# MLG Final Project Report

## Attentive Group Recommendation System

Department of Electrical Engineering, NCKU
N26090693 吳承霖

## 1 INTRODUCTION

Online services are becoming the inseparable parts in our daily life, it's very common to use recommendation system to provide suggestion of items that the service user might be interested in. The recommendation for individuals have achieve a decent accuracy.

The online services also need recommendation for group of users. For example, there're groups in social media platform, and some online video streaming platforms also introduce family accounts, which shared by multiple individual user accounts. However, recommendation for group is still a challenging problem. The members in a group might have different preferences, also the levels of preferring of all members are not necessarily the same.

Thus, the strategy of aggregating group members' preferences is the main challenge of implementing group recommendation system. In this project, we implement *AGREE* model proposed in [1], which uses attention to obtain the weights of influence of group members, then the weights are used in aggregating group members' preference.

The model provides ability to predict both the rating scores of items for individuals and the rating scores of items for groups. The two algorithms share the same model.

In this project, we conduct experiments to examine the effectiveness of attention as well as the overall accuracy on dataset *CAMRa2011* compared with baselines.

## 2 PROBLEM DEFINITION

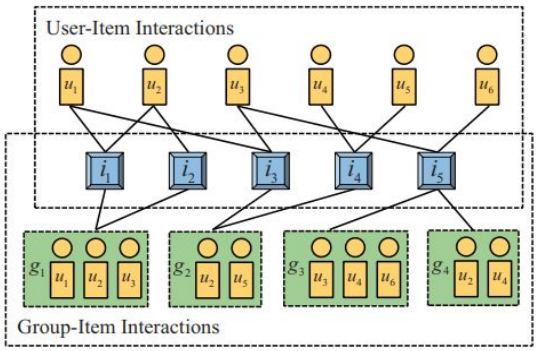
Due to the fact that member preference aggregation makes group recommendation difficult to achieve. We attempt to solve the problem of recommendation for both groups and individuals.

### 2.1 Input

Suppose we have $n$ users $U = \{u_1, u_2, .., u_n\}$, $s$ groups $G = \{g_1, g_2, ..., g_s\}$ and $m$ items $V = \{v_1, v_2, ..., v_m\}$. The mapping between users and groups is $K$. The total input data is:

- users $U$
- groups $G$
- items $V$
- user-item interaaction $Y$
- group-item interaction $R$

The input data of implemented model AGREE is illustrated below:




### 2.2 Output

Because our implemented model is able to predict the rating score from user and from group. The output of our group recommendation system are two personalized ranking functions that map an item to a real value for each group $f_g : V \rightarrow R$, and for each user $f_u : V \rightarrow R$.

## 3 METHODOLOGY

In this project, we have implemented the neural network-based recommendation algorithm, which was proposed in [1]. The implemented model 'AGREE' is an abbreviation of 'Attentive Group REcommEndation'. Generally speaking, AGREE model consists of two components: 1) Group representation learning which utilizes attention network to aggregate members' preference 2) feature interaction learning using NCF network model. In section 3.1, we will explain the model's frontend which embed sparse user id (uid), group id (gip) and item id (iid) into dense feature vectors. In section 3.2, the mechanism of attention will be illustrated, and we will further explain the intuition of applying attention on group representation learning. In section 3.3, we will introduce the final network layers in *AGREE* model which are responsible for performing feature interaction and predicting the input items' rating. Sections 3.4 and 3.5 are mainly about optimization objective for model training and metrics for performance evaluation.

### 3.1 Sparse Input Embedding

In the dataset we used to train our model, there are 290 groups, 602 users and 7710 items in total. Because the ids are the only feature we feed into our model, our features are very sparse, and the value of ids don't have any numerical meaning. Thus, before input ids feed forward to layers of prediction, we use embedding techniques to transform sparse ids into dense vectors.

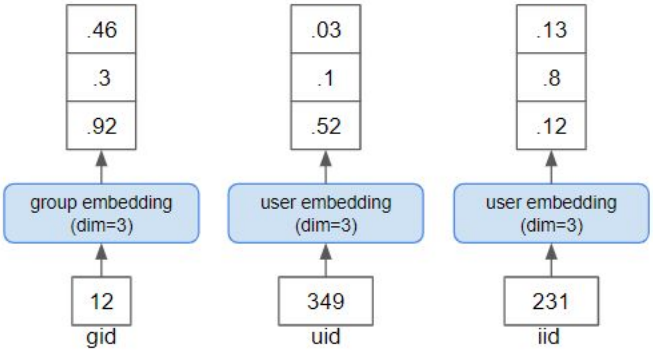



Figure1. Illustration of process of transform sparse ids into dense vectors using embedding.

