ADL Final Project

Team 67

B09902035 李沁柔, B09902064 楊冠柏, B09902067 易冠廷, B09902118 林語涵

January 05, 2023

1 Abstract

We chose the Hahow challenge as our final project. In the introduction, we provided an overview of the task definition for this challenge. In the following section, we discussed related works and the various methods we investigated to solve the challenge. We demonstrated that using an ensemble model to combine multiple methods resulted in the best performance for both seen and unseen domain tasks. Finally, we analyzed the performances and characteristics of our proposed models.

2 Introduction

The competition dataset covers an 8 month period and includes 97,410 examples. Each example can be represented as a vector (u, r_u), where u is the user ID and r_u is the set of courses that user u has purchased. The dataset has been divided into three subsets: training (59,738 examples), validation (19,370 examples), and test (18,302 examples). The validation set is further split into seen (77,48 examples) and unseen (11,622 examples) subsets, indicating users that are present or absent in the training set, respectively. The test set is also split into seen (7,205 examples) and unseen (11,907 examples) subsets. The mean average precision (mAP) with a fixed maximum number of predictions (mAP@k, $k = 50$) is used to calculate the prediction accuracy.

In this competition, we need to predict the courses and subgroups that users will purchase in the test set. We have tried multiple models and methods to model the relevance between the occupations, interests, and recreations of users and the information about courses, to compare the similarity between all courses and those courses that users have purchased (content-based filtering), and to find similar users in the training set and compare the similarity of their courses (collaborative filtering). In addition to modeling the relationship between users' information and courses' content, we also analyzed the statistics of the number of times courses were purchased. From this analysis, we found that free and discounted courses were the most popular. We also observed that there were some courses published after the time interval of the training set data. These courses require special consideration since they are not covered by the training set data, and new courses often have discounts.

Ensemble learning combines the predictions of multiple learning algorithms to achieve better performance than any single algorithm could achieve on its own. In this competition, we explored the use of ensemble learning. We integrated various models and methods into the ensemble model and modified the weights of each model and method in order to capture the diverse features that different methods model. The experimental results and performance on Kaggle both showed that our ensemble model outperformed the other models we proposed and achieved high scores on the leaderboards.

3 Related work

3.1 ALS

Alternating Least Square (ALS) is an optimization algorithm frequently used in Matrix Factorization-based recommendation systems. The typical usage of ALS is to predict the score for missing entries in a sparse matrix where rows are usually users and columns are items to recommend (courses in our case) by matrix factorization. The input of ALS is a sparse matrix where non-empty entries indicate a bought history of an item by a user. To predict the likelihood of a user buying an item, we can simply check the corresponding entry in the output matrix from ALS.

3.2 BM25

Best Matching 25 (BM25) is a series of ranking functions for information retrieval, which is often used by search engines. BM25 method ranks a set of documents by their relevance to a given query, based on Term Frequency (TF) and Inverse Document Frequency (IDF).
TF represents the number of times the query terms appear in a document, the higher the more relevant. Whereas IDF measures how commonly the query terms appear in different documents, which indicates the amount of information the query terms provided. The lower the IDF the less important a query term is.

3.3 CKIP Transformers

CKIP Transformers contains the traditional Chinese transformers models and a set of NLP tools provided by the Chinese Knowledge and Information Processing (CKIP) Lab.
CKIP Lab also provides BERT-based pretrained Chinese language models and NLP task models. The NLP tools include word segmentation (WS), part-of-speech tagging (POS), and named entity recognition (NER).

3.4 Sentence Transformer

Sentence-BERT [3] (Reimers et al., 2019), abbreviated as SBERT, is a model based on BERT that extracts sentence embeddings. Its siamese structure is composed of two BERT with tied weights and pooling layers afterward to derive fixed-size embeddings. SBERT outperforms BERT embeddings (including CLS vector and average embedding) and has a higher inference speed. With its comprehensive documentation and codes, finetuning and running inference can be accessed easily.

