MLBD MOOC Course Recommendation

Chenkai Wang
323452
chenkai.wang@epfl.ch

Zhijie Song
334036
zhijie.song@epfl.ch

Kang Fu
321120
kang.fu@epfl.ch

ABSTRACT

In this project, we conduct course recommendations based on the open MOOC site, XuetangX. Followed previous research, we adopt a new suggested method, Self-Attentive Sequential Recommendation model (SASRec) and compare its performance with the Bayesian Personal Ranking Model(BPR) model and the Non-personalized model as the benchmark. We adopt Hit Rate@10 and NDCG@10 to evaluate and conclude that the SASRec model achieves the best performance among the three models. Besides, we also discussed the main factors that influence the performance of the models.

ACM Reference Format:
Chenkai Wang, Zhijie Song, and Kang Fu. 2021. MLBD MOOC Course Recommendation. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION AND RESEARCH QUESTIONS

Online courses have become much more popular in recent years with the advantage of convenient long-distance education. As suggested, we adopted XuetangX, a massive open online course (MOOC) platform that offers online courses in multiple disciplines and certificate and degree programs. And it is meaningful to explore users' behaviors. Our potential target audience is the online course platform, who intends to improve their course recommendation system.

From the macro perspective, our goal is to predict the next course to recommend to each student, based on their needs or interests. Here we use students' past chosen courses as the representation of interests. Nowadays sequential dynamics are the kernel of many modern recommendation systems. The aim of sequential recommender systems is to combine personalized models of user's historical activities with users' recent actions. To narrow down our research goal, we built three models for our course recommender task. These models are:

- **Non-Personalized Recommendation Model**: This is our baseline that recommends courses based on courses' popularity. Since it is non-personalized, all the students will receive the identical course recommendations.
- **Bayesian Personal Ranking (BPR) Model**: BPR is a classic method for learning personalized rankings from implicit feedback, based on matrix factorization. [5]
- **The Self-attentive Sequential Recommendation Model (SASRec)**: In 2017, a new sequential model Transfomer in Natural Language Processing [8] achieved remarkable performance and efficiency for machine translation tasks. Inspired by this method, we follow the experimental steps from Wang [3]. We apply the self-attention mechanisms to sequential recommendation problems. From former research, the SASRec is proven to perform significantly better than MC/CNN/RNN-based sequential recommendation methods.

Then we analyze their outputs and compare the performance of the models. Hit Rate@10 and NDCG@10 are adopted to compare different models.

2 DATA DESCRIPTION AND EXPLORATORY ANALYSIS

The original dataset is acquired from MOOCData (http://moocdata.cn/data/course-recommendation), with the format of Figure 1. The dataset has 6 subjects, which are 1) the ID number of students; 2) time point of enrolling in the course; 3) the course's index number (unique for courses); 4) the course's name; 5)the course's type; 6) the ID number of course type.

	stu_id	time	course_index	name	type	type_id
0	0	2017/6/1 9:02	0	中国建筑史（上）	艺术·设计 历史	20.0
1	0	2017/7/4 7:52	1	外国工艺美术史	艺术·设计	20.0
2	0	2017/7/4 7:55	2	心理学概论	社科·法律	13.0
3	0	2017/7/20 5:35	3	经济学原理	经管·会计	10.0
4	0	2017/11/14 5:36	4	公司金融	经管·会计	10.0
5	0	2017/11/14 6:15	5	创业102：你能为客户做什么?	创业 经管·会计	1.0
6	0	2017/11/29 3:09	6	e时代的教与学——MOOC引发的混合式教学	教育	9.0
7	0	2018/3/21 7:25	7	党的十九大精神概论	社科·法律	13.0
8	1	2016/12/21 12:09	8	e时代的大佬师——慕课教师的修炼心法	教育	9.0
9	1	2017/1/5 8:05	9	大学物理近代物理	物理	18.0

Figure 1: Overview of the dataset

There are more than 82535 students and 1302 courses in total. And the types of courses are 79, with 23 type ID. Obviously, there is difference between the number of Types and the number of type IDs. The reason is the *type_id* only represents the ID number of the first class type.

We compute the length of users' chosen courses. And the result is shown by Figure 2. The average length is 5.55 with a standard variance of 5.66. The minimum of users' chosen courses is 3, and the maximum is 398. Additionally, the upper quartile is 6. So when the course's length is 10, we can cover the majority of users' actions.

