Chapter 5

Auto-detecting groups based on textual similarity for group recommendations

It is essential to consider the similarity of characteristics among the group members to generate a better recommendation. Members of an automatically identified group will have similar characteristics, so reaching a consensus on generating the recommendations may be very frequent. It requires users-items and their rating interactions from a utility matrix to auto-detect the groups in group recommendations. We may not overlook other intrinsic information to form a group. We hypothesize that textual information also plays a pivotal role in user clustering/grouping. In this chapter, we propose models to auto-detect the groups based on the textual similarity of the metadata (review texts). We also consider the order of user preferences in our proposed models. We have conducted extensive experiments on three real-world datasets to check the efficacy of the proposed models. We have compared the performance of the proposed models with the baseline and the state-of-the-art models. Our results are encouraging.

Outline: The rest of the chapter is organized as follows. Section 5.1 presents the methodology of the proposed model. Experimental setup and results are discussed in Section 6.2. Finally, Section 5.3 summarizes this chapter.

5.1 Methodology

Generally, the user-item and rating interactions in a utility matrix are used for automatic group identification. However, we hypothesize that considering a dataset's intrinsic information may be advantageous while forming a group. We propose multiple models using review texts to auto-detect groups.

5.1.1 Model 1

Our model auto-detects groups using the metadata (review texts of the users). It uses textual similarities of users' reviews to form the groups.

- *Universal Sentence Encoder*: Transfer learning [30] is the reuse of a pre-trained model for a novel task. We can make use of it in a wide range of applications. Of late, many pre-trained language models have been explored to make the natural learning process task easier. *Universal sentence encoder* [25] is a pre-trained language model. This model is publicly available in TensorFlow Hub [1]. The model is trained on a variety of data sources. It takes an input (English text) of variable length size and produces a vector of 512 dimensions as an output.

In each of the datasets, a user gives reviews to multiple items. We create the user profile by merging reviews of the items a user gives. The model employs this pre-trained language model for sentence embedding. We embed the review texts using the universal sentence encoder. Each user will have an embedding of 512 dimensions (generated from his/her user profile). After embedding, a matrix of dimensions $m \times n$ is generated where m is the total number of users, and n is 512.

Semantically similar user profiles will have similar embedding. The model calculates the similarity scores among the embeddings using cosine similarity. Then, it generates a similarity matrix. Finally, applying spectral clustering [2] on the similarity matrix helps to auto-detect user clusters.

[1] https://www.tensorflow.org/hub/tutorials/

[2] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.SpectralClustering.html

5.1.2 Model 2

Numerous research studies [81, 91] have been conducted to create meaningful numerical vector representations of words. With the most recent breakthroughs in deep learning, many incredible deep neural network architectures have been developed. *BERT (Bidirectional Encoder Representation from Transformers)* [37] is a model that makes use of Transformer [116]. BERT is a multi-layer bidirectional transformer. Bidirectional means BERT learns from both the left and right sides of a token's context during training. The BERT base architecture has 12 layers, 768 hidden layers, 12 attention heads and 110M parameters. There are two steps in BERT:

- Pre-training: A model is trained on unlabelled data during pre-training.
- Fine tuning: This step initializes with pre-trained parameters and fine-tunes all the parameters using labelled data from the downstream task.

The BERT model solves the problem of creating these vectors and encodes tremendous amounts of meaning and context from them.

Using BERT contextualized word embeddings (feature vector representation of words), sentence embeddings are generated using two methods:

- Average word embeddings.
- Use the *CLS* vector (a special classification symbol added at the beginning of every sentence) as the sentence embedding.

The proposed approach BERT underperforms; therefore, the Sentence-BERT [99] was proposed to overcome the drawback of BERT. We built a model for detecting the groups automatically by leveraging the Sentence-BERT to generate sentence embeddings.

Sentence-BERT: In [99], authors introduced Sentence BERT, a modification of pre-trained BERT network [37] that uses siamese [18] and triplet network [52] (A variant of siamese network) structure to derive semantically meaningful sentence embeddings that can be compared using cosine similarity. The term 'Siamese' means twins (identical/same). A Siamese network (Figure 5.1) is a deep neural network architecture containing two or more identical subnetworks. Here 'identical' signifies the same configuration with the same parameters and weights. The two convolutional neural networks