3.5 Ensemble Learning

Inspired by the first prize winner of both tracks of KDD Cup 2011 [1] (Chen et al., 2012), we use ensemble learning to combine the predictions of multiple models to make more accurate predictions than any of our individual models. In this paper, we discovered that an ensemble model (blending) can significantly improve overall performance by leveraging the diverse aspects or information provided by multiple individual models. Based on this, we decided to use the ensemble method in our competition.

4 Approach

For course prediction, each method predicts the scores of courses (a higher score indicates that the user is highly likely to purchase this course), then takes the top-50 as the prediction outcome.

For subgroup prediction, most methods do not directly predict the scores of each subgroup, but rather predict the courses first and then convert the outcome of course prediction to subgroup prediction. For the predicted course rank, the conversion process just simply takes the subgroup of higher rank courses as the predicted subgroup. The repeated predicted subgroups with lower course ranks are ignored.

4.1 ALS

We first built a sparse matrix where each row indicates a seen user and each column indicates a course. If we know a user bought a course in training data, we filled the corresponding entry with 1, and leave the rest of the entries empty. Then, we use the ALS algorithm provided by Implicit to train on the previously built sparse matrix. Let the output matrix of ALS be M. To predict the likelihood of a seen user u buying a course i, we simply checked $M[u][i]$. Therefore, to find and rank possible courses that a user u would buy, we would take u-th row of M, i.e.

$M[u]$, and sort the indices i's by value in $M[u][i]$ in descending order, and the resulting sorted indices i's would be the prediction.

In addition, on subgroup domain, other than previously mentioned method of converting course result into subgroup result, we also implemented native ALS on subgroup domain. However, we found that its performance is far less then the one of converting from course result. Therefore, in the remainder of this report, ALS method on subgroup refers to the one that converts from courses result.

4.2 BM25

We used Rank-BM25 to calculate the score between a query and documents. The assumption is that the more relevant a user's information to a course is, the more likely the user will purchase this course.

For a set of query terms Q to represent the information of a user u, we used the fields `gender`, `occupation_titles`, `interests`, `recreation_names`. The `gender` field contains the options: *male*, *female*, and *other*. We converted those options into ”男”, ”女”, ”其他” because the information of the courses is in Chinese. The options in the `interests` field are the concatenation of groups and subgroups. To consider the relevance of both groups and subgroups to a course, we separated them into two different terms and removed the repeated group terms. For the `occupation_titles` and `recreation_names` fields, each option was derived from the multiple choice questions, so we simply added them into Q.

$$Q = \{q_1, q_2, \ldots, q_n\} \tag{1}$$

where q_i is a query term from the processed fields.

For the corpus D composed of courses information, the fields `course_name`, `groups`, `sub_groups`, `description` were concatenated to represent a course, then we used the tool `CkipWordSegmenter` to segment the concatenated fields into the words w_k of a document d_j.

$$D = [d_1, d_2, \ldots, d_m] \tag{2}$$

$$d_i = [w_1, w_2, \ldots, w_{l_j}], \ \forall d_j \in D \tag{3}$$

where l_j is the length of document d_j.

We chose the `BM25Okapi` algorithm to compute the scores of the courses $d_j \in D$ to a given query Q.

$$\begin{aligned}\text{score}_{j,Q} = {} & \Sigma_{q_i \in Q}\ IDF_{q_i} \\ & \times \frac{TF_{j_{q_i}} \times (k_1 + 1)}{TF_{j_{q_i}} + k_1 \times (1 - b + b \times \frac{|d_j|}{avgdl})}\end{aligned} \tag{4}$$

$$avgdl = \frac{\Sigma_{d_j \in D}\ |d_j|}{|D|} \tag{5}$$

where $\text{score}_{j,Q}$ is the score of d_j given Q, IDF_{q_i} is the inverse document frequency of the term q_i to D, $TF_{j_{q_i}}$ is the term frequency of q_i to d_j, $k_1 = 1.5$ and $b = 0.75$ are the parameters of `BM25Okapi`, $|d_j|$ represents the length of d_j, and $avgdl$ represents the average length of documents in D.