Then we compute the distribution of *type_id*. Also, the result is viewed by Figure 3. Computer Science has the most chosen times, which is more than 110,000 times. And the second type is medical health, which is more than 60,000 times. The third one and the fourth one are law and engineering separately.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

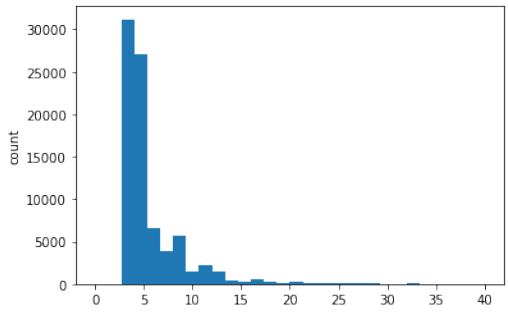

Figure 2: Distribution of users' course length

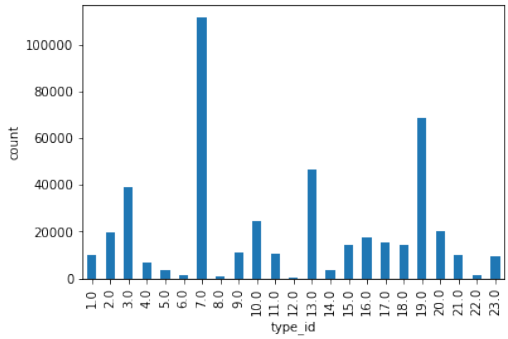

Figure 3: Distribution of *type_id*

Table 1: Frequency of Special Characters

Rank	Type ID	Type name	Chosen times
1	7	Computer Science	111542
2	19	Medical health	68895
3	13	Law	46372
4	3	Engineering	38911
5	10	Accounting	24394
6	20	Design	20185
7	2	Electrics	19840
8	16	Foreign language	17659
9	17	Literature	15163
10	18	Physics	14531

Similarly, we explore the type number of different users. The average length is 2.89 with a standard variance of 1.87. This means on average each user chooses courses from nearly 3 types. The minimum is 1, and the maximum is 21. Additionally, the upper quartile is 3.

3 THE PROPOSED APPROACH

3.1 Motivation in model selection

We have adopted three different models. One is the baseline, which has been chosen as non-personalised recommendation model. This

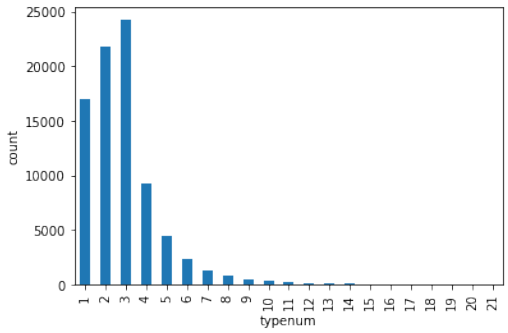

Figure 4: The number of chosen types for each user

model is easy to implement and used as comparison. As for personalised recommendation, we use BPR and SASRec for the two main areas of methods for recommendation systems.

- Matrix Factorization (MF) methods is a very important kind of models in recommendation systems, which seek to uncover latent dimensions to represent users' preferences and items' properties, and estimate interactions through the inner product between the user and item embeddings [1][4]. From this line, we select BPR model.
- Recently, due to their success in related problems, various deep learning techniques have been introduced for recommendation [7][9][2]. The SASRec model is a good option in deep learning.

In the following model realization part, only two features in the dataset, *user_id*, and *course_id*, are used in the model. Let $U = u_1, ..., u_n$, be a set of users and $C = c_1, ..., c_n$ be a set of courses in the MOOC platform. For each user u, given his historical enrolled courses $E^u = e_1^u, ..., e_{t_u}^u$, we are focusing on recommending the course at the next time point t_{u+1}. So we deal with the relative time instead of the absolute time [6].

3.2 Split the dataset

We split the dataset into three parts: training set, validation set and test set and randomly sampled k users in each batch. And the detailed splitting method can be viewed in Fig. 5.

The last item of each user is selected to be the test set. The second last item is selected to be the validation set. All the remaining items are taken as the training set.