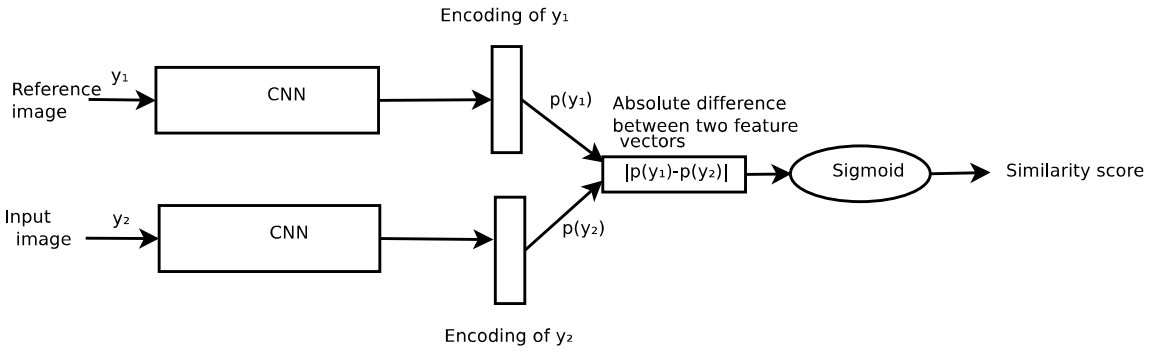


Figure 5.1: Siamese neural network

(CNNs) [71] are two copies of the same network. When parameter updates, this action is reflected in both the subnetworks. The subnetworks share the same parameters. The produced feature vectors can be used to generate the similarity of inputs, i.e., review texts (in the case of our application). The siamese network aims to learn the embedding on a deeper level and store the same classes/concept close enough to know semantic similarity more efficiently. The advantage of a siamese neural network is that it builds a robust model in the unavailability of large training data.

Similarly, in the Model 2, a user profile is created by merging reviews of the items reviewed by the user. After applying the sentence-BERT model to user profiles, we generate a matrix of dimension $p \times q$. Here, p is the total number of users, and q is 768. Model 2 computes the cosine similarity between the user profiles. The threshold is set to 0.5 to map those values to 0 and 1. A value greater than or equal to 0.5 maps to 1 and less than 0.5 or equal to maps to 0. We employ the Leiden algorithm [115] (a community detection algorithm) on the matrix to auto-detect a group of users. The Leiden algorithm outperforms Louvain algorithm [12] and detects better communities

5.1.3 Model 3

Sometimes it requires using a smaller, cheaper and computationally efficient model. The proposed model uses DistliBERT [104] to embed review texts. DistliBERT is a variant of the BERT based approach where the model uses fewer parameters (66M parameters). Due to this, the training time reduces drastically. The model is four times faster than the BERT model, and the performance degradation is only 3% compared to the original BERT model. The DistliBERT model also does not consider the pooler and token-type

embedding layer. DistliBERT uses knowledge distillation [51] in the neural network during training. In the teacher-student training, a student network (basically the DistliBERT network) is trained to mimic the full output distribution of the teacher network (in the supervision of the BERT network [37]). A smaller language model, i.e., DistliBERT, has been built using the teacher network. In the proposed model, the system further tries to reduce the search space using the *SIMHASH* [27]. SIMHASH (SIMilarity HASHing) is used in the approximate nearest neighbor search problem. SIMHASH is a locality-sensitive hashing (LSH) technique [44, 54]. The similarity metric is cosine similarity in the case of this LSH family. SIMHASH uses locality-sensitive hashing with random projection to find the nearest neighbors.

In Figure 5.2, multiple planes are placed in a higher-dimensional area. So, clusters based on data points can be determined by whether they are above or below the plane. We can find whether a vector is on that plane's positive or negative side for each plane. It requires to calculate the dot product between a point in the space (a vector) and the normal vector (a vector perpendicular to one of the planes). A positive value determines that the vector is on the same side as the normal vector. A negative value signifies that the vector is on the opposite side of the normal vector, and a zero value if the vector resides on the plane itself. The model determines the sign for each plane. In general, it provides multiple signals for each plane, and the objective is to find a way to combine all of those signals into a single hash value. The rule says that if the sign of the dot product is greater than or equal to 0, assign the intermediate hash value (h_i) to 1 and vice-versa. Equation 5.1 provides the single hash value ($Hash$). The single hash value defines a particular region within the vector space.

$$Hash = \sum_{i}^{H} 2^i \times h_i \tag{5.1}$$

Given a list of planes and vectors, we calculate intermediate hash value (h_i) corresponds to a plane and accumulate the sum of intermediate hash values (h_i) multiplied by 2^i to get the final hash value ($Hash$). H is the total number of planes. Here, i takes the value which starts from 0 and goes up to 2. If there are n documents (or data points), there will be n corresponding vectors. Each hash entry (bucket) should not hold more than 16 vectors. This case requires $\frac{n}{16}$ buckets. Each plane separates the search space