To predict the courses a user would purchase, we ranked the courses with $\text{score}_{j,Q}\ \forall d_j \in D$ and took top-50 as our predictions. For subgroup prediction, we took the subgroups of the top 50 predicted courses as our subgroup prediction.

4.3 Predict by Course Similarity

In this approach, we first concatenated the following fields of courses into a sentence: course name, groups, subgroups, topics, description, and target group. In addition, spaces and HTML tags are removed from these fields. Then, we put the resulting sentence into the tokenizer of our model, and the tokenized sentence will be then fed into the model. The output of the model, i.e. token embedding, would be passed to the mean pooling layer. The output of the mean pooling layer would be an embedding representing the input sentence, i.e. a course. To calculate the similarity between 2 courses, we can simply convert the raw data into course embedding by the above method, and then calculate the cosine similarity between the embedding of these two courses. Therefore, to predict how likely a seen user would buy a course, we can calculate cosine similarity between the bought courses and other courses.

Our approach works like this: For every bought course b_i of the user, we will find 50 most similar courses that the user had not yet bought. We would use $c_{i,j}$ to represent j-th similar course of bought course b_i. Intuitively, if a course is similar to multiple bought courses, then it is more likely to be bought in the future. Therefore, after we found all $c_{i,j}$ for all bought courses b_i, we would combine repetitive $c_{i,j}$'s. Formally, if there exists $c_{i,j} = c_{i',j'}$, then we remove $c_{i',j'}$ from our list, and the similarity score $s_{i,j}$ of $c_{i,j}$ would become $s_{i,j} + s_{i',j'}$. Finally, we would re-rank all $c_{i,j}$'s by $s_{i,j}$'s, and the result would be our prediction. In addition, to make our similarity score between 0 and 1, we normalized the resulting similarity score. Note that since we need to know bought courses, this approach only works on the seen domain.

4.4 Predict by Course and User Similarity

In order to cope with the problem that the previous approach in section 4.3 works only on the seen domain, we came up with another approach, which allowed us to find similar seen users first, and then use the same method in section 4.3 to predict for unseen users. Similarly, in order to find the similarity between two users, we used the same method mentioned in section 4.3 to convert a sentence of a user into an embedding. As for the sentence representing a user, we first sorted the interest field so that the interests under the same group are located near each other, and then separated the group and subgroups into distinct words. Afterward, we've added some prompts like "我的職業是...", "我喜歡...", and "我常常..." in front of fields of occupation, interests, and habits. The reason why we do this is that pretrained transformer models are usually trained with contextualized texts, and thus we have to add context to those fields of users by adding these prompts, which is a common technique when using pretrained language model.

Our approach works like this: For a user u to predict, we first use a similar method of finding similar courses in section 4.3 to find the 25 most similar seen users. We would denote k-th similar user as u_k. Then, we run the approach of section 4.3 for every u_k's. We would denote j-th course prediction of u_k as $c_{k,j}$. Similarly and intuitively, if a course is likely to be bought by multiple similar users, then our target user u is more likely to buy it. Therefore, after we found every $c_{k,j}$ for every u_k, we combine repetitive $c_{k,j}$'s. Formally, after we found all $c_{k,j}$ for all similar user u_k, if there exists $c_{k,j} = c_{k',j'}$, then we remove $c_{k',j'}$ from our list, and the similarity score $s_{k,j}$ of $c_{k,j}$ would become $s_{k,j} + s_{k',j'}$. Note that the similarity score $s_{k,j}$, in this case, will be the similarity score us_k between u_k and u times course similarity score $cs_{k,j}$ from the output of applying the approach in section 4.3 on u_k, i.e. $s_{k,j} = us_k \times cs_{k,j}$. Finally, we would re-rank all $c_{k,j}$'s by $s_{k,j}$'s and get our prediction, which is the same as what we do in the approach of section 4.3. In addition, we also normalize the resulting similarity score. Note that this approach can be applied to both seen and unseen domains.