### 3.2 Attentive Group Representation Learning

In this project we use *CAMRa2011* dataset to train our model and conduct model evaluation. A group consists of multiple users, thus we can view group members' preference as 'features' of the group. Our intuition is to use aggregation of member users' preference to represent the group instead of treating the group as a virtual individual then predicting its rating.

Most existing group recommender systems aggregate the scores of group members via some predefined strategies like 'average' strategy.

However, predefined strategies lack the flexibility to dynamically adjust weights of group members. This flexibility is particularly useful when a group makes decision on items of different types. Thus, our implemented AGREE model adopts attention network layer to learning the weights for each member's preference vector, i.e. using neural network to learning the weight of influence of each group member.

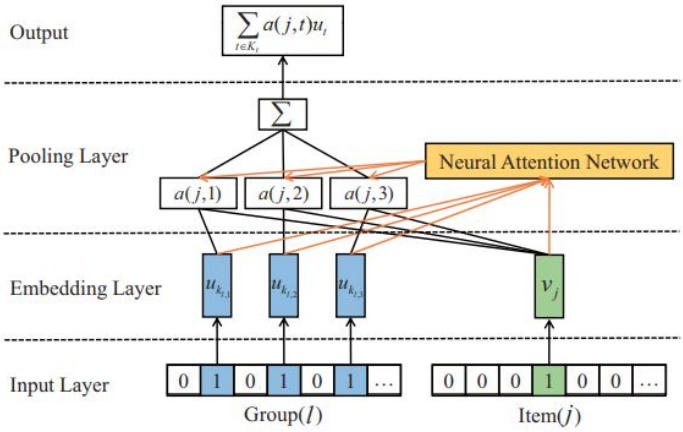



Figure 2: Illustration of group representation learning and member aggregation.

Let $u_i$ and $v_j$ be the embedding vector for user i and item j respectively. To implement recommendation for group, our target is obtaining a embedding representation for each group through aggregation. To learn dynamic aggregation strategy from data, it is necessary to define the group embedding as dependent of the embeddings of its member users and the target items, which can be abstracted as:

$$g_l(j) = f_a(v_j, \{u_t\}_{t \in k_l})$$

which $g_l$ denote the embedding representation of group $l$, the aggregation function $f_a$ embed item $j$ and each member of the group, then perform aggregation. In the implemented model, the aggregation consists of two components: 1) user embedding aggregation and 2) group preference embedding. The equation of whole process is listed below:

$$g_l(j) = \sum_{t \in K_l} \alpha(j,t) u_t + q_l$$

The first term in the above equation is user embedding aggregation, and $q_l$ denotes the group preference embedding.

Next, we will elaborate the two components of group representation learning.

**User embedding aggregation**. We perform a weighted sum on each members embedding vector. The weights $\alpha(j,t)$ which denote the influence of each member are learnable variables. The influence weights are learned by neural attention network which is implemented by simply fully-connected layers and dropout layers.

The input of attention network are the result of concatenation of user embedding vector and item embedding vector. The embedding vector of users represents the preference of individual users, and the embedding vector of items represents the item's self property. The attention network takes the concatenation of user vector and item vector as input and predict the influence of the user toward this item.

The above process is intuitive in the real life group decision making. For example, suppose a family consists of two members: user A and user B. The rating of item J from this group is possibly affected by the preference of both user A and B. If user A is familiar to item J or user A like item J much more than user B dose, then the influence weight of A toward item J is more likely higher than user B.

In the attentive group representation learning, the influence weight for user $t$ on item $j$ is learned in following equations:

$$o(j,\ t) = h^T ReLU(P_v v_j\ +\ P_u u_t + b)$$
$$\alpha(j,\ t)\ =\ softmax(o(j,\ t))$$

where $P_v$ and $P_u$ are weight matrices of the attention network that convert item embedding and user embedding to hidden layer, respectively, and $b$ is the bias vector of the hidden layer. We use $ReLU$ as the activation function of the hidden layer, and then project it to a score $o(j,\ t)$ with a weight vector h. Lastly, we normalize the scores with a softmax function, which is a common practice in neural attention network.