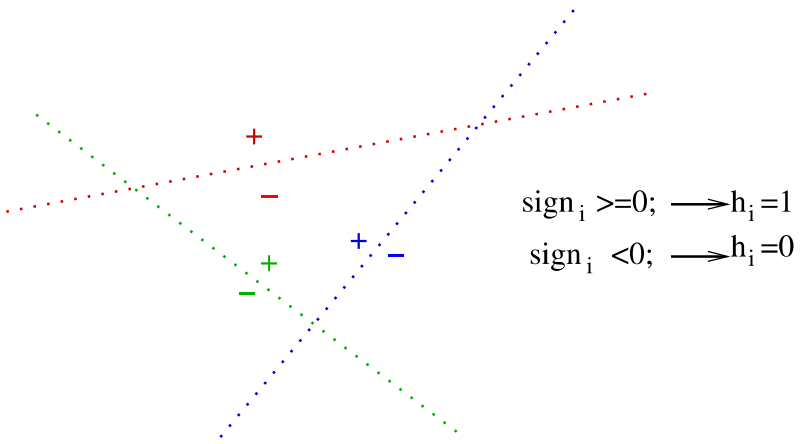


Figure 5.2: Assigning single hash value in multiple planes

into two parts (on the same and opposite sides). Hence, if there are p planes, then

$$\begin{aligned} 2^p &= \frac{n}{16} \\ p &= \log_2\left(\frac{n}{16}\right) \\ p &= \log_2(n) - \log_2(16) \\ p &= \log_2(n) - 4 \end{aligned} \tag{5.2}$$

Equation 5.2 is used to find the number of planes.

The vector space is divided into multiple sub-regions. A system does not know apriori the best set of planes to separate the vector space. It requires multiple sets of random planes to divide the vector space into numerous independent sets of hash tables. Multiple data points hash to the individual hash table, and those data points of the hash table become a candidate set to form a group.

We have a more robust way of searching the vector space for a set of vectors that are possible candidates to be nearest neighbors by using multiple sets of random planes for locality-sensitive hashing. The model does not search the entire vector space in these approximate nearest neighbors, but just a subset of it. So, this is not the absolute k-nearest neighbors, but an approximation of the k-nearest neighbors. The model may sacrifice some precision to gain efficiency in this search.

The proposed model sacrifices some precision in the result, but the time complexity to find similar users is linear. The proposed model produces k nearest neighbors. The model forms a cluster by comprising those neighbors. The model is better than Model 1 in terms of time complexity, where the time complexity of finding similar users from

the affinity matrix is quadratic and now reduces to linear runtime. The worst-case time complexity to form a group using Model 1 and Model 2 algorithms are $\mathcal{O}(NE \times k + n^2)$. Where N is the number of training examples, and E represents the number of epochs. k is the total time required to complete one epoch by a pre-trained language model and n is the number of users. In the case of model 3, the worst-case time complexity to auto-detect a group is $\mathcal{O}(NE \times k + H \times l)$, where H is the total number of planes and l is the number of the hash tables used in the algorithm.

5.1.4 Cluster validation

It needs to perform cluster validation over the obtained auto-detected groups. The most popular index is the Adjusted Rand Index (ARI) [3] [19,114] for cluster analysis. The ARI can get a value ranging from -1 to 1. The higher the value, the better it matches the original data.

$$ARI = \frac{RI - Expected(RI)}{Max(RI) - Expected(RI)} \tag{5.3}$$

Where RI is the rand index[4]. The rank index [19, 98] is a metric to evaluate the quality of clustering result. The range of the rand index is [0,1]. The value '0' signifies that the two data clusters do not agree on any pair of points, while the value '1' signifies that both the clusters are the same. The Rand Index (RI) is defined as follows:

$$RI = \frac{TP + TN}{TP + FP + FN + TN} = \frac{a + d}{\binom{n}{2}} \tag{5.4}$$