4.5 User-Course Similarity

Our user-course similarity model predicts in accord with the cosine similarity between user embedding and course embedding.

The embedding is directly produced by an SBERT model finetuned with the training dataset. The course and user input sentences are processed as in sections 4.3 and 4.4. We finetune the model with its MultipleNegativesRankingLoss, where each pair of (user, bought course) in the training data serves as a positive pair and all others are seen as negative pairs. As stated in the research of this loss function [2], the training goal is to minimize the approximated mean negative log probability.

The top 50 courses are directly ranked by the cosine similarity of each user-course pair produced by the finetuned model.

4.6 Statistical Model

A statistical model is a model that uses observations from the training data set to extract meaningful insights or patterns. These patterns are then used to make predictions about future courses. We chose to use this technique because we found that free courses are popular within the data set.

Based on our analysis of the training data set, we noticed that users often purchase free courses. As a result, our model prioritizes free courses and arranges them in order of date, starting with the most recent. Furthermore, we noticed that courses occurring close to the time frame of the prediction tend to have higher enrollment. To account for this, our model prioritizes these courses and arranges them by price, with higher-priced courses coming first. This order is used because courses that are closer in time often have promotional pricing. For the remaining courses, we arranged them based on the number of times they were purchased.

The probability of each prediction is determined linearly based on the ranking of the courses in the prediction:

$$\hat{r}_{ui} = k \cdot r + b \tag{6}$$

where r is the ranking of the user in $\hat{r}_u$ and k and b are constants. In this case, we set $b = 0.02$ and $k = -0.0002$ to generate the probabilities for each course. The probability generated by this model does not take into account the popularity of the courses, which could potentially improve its accuracy. However, since many popular courses were not available for purchase during the training period (probably because they haven't been published), we were unable to incorporate this information in our analysis.

It's worth noting that one of the courses was never purchased in the validation set. To improve the accuracy of our validation score (mAP), we exclude this course from the predicted courses $\hat{r}_u$ when making predictions for the validation set. However, this method was not used when predicting the test data set. Moreover, we also observed that some courses have a higher number of purchases based on additional data from a public API. However, since this data is external to our training data set, we did not include it in our predictions, even though it could significantly improve the performance of our statistical model and ensemble model.

4.7 Ensemble

Our ensemble model combines the predictions of multiple models using a weighted averaging method, which does not require training a meta-model. While this technique is easy to implement, it may not always perform as well as other ensemble methods like boosting, bagging, and stacking. This is because it does not consider the unique characteristics of the individual models and simply combines their predictions without taking their relationships into account. With weighted averaging, the final prediction is the weighted average of the predictions made by the individual models. This method can be used for regression problems, where the final prediction is the mean of the individual model predictions.

To obtain the final probability $\hat{r}_{ui}$ for each user-course pair (u, i), we generate probabilities for each model and combine them using weighted averaging. The weights we use are the mAP scores of the models in the validation or test data set.

$$\hat{r}_{ui} = \frac{\sum_{k=1}^{n} w_k \cdot p_k}{\sum_{k=1}^{n} w_k} \tag{7}$$

where w_k represents the mAP score of model k, and p_k is the probability of user u purchasing course i, as predicted by model k. The models we included in the final ensemble model were chosen based on their performance, but we also included some models that were worse than others in our experiments for the sake of diversity.

The performance of the ensemble model can be seen in Table 1. We will discuss various variant techniques of ensemble models in more detail in section 5.3.

4.8 Post Processing

To improve the accuracy of our predictions, we performed post-processing on our final output by removing courses that were purchased in the training and validation datasets. For each user in the test dataset, we compiled a list of courses that were purchased in the training and validation datasets. If a course appeared on this list, it was removed from our prediction for the test dataset. We did this because the time period of the test dataset is after the training and validation datasets, so our predictions for the test dataset may include courses that a user has already purchased. By removing these courses, we were able to improve the overall performance of our models.

5 Experiments

5.1 Performance

The results of all of our methods on the validation set are presented in Table 1, and the results on the test set on Kaggle can be found in Table 2. Bold text states for the best model and scores.