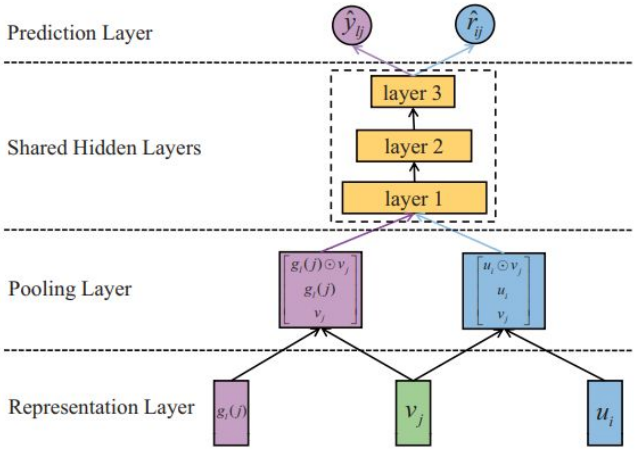



Figure 3: Illustration of pooling layer and NCF layers for learning feature interaction and prediction.

### 3.3 Interaction Learning with NCF

NCF proposed in [2] is short for Neural network -based Collaborative Filtering, which is generic and able to express and generalize matrix factorization. It is a multi-layer neural network framework for item recommendation. Its idea is to feed user embedding and item embedding into a dedicated neural network (which needs to be customized) to learn the interaction function from data.

As neural networks have strong ability to fit the data, the NCF framework is more generalizable than the traditional MF model, which simply applies a data-independent inner product function as the interaction function.

Due to the features mentioned above, we choose NCF to perform an end-to-end learning on both embeddings (that represent users, items, and groups) and interaction functions (that predict user-item and groupitem interactions).

Figure 3 illustrates our customized NCF solution.

In this project, we aim to implement algorithm for recommendation tasks for both groups and users simultaneously, we design the solution to learn the user-item and group-item interaction functions together. Specifically, given a user-item pair ( $u_i,\ v_j$ ) or a group-item pair ( $g_l,\ v_j$ ), the representation layer first returns the embedding vector for each given entity. Then the embeddings are fed into a pooling layer and hidden layers (shared by the two tasks) to obtain the prediction score.

For pooling layer, assuming we are going to predict the rating between group and item, then the input will be a group-item pair ( $g_l,\ v_j$ ), the pooling layer first performs element-wise product on their embeddings, and then concatenates it with the original embeddings:

$$e_0 = \varphi_{pooling}(g_l(j),\ v_j)\ =\ \left[g_l(j) \odot v_j, g_l(j), v_j\right]$$

If now we are going to predict the rating between individual user and item, the the input will be an user-item pair ( $u_t,\ v_j$ ), thus the pooling process will be:

$$e_0 = \varphi_{pooling}(u_t(j),\ v_j)\ =\ \left[u_t(j) \odot v_j, u_t(j), v_j\right]$$

The reason of doing pooling on the user/group embedding vectors and item embedding vectors is to achieve feature interaction using element-wise product. The results will be the interactions in higher level.
However, the element-wise product may lose some information in the original embeddings which may be useful for later interaction learning.To avoid such information loss, we need to further concatenate it with the original embeddings.

To summarize the architecture of network model *AGREE* implemented in this project, the brief illustration is shown in figure 4.

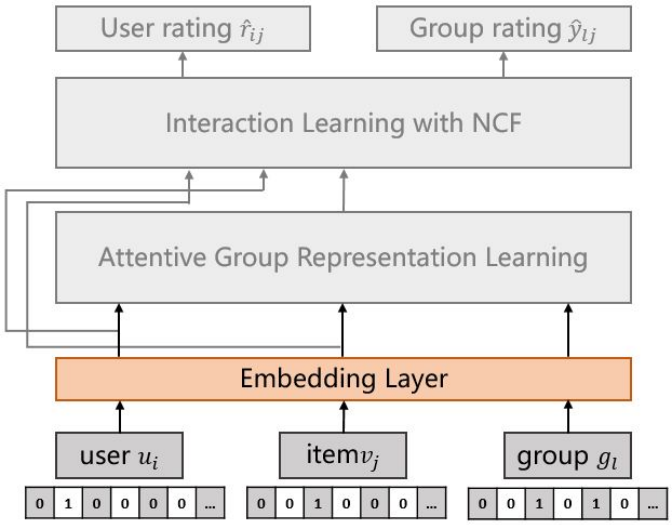



Figure 4: Illustration of model architecture of *AGREE*.