Where

1. TP (True Positive) : A true positive decision assigns two similar documents to the same cluster.

2. TN (True Negative): A true negative decision assigns two dissimilar documents to different clusters.

3. FP (False Positive): A false positive decision assigns two dissimilar documents to the same cluster.

4. FN (False Negative): A false negative assigns two similar documents to different clusters.

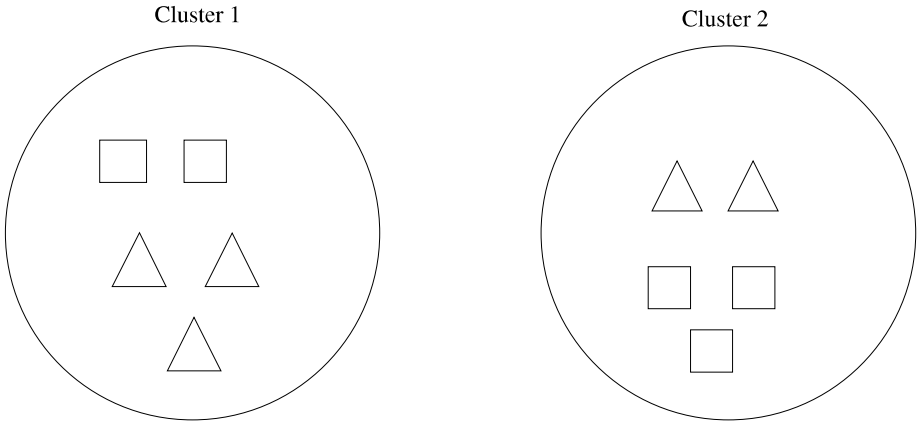


Figure 5.3: Two clustering assignments

Table 5.1: Contigency Table

	Same cluster	Different cluster
Same class	TP(a)=8	FN(b)=12
Different class	FP(c)=12	TN(d)=13

ARI [19] can also be represented as:

$$ARI = \frac{a - [\frac{(a+c)\times(a+b)}{\binom{n}{2}}]}{\frac{(a+c)\times(a+b)}{2} - [\frac{(a+c)\times(a+b)}{\binom{n}{2}}]} \tag{5.5}$$

Labels are also used for cluster assignment. Figure 5.3 shows ten data points ($n = 10$) into two partitions (cluster 1 and cluster 2). Each of the clusters contains 5 data points. Rand index works by considering pair of points. Out of all pairs, how many pairs agreed in cluster assignments? First, we compute TP+FP. The total number of positive or pairs of documents that are in the same cluster are: TP+FP= $\binom{5}{2} + \binom{5}{2} = 20$

Out of these, both square and triangle kinds of pairs in both clusters are positive. So,

TP=$\binom{2}{2} + \binom{3}{2} + \binom{2}{2} + \binom{3}{2} = 8$; FP=20 - 8=12 and

FN=$2 \times 3 + 3 \times 2 = 12$ and TN=$2 \times 2 + 3 \times 3 = 13$

Table 5.1 represents the contingency table. So, RI=$\frac{21}{45}$=0.46 and ARI=$\frac{8-[\frac{400}{45}]}{20-[\frac{400}{45}]}$=-0.08

[3]//scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_rand_score.html

[4]https://scikit-learn.org/stable/modules/generated/sklearn.metrics.rand_score.html

While $Precision(P) = \frac{a}{a+c} = \frac{8}{20} = 0.4$, $Recall(R) = \frac{a}{a+d} = \frac{8}{21} = 0.38$ and $F-measure = \frac{2PR}{P+R} = \frac{2\times0.4\times0.38}{0.4+0.38} = 0.39$

Once the community (group) is formed, the proposed models recommend items to maximize group members' satisfaction. Models use different consensus functions to generate the recommendation list for a detected group, determining that a group user is maximally satisfied with the recommended item set. The proposed models recommend items for a group by considering the order in user preferences. In the order preference model [1], first, each group member provides preferences in order. Then, the model aggregates individual user preferences using a consensus function to generate a recommendation vector of group budget 'k'.

We adopted the variants of least misery, most pleasure and aggregated voting methods in our models to generate the recommendation vectors:

- Least Misery Method (LMM) : We choose the least satisfied user in each iteration, and the item of his/her preference list that maximises the overall group satisfaction score is added in the recommendation vector.

- Least Misery Method with Priority (LMMP): The priorities of users are calculated to break a tie to choose a least satisfied user.

- Most Pleasure with Priority (MPP): Users' priorities are calculated to break a tie to choose the most satisfied user in each iteration, and the item that maximizes the overall group satisfaction score is added in the recommendation vector.

- GReedy Aggregated Method (GRAM): In each iteration, it greedily chooses an item from the user satisfaction score matrix [1] that maximizes the overall group satisfaction score and appends that item into the recommendation vector.

- Hungarian Aggregated Method (HAM): This method is an extension of the aggregated method that provides an optimal solution. Moreover, it produces the best results when we consider a smaller number of users and items.

The size of the preference list would be less than or equal to the group budget in the flexible user preference-based model [34]. Users of a group provide at least one item or a maximum of k items in the preference lists, and finally, the model generates a recommendation vector

of size k. We adopt the similar consensus functions of our previous work [1] in the proposed models.