5.2 Fine-tuning SBERT

Since sentence embeddings play a crucial part in our prediction, we have tuned several versions of SBERT models. From table 1, we can see that finetuned models perform better.

5.3 Ensemble learning

To further improve the performance of our ensemble model, we implemented several ensemble variants including a model-based ensemble model and a powered-weighted ensemble model. In the model-based ensemble model, rather than combining models with different parameters directly, we first calculate the weighted average of each type of model and then calculate the weighted average of the probabilities of all models. This model can improve the performance of predictions for seen and unseen courses but may reduce the accuracy of predictions for subgroups. In contrast, the powered-weighted ensemble model raises each weight to the power of 5.5 to adjust the weights of the individual models:

$$\hat{r}_{ui} = \frac{\sum_{k=1}^{n} w_k^{5.5} \cdot p_k}{\sum_{k=1}^{n} w_k^{5.5}} \tag{8}$$

This technique can improve the performance of predictions for seen and unseen subgroups but may decrease the accuracy of predictions for courses. In our experiments, we found that using a power of 5.5 was the most effective at improving the performance of predicting subgroups. However, we do not have a mathematical explanation for this result.

Additionally, we heuristically choose the weights of each model to make the performance better. The performance of the ensemble models on the validation set is in Table 1.

6 Discussion

6.1 Seen vs Unseen Tasks

In our methods, the seen and unseen tasks aren't treated very differently; the main difference is the removal of bought course in the seen tasks. However, we've discovered that a method's properties affect its performance on these tasks.

The methods that take advantage of seen data, undoubtedly, performed better on seen tasks. For example, predicting by user-user then course-course similarity (section 4.4) makes use of training data to find similar users. This method scored higher in the seen tasks, which are on average 34.0% and 11.9% increases in the course and topic prediction. It is noteworthy that the more epoch the model is tuned, the wider the performance gap is. Our statistical model (section 4.6) also showed this tendency with increases of 73.5% and 49.8% on average in the course and topic prediction. Since the purchase patterns were observed in the training data, its predictions are more related to the seen domain.

On the other hand, methods related to information retrieval performed better on unseen tasks. For BM25 (section 4.2), the unseen tasks outperform the seen tasks by 39.3% and 20.8% on average separately. The user-course similarity method (section 4.5) can be perceived as information retrieval due to its query-response property. We have not reached a conclusion of this phenomenon's cause yet, but perhaps the "seen users" and "unseen users" are different beyond the received data, such as how long they have joined Hahow.

In the end, our ensemble model performed equally well on both seen and unseen tasks. We believe this result implies that it effectively chose the predictions and mixed the advantages of each method together.

6.2 Language Model

We have totally 3 approaches that run on language models, which are predicted by course similarity in section 4.3, predict by course and user similarity in section 4.4, and user-course simi-

Table 1: Results on validation set

	Seen		Unseen	
Model	Course	Topic	Course	Topic
ALS	0.03751	0.22236		
BM25	0.02814	0.19029	0.04698	0.23606
CS0	0.03764	0.23952		
CS10	0.05829	0.24542		
CS15	0.11335	0.27131		
CS18	0.11553	0.27203		
US0	0.02505	0.24379	0.02132	0.22981
US10	0.04363	0.20274	0.02866	0.21459
US15	0.11569	0.24127	0.07554	0.18302
US18	0.13965	0.26889	0.09515	0.20680
UC0	0.01657	0.18946	0.02248	0.23410
UC3	0.03707	0.20490	0.06412	0.27869
UC10	0.03492	0.20352	0.06054	0.28012
Statistics	0.13757	0.22559	0.07367	0.14450
Ens	0.16447	0.32625	0.14754	0.30974
Ens Avg	0.17125	0.31415	0.17358	0.30974
Ens 5.5	0.16188	0.33041	0.10746	0.32661
Ens Heu	0.18364	0.33087	0.18067	0.33449

larity in section 4.5. From table 1 on seen domain, we can find out that methods of predicting by course or user similarity perform better than methods of user-course similarity. We think the possible reason for this is that predicted by similar courses or users can take more advantage of seen information, and thus perform better on the seen domain. However, when it comes to the unseen domain, the user-course similarity method does not have that much performance difference between the method of user and course similarity, especially on the unseen topic domain, which has significantly higher performance than the method of user and course similarity.