### 3.4 Optimization Objective

To simply our task, in this project we transform the observed interactions score to 1, and the other un-observed interactions to 0. In the *AGREE* model, we use pairwise learning method to optimize model parameters. The assumption of pairwise learning is that an observed interaction should be predicted with a higher score than its unobserved counterparts. Specifically, we employ the regression-based pairwise loss, which is a common choice in item recommendation:

$$L_{user} = \sum_{(i,j,s)\in O} (r_{ijs} - \hat{r}_{ijs})^2 = \sum_{(i,j,s)\in O} (\hat{r}_{ij} - \hat{r}_{is} - 1)^2$$

Where $O$ denotes the training set, the triple $(i,j,s)$ denotes that user $i$ has interacted with item $j$, but user $i$ hasn't interacted with item $s$ before. The item $s$ is what we call 'negative instance', which is sampled from the un-observed interactions in the dataset.
$\hat{r}_{ijs} = \hat{r}_{ij} - \hat{r}_{is}$ means the margin of the prediction of observed interaction $(u_i, v_j)$ and unobserved interaction $(u_i, v_s)$. Since we focus on implicit feedback, where each observed interaction has a value of 1 and unobserved interaction has a value of 0, we have $r_{ijs} = r_{ij} - r_{is} = 1$.

### 3.5 Evaluation Metrics

The dataset used by this project is CAMRa2011, which is a real-world dataset containing the movie rating records of individual users and households. The dataset are divided into training set and test set. The authors of [1] have processed the dataset, they added negative instances to every row in test set. Thus, the row format i test is something like:

$$uid,\ pos\ iid,\ neg\ iid\ 1,\ neg\ iid\ 2,\ \ldots,\ neg\ iid\ 100$$
$$or$$
$$gid,\ pos\ iid,\ neg\ iid\ 1,\ neg\ iid\ 2,\ \ldots,\ neg\ iid\ 100$$

Where *uid* and *gid* denotes the id of user and group respectively. After the *uid* / *gid*, the following term *pos iid* represent the item id which it has been interacted by the user/group. And the

rest terms *neg iid* are the id of items which haven't been interacted with the user / group. There are 100 negative instances in total in one row of testset.

In our implemented recommendation system, we expect that our algorithm will have output close to 1 for input pair ( *uid*, *pos iid* ), and generate output close to 0 for input pairs ( *uid*, *neg iid* 1 ), ( *uid*, *neg iid* 2 ), … , ( *uid*, *neg iid* 100 ). To sum up, there're 101 pairs passed through *AGREE* model and generate 101 predicted rating scores. Among these predicted ranks, we hope that the rankings for input ( *uid*, *pos iid* ) would be higher than predicted rankings of other negative instance pairs.

To examine the performance of our implemented *AGREE* model, we adopt two evaluation metrics:

- **HitRate@5** - After all 101 pairs are passed into model, we will further sort the output ranking from highest to lowest. If the output of input pair ( *uid*, *pos iid* ) is in the highest 5 predicted rankings, then we can it 'hit', the value of hit rate@5 would be 1. Instead, if the output of input pair ( *uid*, *pos iid* ) is not in the highest 5 predicted rankings, then the hit rate = 0.

- **NDCG@5** - NDCG is abbreviation for Normalized Discounted cumulative gain, which is a measure of ranking quality. In information retrieval, it is often used to measure effectiveness of web search engine algorithms or related applications. Using a graded relevance scale of documents in a search-engine result set, DCG measures the usefulness, or gain, of a document based on its position in the result list. The gain is accumulated from the top of the result list to the bottom, with the gain of each result discounted at lower ranks. The reason why we pick only highest 5 rankings is because we focus on the performance and accuracy on highly recommended items instead of the accuracy of the recommendation for not recommended items.

## 4 EXPERIMENTAL ANALYSIS

In this project, we adopt attention mechanism to simulate the complicated and dynamic process of group decision making. In order to examine the effectiveness of attention mechanism in group recommendation, we compare our implemented *AGREE* model with several models:

1. ***GREE* model** - since AGREE is short for 'Attentive Group Recommendation', model *GREE* is AGREE without attention mechanism. The way we implement this model is to remove attention layers in group representation learning. Instead, we assign uniform weight for each user's embedding vector (preference vector). Those uniform weights assigned to user embedding vectors are also similar to '*average*' group aggregation strategy, which is a fixed and predefined way of group representation.

2. **GNCF** - instead of using aggregation of all members' embedded preference vectors to represent a group. In GNCF, which stands for '*Group NCF*', we treats each group in dataset as a virtual individual. This is a intuitive and primitive way to implement recommendation system. However, the omitting of group member aggregation will

make the model have its members' preference. So we expect that the performance of *GNCF* in evaluation of following experiment will be weaker than *AGREE* and *GREE* model.