5.2 Experimental setup and result analysis

We took three real-world datasets: Amazon-music (Digital Music) [5], Clothing Fit Data (Modcloth) [6] and Goodreads Book Reviews [7] for the evaluation of the proposed work.

Amazon-music (Digital Music): The digital music dataset holds reviews and metadata information from amazon. The digital music dataset has 5148 distinct reviewers and 2582 distinct asin (a unique number assigned to each product/music). The rating scale (named "overall") is 1-5. The total number of reviewers and asin (item) interactions over 'overall' attribute are 51796 (total instances). The dataset has also review text as metadata.

Clothing Fit Data (Modcloth): The dataset contains measurements of clothing fit from ModCloth[8]. The total number of distinct users are 47958. The total number of distinct products (items) is 1378. The total number of users and items interactions (transactions) are 82790. Each user has rated a product using the attribute "quality" on the rating scale of 1-5. There are eighteen attributes in the dataset. In addition, the dataset has auxiliary information like ratings and reviews, fit feedback, measurements of users and items, category, etc.

Goodreads Book Reviews: The complete dataset comprises 15,739,967 review texts. There are 2,080,190 different items, and 465,323 unique users are available. There are six different rating scales. The rating scale varies from 0 to 6.

Initially, all our experiments to generate sentence embedding are conducted on Google Colab Notebooks with the following specifications:

- GPU: 1xTesla K80 , compute 3.7, having 2496 CUDA cores , 12GB GDDR5 VRAM
- CPU: 1xsingle core hyper threaded Xeon Processors @2.3Ghz i.e(1 core, 2 threads)
- RAM: 12.69 GB Available

[5]https://jmcauley.ucsd.edu/data/amazon/

[6]https://cseweb.ucsd.edu/ jmcauley/datasets.html

[7]https://sites.google.com/eng.ucsd.edu/ucsdbookgraph/reviews

[8]https://modcloth.com/

- Disk: 107.72 GB Available
- n1-highmem-2 instance
- idle cut-off 90 minutes
- maximum 12 hours

Later, some of the experiments are also conducted on a CPU node of the ParamShivay supercomputer. It has an IntelxeonSKLG-6148 processor and 192 GB RAM with Linux operating system installed. All the algorithms are implemented in python language only.

5.2.1 Pre-processing step:

In the Amazon-music, Clothing Fit Data (Modcloth) and Goodreads Book Reviews [120] datasets, the irrelevant attributes (attributes that do not feed as inputs in the proposed model) are dropped from these datasets. These attributes do not play any role in our model during the data pre-processing task. Models consider the order of items/asin/books as they appear in the datasets. In the Amazon-music dataset, there is negligible empty review text from a particular user. So, replaces the NaN entry of the attribute 'reviewText' with a stopword. But, in the case of the Modcloth dataset, many entries in the attribute 'review_text' are empty, i.e., assigned with NaN in instances. So, the model drops those instances. Finally, the total number of users and items are 44856 and 1324, over 76065 transactions. In the Goodreads Book Reviews dataset, we built models on the 50000 reviews texts (instances). During pre-processing of the dataset, the NaN entries of the review_text attribute of the Goodreads Book Reviews dataset are replaced with a stopword.

5.2.2 Group formation step:

In the Model M1, the system employs the spectral clustering algorithm over the obtained affinity matrix after getting the embedded sentence vectors using the universal sentence encoder. Afterwards, it applies cosine similarity to calculate the similarity among review texts. The given datasets have text reviews in the English language. The system uses the multiple pre-trained language models for sentence embedding in the proposed models. In the case of multilingual text, the system could apply the multilingual universal sentence

encoder [127], the Sentence-BERT [99] or the DistliBERT [104]. Sentence-BERT and DistliBERT has trained over 100 languages. We consider users of each dataset to partition into some clusters. The system applies the silhouette coefficient to find the optimal number of clusters. Silhouette coefficient (s) can be defined as:

$$s = \frac{y - x}{max(x, y)} \tag{5.6}$$

Where x is the mean distance between a sample and all other points in the cluster.