In addition, we can also find out from the table 1 that all 3 methods using a language model do not perform well when directly using zero-shot transfer, and all of them perform better after finetune. On the method of user-course similarity, we find out that it performs the best when it is finetuned for 3 epochs, however, on the methods using user or course similarity, it performs the when it is finetuned for 18 epochs.

6.3 Statistical

In this section, we will explain our reasoning behind the ordering of the courses and the potential factors that influenced this decision.

First of all, we prioritize free courses. Our analysis of the training data revealed that users tend to purchase free courses. It makes sense that users would be more likely to purchase free courses without the added pressure of financial cost. Next, we prioritize courses that are occurring in the near future. Our analysis of the training data showed that these courses tend to have higher enrollment. We believe this is due to promotional pricing or marketing efforts for these courses. Therefore, cheaper courses may not necessarily be the most popular, as they may only have a small discount. On the other hand, more expensive courses that offer larger discounts and promotions may be more attractive to users. As

Table 2: Performance on Kaggle test set

	Seen		Unseen	
Model	Course	Topic	Course	Topic
ALS	0.02751	0.10691		
BM25	0.03250	0.18469	0.05290	0.23749
CS10	0.07598	0.30349		
CS15	0.11150	0.27980		
US10	0.06154	0.21342	0.05103	0.25591
US15	0.11318	0.24367	0.10175	0.20595
UC	0.03316	0.21414	0.06789	0.28662
Statistics	0.15220	0.27436	0.10331	0.19807
Ens	0.17852	0.33475	0.17798	0.34371
Ens Avg	**0.18192**		**0.20664**	
Ens 5.5		**0.34222**		**0.34813**
Ens Heu	0.19643		0.21880	0.35826

a result, our model orders the courses in the near future by price, with higher-priced courses coming first. Additionally, we divided the near future into smaller time slots for a more detailed analysis. This decision was based on heuristics, as we do not have a specific technique to determine the optimal time slot size for slicing. Finally, we order the remaining courses by the number of purchases they received. It is logical to assume that courses with more purchases are more appealing to users.

6.4 Ensemble Method

We have tested two methods for predicting subgroups through the ensemble model. The first one is to take the weighted averaging on the subgroup predictions from each model, *i.e.* do ensemble on the subgroup predictions. The second one is to ensemble the course predictions from each model, then convert the course prediction by ensemble model to subgroup prediction.

We compare two methods on the validation set, from Table 3, method 1 (directly do ensemble on subgroups) outperforms method 2 (converted from the course prediction) on both seen domain and unseen domain. To improve the performance of method 2, we also tried different weights for the ensemble model. w_c represents that the weights are determined by the performance of individual models on course prediction, and w_s represents the weights derived from the performance of individual models on subgroup prediction. We intuitively thought that using w_s on the subgroup prediction would have better performance than using w_c to do subgroup prediction. We could confirm our intuition from Table 3 We could also observe that method 1 was still better than two variants of method 2, so we chose method 1 to do subgroup prediction by ensemble model.

We thought that the models we implemented above could capture different features about whether a user was willing to purchase courses. For those models similar to information retrieval, such as BM25 (section 4.2) and user-course similarity (section 4.5), they calculate the correlation between user information and course information. For those which compare similarity between courses and users, like section 4.3 and section 4.4, they use the assumptions that users are more likely to purchase courses that are similar to the courses they purchased or similar to the courses purchased by other users with high similarity. For the statistical model, it gains insight into the relationship between a course's price, re-

Table 3: Results of different subgroup prediction methods by ensemble model

Method	Seen Topic	Unseen Topic
Method 1	0.32625	0.30974
Method 2 with w_c	0.28702	0.26614
Method 2 with w_s	0.30140	0.29993

leased date, number of purchases, and the likelihood of future purchases, which happen to be the features not captured by other models.