Apart from the experiment for effectiveness of attention mechanism, we further conduct series of test about the training parameter of our implemented AGREE model. In the process of model training, for each row in training set, we randomly sample negative instances (the items that user of the row has never interacted with). According experiment settings from the original paper of AGREE [1], the values of negative instance sampling range from 4 to 6, i.e. randomly sample 4 to 6 negative instances for each user-item pair in training set. We expect that the more negative instances are sampled, the better the trained *AGREE* model could perform.

The following sections will provide detailed results and analysis for previously mentioned experiments.

### 4.1 Effectiveness of Attention

In this experiment we compare the performance of several models: 1) *AGREE* 2) *GREE* 3) *GNCF*. The training set and test set for these three model are indentical. Due to the training time issue, we give 15 training epochs for each model and pick the highest values of evaluation metrics as its performance.

From experiment results shown in Table 1 and Table 2, we discover that model *AGREE* has better performance than the other two. However, to our surprise, *GNCF* performs better accuracy than *GREE* does. The reason behinds the observation might caused by the uniform weights of group aggregation used by GREE model which are insufficient to use member preference vector as the representation of a group.

Figure 5 and Figure 6 are trend of HR and NDCG during training. We can observe that our implemented model has overall best performance.

| Model | Hit Rate@5 | NDCG@5 |
|---|---|---|
| AGREE | 0.5434 | 0.3516 |
| GREE | 0.5082 | 0.3212 |
| GNCF | 0.5179 | 0.3286 |

Table 1: Comparison of three models on group recommendation.

| Model | Hit Rate@5 | NDCG@5 |
|---|---|---|
| AGREE | 0.5720 | 0.3742 |
| GREE | 0.5358 | 0.3446 |
| GNCF | 0.5661 | 0.3658 |

Table 2: Comparison of three different models on individual user recommendation.

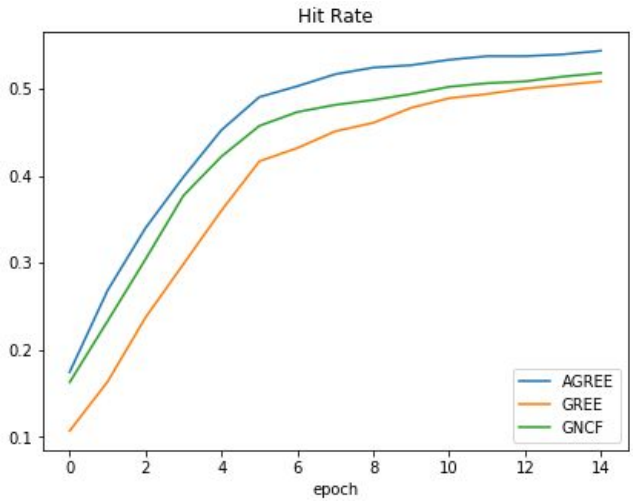



Figure 5: Trend of Hit Rate@5 performance of three different models.

### 4.2 Negative Sampling Number Test

In this experiment, we use different number of randomly sampled negative instances paired with each training example (positive instance) in *AGREE* model training. We execution only 5 epochs for each test's *AGREE* model training because the training time of an epoch takes lots of time to complete.

The result of this experiment are listed in Table 3 and Table 4. As we expected, the more negative instance are sampled during model training, the better accuracy the trained model could perform.

Figure 7 and Figure 8 are the trend of value of our evaluation metrics.

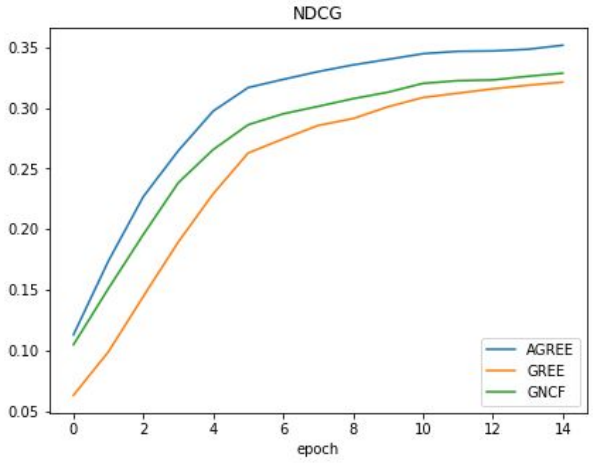



Figure 6: Trend of NDCG@5 performance of three different models.