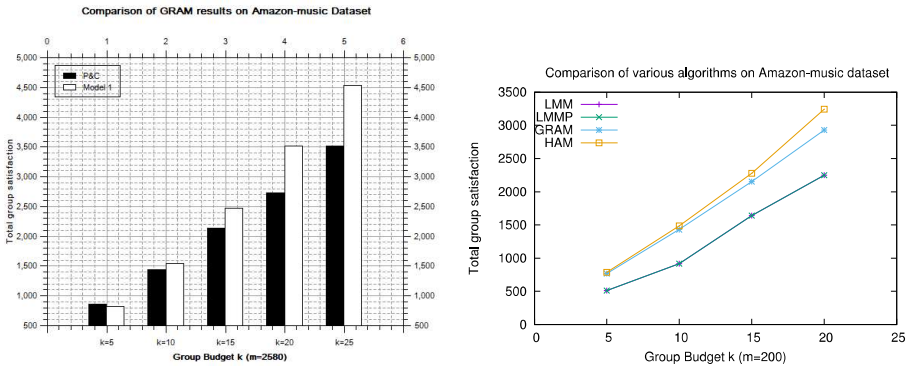
y is the mean distance between a sample and all other points in the next nearest cluster. The number of optimal clusters is two in Amazon-music and Modcloth datasets.

To measure the effectiveness of formed groups, we compare them to the baseline method. In baseline, *Predict & Cluster* (P&C) [15] model is used in a comparative study to check the efficacy of our proposed model. In the baseline, the system first calculates the scores of unrated items using collaborative filtering and applies the K-means technique for group formation. The hyper-parameters in the K-means-based model are the number of clusters ($k = 2$), seeds (initial value=0), and euclidean distance as distance measure.

5.2.3 Recommendation step:

The score vector holds values based on user satisfaction with order [1] function. To generate the score vector [1], the proposed models considers values of a and regularization factor c as 2 and 1, respectively. It determines the score based on the index by taking the absolute difference between the position of the item i of the user preference vector and the recommendation vector's item position p. The lower the difference, the higher the value it contains.

It has also been observed that the number of user preferences of a group in the Clothing Fit Data dataset is less than that of the Amazon-music dataset. In the proposed model, the choices of each member of the group are in order. Then, the model aggregates individual user preferences using a consensus function. Finally, the generated recommendation vector holds items of group budget 'k' (where 'k' is the maximum number of items that can be recommended) based on the proposed evaluation metric: group satisfaction score. The proposed models use aggregation strategies (e.g., LMM, LMMP, MPP, GRAM and HAM) to produce the recommendation for an automatically identified group. As a result, the proposed models maximize the overall group satisfaction score



(a) (b)

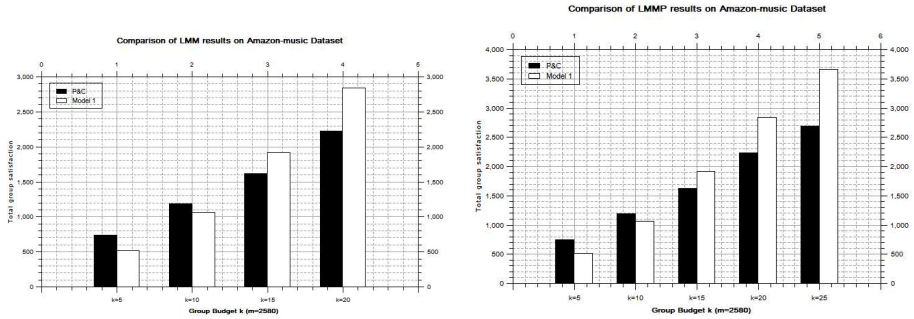
Figure 5.4: a.) GRAM results b.) Results using various consensus functions on Amazon-music Dataset

compared to the baseline method.

5.2.4 Results and discussions

We compare the result with the baseline model (P&C) and other state-of-the-art models to auto-detect groups to find the effectiveness of automatically detected groups. We used an external metric for cluster validation. We also found the optimal number of clusters among datasets. In the Amazon-music dataset, the rand index (RI) and adjusted rand index (ARI) scores are 0.5 and 0.046, respectively. The rand index score is much higher than the adjusted rand index score and validates the obtained clusters perfectly. We used the adopted versions of two widely known versions of consensus functions for recommendation: aggregated voting and least misery to prove the efficiency of the proposed system. The adopted versions that we used in this chapter are the Least Misery Method (LMM), the Least Misery Method with Priority (LMMP), the GReedy Aggregated Method (GRAM) and the Hungarian Aggregated Method (HAM).

We applied different consensus functions in automatically detected groups and measured the performance by varying values of group budget (k) and number of items (m). We conducted a comparative study between the proposed and baseline models to check the overall group satisfaction score improvement. The consensus functions generated group scores, which reflect the interests and preferences of all the group members.



(a) (b)

Figure 5.5: LMM and LMMP results on Amazon-music dataset.

