finding similar users. Based on the running times our system can be trained. Based on the times observed, we can compute that training the whole system on an average AWS (Amazon Web Services) instance, with a frequency of 20GHz, when we have a million transactions would be nearly 2 hours, given that the training data is representative of the overall load of the system.

Table 3: Recommendations generated for five randomly chosen user when they had {rolls/buns, whole milk}(cart A) and {sausage, root vegetables, other vegetables}(cart B)

User ID	Current cart	Recommendations
1000	A	{sausage, root vegetables, other vegetables, soda, yogurt}
	B	{whole milk, soda, rolls/buns, yogurt, tropical fruit}
2001	A	{other vegetables, yogurt, soda, tropical fruit, sausage}
	B	{whole milk, yogurt, soda, tropical fruit, citrus fruit}
2022	A	{soda, root vegetables, shopping bags, tropical fruit, pip fruit}
	B	{soda, shopping bags, tropical fruit, pip fruit, bottled beer}
2103	A	{frozen dessert, flower (seeds), sliced cheese, frozen fish, yogurt}
	B	{frozen dessert, whole milk, flower (seeds), sliced cheese, frozen fish}
3004	A	{soda, yogurt, sausage, pastry, other vegetables}
	B	{whole milk, soda, yogurt, rolls/buns, pastry}

To analyze the space complexity, we consider the main memory required for the algorithm to run. The FP-growth used a tree representation to store itemsets requiring $O(N^2)$ space, and majority of the space is utilized by the utility matrix which would be $O(M * N)$, lastly to store the similarity matrix for the collaborative filter will require $O(M^2)$ space. Based on the above space complexity evaluations, we can compute that if each entry is stored in a floating point number, then the deployment server would required nearly 12GB of main memory to preform the calculation, but once the model is trained we only need to store the nearest neighbors for the users, which required far less memory.

Based on the above numbers, we can conclude that the system can scale to a million transactions with a server running at 20GHz (average AWS frequency), with nearly 12GB of memory. The server can also be refreshed daily with another parallel instance, generating relevant recommendations for users every day.

9 Conclusion

With a wide range of products sold at the store, we observed that a majority of less popular products from the long tail contributed significantly towards the profit and recommender systems can be leveraged to increase sales. We trained a baseline and different collaborative filtering models and proposed an alternative to the SVD based sparsity reduction technique by using clustered association rules to reduce the sparsity of the utility matrix, and training a user based k-NN collaborative filtering system to generate recommendations for the users. We observed that the proposed method was able to correctly recommend around 22% of the items, which upon performing a t-test against the SVD model gave a statistically significant improvement at 90% confidence level. Other than the significant improvement in recommend relevant items to users, our method can also be used to easily explain the produced recommendations, which is not possible with latent factor models like SVD. In conclusion, we were able to utilize the buying patterns of the groups to significantly improve the recommendations made by regular collaborative filters and SVD sparsity reduction technique.

10 Reflection

We observed that the sparsity problem in training collaborative filters can be effectively alleviated by making use of explainable association rules. In the frequent itemsets mining, we can adopt parallel FP-growth algorithm which would result in enhanced performance of the system.

Moreover, by clustering the users we were able to reduce the sparsity for each users in a more personalized way than using global association rules. To cluster the users PCA was used for our implementation, we want to explore the use of density based clustering algorithms to automate the grouping discovery.

To impute the ratings we used the lowest score in order to avoid any bias, but further exploration to the imputed rating using association rules can lead to more accurate imputations.

Finally, time plays an important role in ever changing user preferences, currently the frequent retraining of the system handles the changing preferences but in future we would like to incorporate time sensitive collaborative filtering methods (Sun et al. 2016) which can reduce the frequent refresh of system and possibly predict the change in preferences over time.

Future works include the improvements in our recommendation system as follows:

In the frequent itemsets mining, we can adopt parallel fp-growth algorithm which utilizes multiple processors to mine frequent itemsets in parallel to further reduce the execution time.

In the choice of clustering algorithm, we can use density-based clustering algorithms such as DBSCAN which are able to remove outliers and identify clusters of different sizes automatically without the need for specifying the number of clusters beforehand.

We can also include the time factor in weighting our transactions. In our dataset, we have transactions from 2014 to 2015. More recent transactions can be weighted higher as they are more representative in terms of purchasing pattern. This can further increase prediction accuracy of our recommendation system.

References

Borgelt, Christian (2005). "An Implementation of the FP-growth Algorithm". In: *Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations*, pp. 1–5.

Dorrity, Michael W et al. (2020). "Dimensionality reduction by UMAP to visualize physical and genetic interactions". In: *Nature communications* 11.1, p. 1537.

Leskovec, J., A. Rajaraman, and J. D. Ullman (2014). *Mining of Massive Datasets*. Cambridge University Press. Chap. 9, pp. 319–352.

Ma, Chih-Chao (2008). "A guide to singular value decomposition for collaborative filtering". In: *Computer (Long Beach, CA)* 2008, pp. 1–14.

Netflix prize, (n.d.). URL: `https://en.wikipedia.org/wiki/Netflix_Prize`.

Pazzani, Michael J and Daniel Billsus (2007). "Content-based recommendation systems". In: *The adaptive web: methods and strategies of web personalization*, pp. 325–341.

Sun, Limei et al. (2016). "A Time-Sensitive Collaborative Filtering Model in Recommendation Systems". In: *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, pp. 340–344.

Zoumanan (n.d.). *Comparative Analysis Between Apriori and FP Growth*. URL: `https://www.kaggle.com/code/keitazoumana/comparative-analysis-between-apriori-and-fp-growth`.