Our ensemble method integrates features from different models and increases the utilization of information. By adjusting the weights of different models, the ensemble model can benefit from various features to predict better.

7 Conclusion

7.1 Conclusion

In the Hahow challenge, we proposed six methods and an ensemble model fusing their predictions based on their individual performances. The methods ranged from statistics to language model embeddings, exploiting features in distinct aspects. Our results showed that the ensemble model achieves significant improvements by combining the strengths of each method.

7.2 Future Work

As NLP technologies progress, sentence embedding extraction has improved significantly. The SBERT model we used was the state-of-the-art model in 2019, but now ranks 34 on the STS benchmark. Therefore, newer or more advanced models could be tested out. In addition, semantic-level adjustments could also be done. Prompts and interest field selection were implemented in our methods, thus trying different prompts and combinations of chosen fields might also lead to better performance.

During our work, we discovered that language models may miss certain information that can be supplemented by statistical models. To make the most of the available data, we applied statistical tools to gain additional insights more than our statistical model.

Additionally, we believe that our model could benefit from further consideration of the time attribute, as the number of purchases is significantly influenced by the time period, such as promotions for newer courses (as discussed in 4.6). Using techniques like time deviation may improve the model's performance.

Furthermore, we have additional models that utilize matrix factorization techniques that were not used in this competition. If we integrate these models into our ensemble model, we believe that we can extract more information from the dataset and produce more accurate predictions.

Finally, while our ensemble model uses simple weighted averaging, there are other techniques such as stacking, bagging, and boosting that can significantly improve the performance of the ensemble model. Unfortunately, we are unable to train on the validation set. However, we believe that by making some modifications to our ensemble model configuration and training it on the training dataset, we can improve its performance.

8 Work Distribution

Our work distribution of this project is shown in table 4.

9 Appendix

9.1 Appendix A. Terms

- u: user ID
- i: course ID

Table 4: Our work distribution in this project

Member	Work
李沁柔	User-course Similarity, User similarity, Slide, Report
楊冠柏	Statistical method, Ensemble learning, Slide, Report, Figures and Tables
易冠廷	ALS, Course Similarity, User Similarity, Presentation video editing, Slide, Report
林語涵	BM25, Ensemble Learning, Slide, Report

- r_u: courses that user u will purchase
- $\hat{r}_u$: predicted courses that user u will purchase
- $\hat{r}_{ui}$: The predicted probability that user u will purchase course i
- p_i: the price of course i
- Q: The query terms.
- q_i: Each query term in Q.
- D: The corpus of information of courses.
- d_j: The document representing a course in corpus D.
- w_k: A word in the documents.
- $c_{i,j}$: course i's j-th similar course.
- $s_{i,j}$: course similarity between course i and j.
- u_k: user u's k-th similar user.

References

[1] Po-Lung Chen, Chen-Tse Tsai, Yao-Nan Chen, Ku-Chun Chou, Chun-Liang Li, Cheng-Hao Tsai, Kuan-Wei Wu, Yu-Cheng Chou, Chung-Yi Li, Wei-Shih Lin, Shu-Hao Yu, Rong-Bing Chiu, Chieh-Yen Lin, Chien-Chih Wang, Po-Wei Wang, Wei-Lun Su, Chen-Hung Wu, Tsung-Ting Kuo, Todd G. McKenzie, Ya-Hsuan Chang, Chun-Sung Ferng, Chia-Mau Ni, Hsuan-Tien Lin, Chih-Jen Lin, and Shou-De Lin. A linear ensemble of individual and blended models for music rating prediction. In *Proceedings of the 2011 International Conference on KDD Cup 2011 - Volume 18*, KDDCUP'11, page 21–60. JMLR.org, 2011.

[2] Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun-Hsuan Sung, Laszlo Lukacs, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. Efficient natural language response suggestion for smart reply. May 2017.

[3] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.