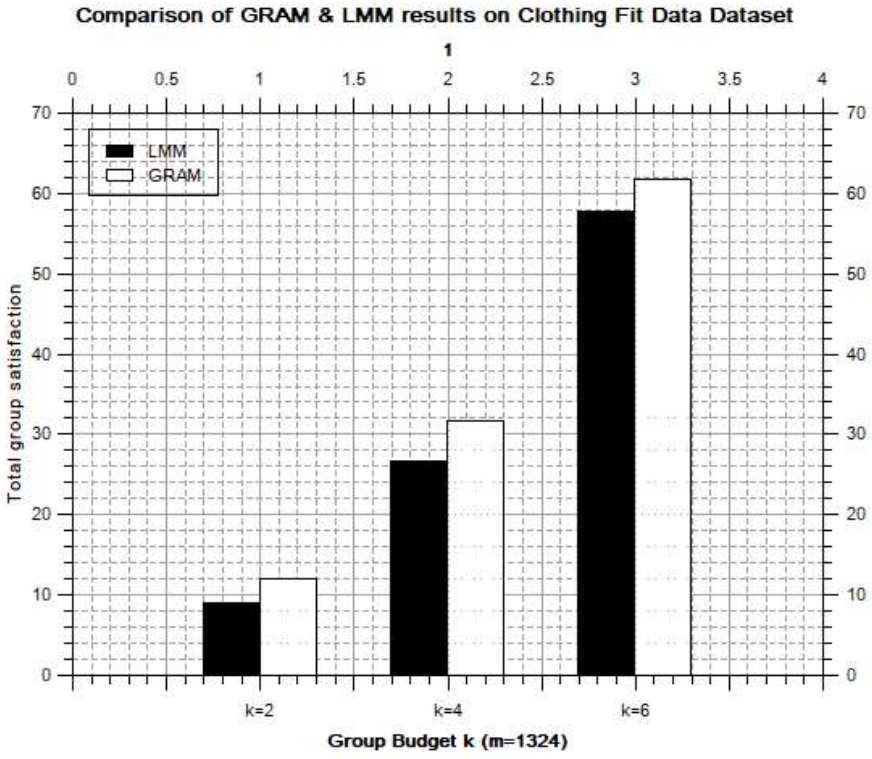
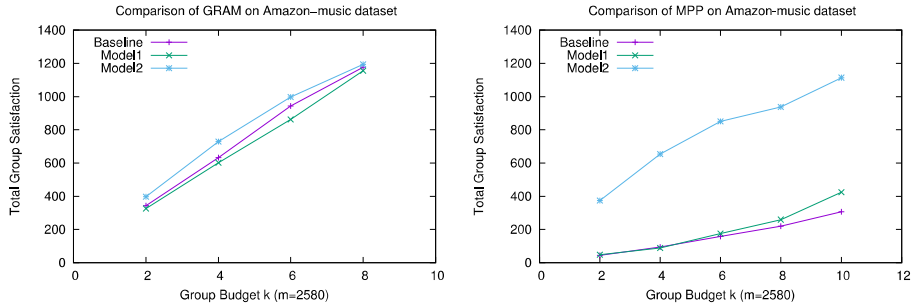


Figure 5.6: GRAM & LMM results on Clothing Fit Data Dataset



(a) (b)

Figure 5.7: Total group satisfaction of cluster size ($130 \leq clusterSize \leq 140$) on Amazon-music dataset using a.) GRAM b.) MPP

5.2.5 Baseline and state-of-the-art models:

In the proposed work to evaluate the efficacy of the models and the comparison purposes, various state-of-the-art models have been taken.

Predict and cluster (P&C): This (P&C) approach requires predicting the unknown ratings, and the interaction matrix is no more sparse. The traditional K-means algorithm is employed over the interaction matrix to obtain user clustering.

Bisecting K-means: The algorithm (Bisect K-means) produces a cluster of similar sizes. The method is more efficient when 'K' (number of clusters) are large. The Bisecting K-means works iteratively, and the step-wise procedures are as follows:

- First, it requires selecting the number of clusters (K) that would be formed.
- All data points are treated as a single cluster.
- Initially uses traditional K-means algorithm with 'K'=2 to split the cluster
- Measure the distance by taking the difference between each intra-cluster to the sum of square distance. Intra-cluster distance is the distance among members of a cluster rather than the difference between two clusters.