| Model | Hit Rate@5 | NDCG@5 |
|---|---|---|
| neg = 4 | 0.4524 | 0.2974 |
| neg = 5 | 0.448 | 0.3181 |
| neg = 6 | 0.5103 | 0.3318 |

Table 3: Comparison performance of different number of negative sampled instances on group recommendation.

| Model | Hit Rate@5 | NDCG@5 |
|---|---|---|
| neg = 4 | 0.4936 | 0.3200 |
| neg = 5 | 0.5328 | 0.3425 |
| neg = 6 | 0.5511 | 0.3553 |

Table 4: Comparison performance of different number of negative sampled instances on individual recommendation.

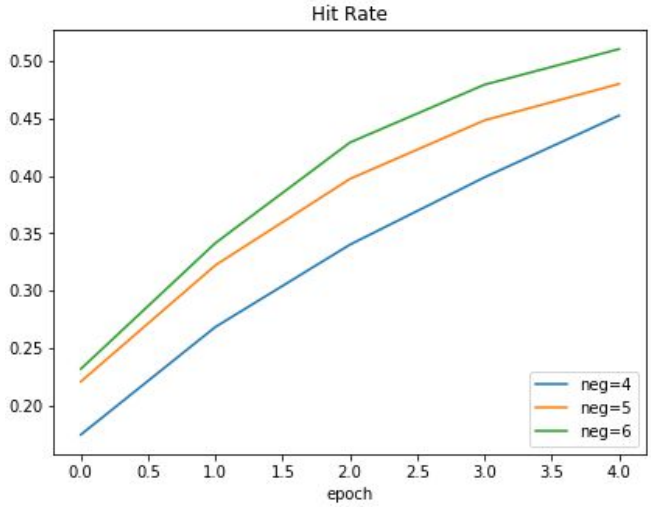



Figure 7: Trend of Hit Rate@5 performance of *AGREE* model using different number of negative instance.

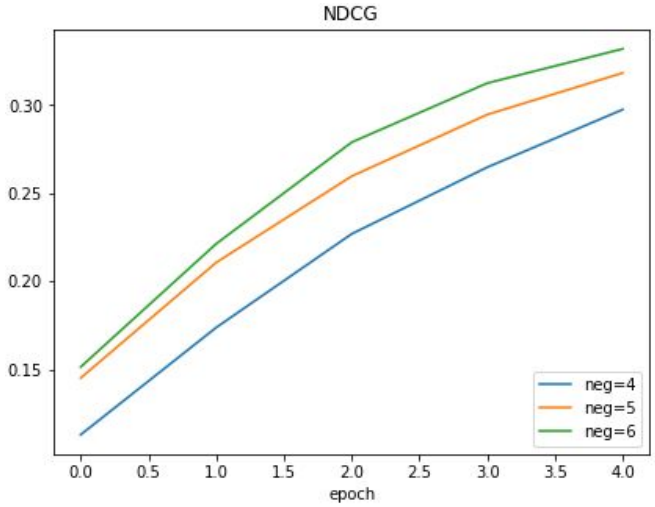


Figure 8: Trend of NDCG@5 performance of *AGREE* model using different number of negative instance.

## 5 CONCLUSION

In this project, we implement AGREE model which is proposed to solve group recommendation problem.

In our implemented model, it adopts attention mechanism to perform aggregation of member preference vector in order to represent the group using its members. In the process of training model, we adopt several evaluation metrics to investigate the performance on test set.

Besides implementing algorithm to solve group recommendation problem, we further conduct 2 experiment to get insights from our models and parameter settings. The first experiment shows the effectiveness of attention network in group representation, and the second experiments proves that higher amount of negative instances added into training set helps the model performance.

Although the performance of attention helps model *AGREE* beats other baselines on the evaluation of hit rate and NDCG. However, the values of hit rate and NDCG are still not very convincing in the dataset CAMRa2011. We need to further conduct training and experiment on another dataset containing group-item interactions. Also, the way of user embedding vectors interacts with item embedding vectors can be modified using other techniques like vector product or element-wise product instead of just concatenating the two and passing them into dense layers.

## 6 REFERENCE

[1] Da Cao, Xiangnan He, Lianhai Miao, Yahui An, Chao Yang and Richang Hong. 2018. Attentive Group Recommendation. In SIGIR' 18.

[2] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, Tat-Seng Chua. Neural Collaborative Filtering. In WWW '17.