Fuzzy c-means (FCM): Fuzzy c-means belongs to a soft-clustering algorithm. Data points belong to other clusters as well. Consequently, it becomes vital to assign weights to each data point and each cluster that indicates the degree to which the data belongs. Weights are chosen randomly, and there is a constraint that the sum of weights

Table 5.2: Comparison with baseline methods on the basis of total group satisfaction scores of the automatically identified group of cluster size (n=40, m=2580) using GRAM on Amazon-music dataset

Group Budget(k)	Bisect K-means	FCM	Model 1	Model 2
2	120.7	13.8	84.1	126
4	234.9	43.6	168	242.9
6	350.4	101.1	237.9	372.9
8	489.3	218.1	302.9	524.9
10	643	333.6	434.2	684

associated with each data point equals 1. In [7], authors used FCM to find similarities between users and, based on the principal component analysis's score to determine candidate pairs of users that form an auto-detected group.

There are positive impacts on the results of the proposed models by varying group budget (k) and the number of items (m). It is evident from our experiments that a higher value of k provides a greater satisfaction. Figures 5.4, 5.5 and 5.6 show this characteristic. The overall group satisfaction increases with the increase in the number of items (m). Figure 5.4 validate this characteristic. In Figure 5.4, both LMM and LMMP algorithms show similar behaviour when considering items (m=200). Figure 5.7 shows a comparative study among the proposed models and the baseline model (P&C). The experiments show that the overall group satisfaction scores of the proposed models' (Model 1 and Model 2) are better than the baseline as the number of requested recommendations increases, i.e., group budget $k \geq 10$. Figure 5.7 shows the result of MPP algorithm on the proposed models (Model 1 and Model) and the baseline (P&C). Figures 5.5 and 5.7 show that the proposed model outperform compare to the baseline model (P&C). Table 5.2 presents comparisons among baseline and state-of-the-art methods and shows the efficacy of the proposed models (Model 1 and Model 2) to a great extent in the case of the Amazon-music dataset. We conclude from this study that consensus functions like LMM, LMMP, MPP, GRAM and HAM on the proposed models (Model 1 and Model 2) results in better group satisfaction by considering ordering in user preferences. The proposed Model 2 produces an overwhelming result. On the contrary, the Model 3 has a better runtime efficiency. The proposed group formation techniques have a positive role in improving the overall group satisfaction score.

Table 5.3 and Table 5.4 provide a comparison based on the total group satisfaction

Table 5.3: Comparison with baseline methods on the basis of total group satisfaction scores of the automatically identified group of different group sizes (n) and (m=36742) using GRAM on Goodreads Book Reviews dataset

k	Users(n)	Model 2	Model 1	Bisect K-means	FCM	P&C
5	200	76.9	80.03	82.9	85.5	87.7
10	141	281.5	297.2	264.4	286.3	287.5
15	107	513.6	523.2	467.1	481.4	484.5
20	95	856.2	854.7	837.6	798.1	825.4
25	82	1302.4	1232.1	1231.7	1162.9	1253.2

Table 5.4: Comparison with baseline methods on the basis of total group satisfaction scores of the automatically identified group of different group sizes (n) and (m=36742) using LMM on Goodreads Book Reviews dataset

k	Users(n)	Model 2	Model 1	Bisect K-means	FCM	P&C
5	200	32.9	39.24	32.4	39.2	43.1
10	141	147.3	159.8	154.8	159.8	137.6
15	107	375.2	313.2	348.2	310.4	309.9
20	95	621.1	639.9	613.6	548.7	621.3
25	82	1028	918.6	945.5	881.1	984.5

score of the automatically detected group of different sizes (n) using GRAM and LMM algorithms on the Goodreads Book Reviews dataset. We composed and detected groups of the different number of users who opted for at least 'k' items and accordingly evaluated the overall group satisfaction scores. The results in the tables have been depicted and concluded that the proposed approaches (Model M1 and Model M2) are promising methods to auto-detect the groups. The Model M3 reduces the time complexity for auto-detecting groups.

5.3 Summary

In this chapter, we auto-detect groups using text reviews and generate recommendations for users of the automatically identified group. One of the proposed model has a better overall group satisfaction score than the state-of-the-art. This chapter presents three different models, and a comparison study is performed among them. The proposed models (Model 1 and Model 2) are promising approaches to auto-detect the group in group recommendations. The Model 3 aims to reduce a problem in high-dimensional space to

an almost equivalent problem in much lower-dimensional, and time complexity is better than previously proposed models.

The extensive experiments over datasets show that review texts are auxiliary information for user clustering. Group recommender systems need to deal with scalability and sparsity problems over a utility matrix to compose a potential group but using our proposed approaches are more desirable. The proposed models may alleviate the sparsity problem to a great extent. In order to overcome the sparsity problem, there is a need to introduce more dimensionality reduction techniques to provide compact representations of users and items.