3.3 Non-Personalized Recommendation Model

The non-personalized recommendation system is the most basic form of recommendation system. As its name suggests, this model is not personalized and do not take the users' interest or preference into account. In our project, the selected indicator for non-personalized recommendation is the popularity of the courses, which is calculated by the number of course selected by students. Figure 6 shows the most popular 10 courses and those courses will be recommended to all students.

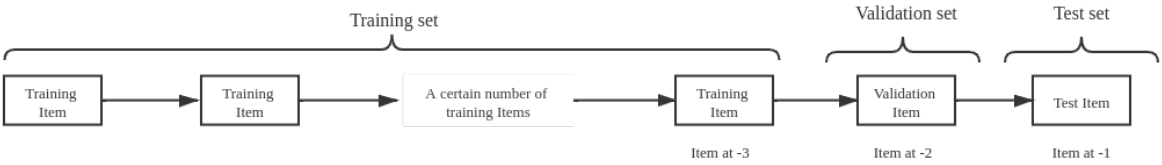

Figure 5: Partition of the dataset

course_index	name	type	type_id	choosen_times
17	数据结构（上）	计算机	7.0	11058.0
14	C++语言程序设计基础	计算机	7.0	9050.0
10	计算机科学和Python编程导论	计算机	7.0	8292.0
26	网络技术与应用	计算机	7.0	8096.0
18	操作系统	计算机	7.0	6874.0
191	运营概论（微慕课）	工程 经管 会计	3.0	5949.0
15	生活英语听说	外语	16.0	5901.0
290	网络、群体与市场	计算机	7.0	5726.0
148	面向对象程序设计（C++）	计算机	7.0	5699.0
17	数据结构（上）	计算机	7.0	5684.0

Figure 6: The most popular 10 courses

3.4 Bayesian Personal Ranking Model

Bayesian Personalized Ranking (BPR) is derived from personalized ranking, which provides users with item recommendations of a ranked list of items. The ranked list of items is calculated from the users' implicit behavior. BPR is based on matrix factorization. It is a mature recommendation method, which has already applied on movie recommendation and online shopping recommendation. Take course recommendations as an example, the users' implicit behavior included the past courses students have chosen or their interest. The chosen courses can be seen as positive datasets while the remaining courses can be a mixture of negative and missing values. Typically, the course recommenders output the personalized score X_{ui} (u is a student and i is a course) based on the preference of the student for the courses, and courses are sorted from the predicted score. The machine learning model of course recommenders provides the training data by giving pairs (u, i) ∈ S as a positive class label and all other combinations in (U × I) \ S as the negative one. Here all the negative user-course pairs are replaced by 0.

Bayesian Personalized Ranking combines the Bayesian analysis using the likelihood function for $p(i >_u j|\theta)$ and the prior probability for the model parameter $p(\theta)$. BPR's goal is to maximize the posterior probability, which is proportional to the multiplication of the likelihood function and the prior probability [5]:

$$p(\theta|i >_u j) \propto p(i >_u j|\theta)p(\theta)$$

$p(i >_u j|\theta)$ is the likelihood function, it captures the individual probability that a user really prefers course i over course j. We compute this probability with the form:

$$p(i >_u j|\theta) = \sigma(r\hat{u}ij(\theta))$$

where: σ is the logistic sigmoid function:

$$\sigma(x) = 1/(1 + e^{-x})$$

And $r\hat{u}ij(\theta)$ captures the relationship between student u, course i and course j, which can be further decomposed into:

$$r\hat{u}ij = r\hat{u}i - r\hat{u}j$$

Here we define the prior probability $p(\theta)$ as the normal distribution with zero mean and variance-covariance matrix $\Sigma\theta$.

$$p(\theta) \sim N(0, \Sigma\theta)$$

According to [5], to reduce the number of unknown hyperparameters, $\Sigma\theta$ is set as $\lambda_\theta I$ where λ_θ is model specific regularization parameter.

$$\begin{aligned}
BPR-Opt &= \prod_{u,i,j} p(i >_u j|\Theta)p(\Theta) \\
&= ln(\prod_{u,i,j} p(i >_u j|\Theta)p(\Theta)) \\
&= \sum_{u,i,j} ln\sigma(\hat{r}_{ui} - \hat{r}_{uj}) + lnp(\Theta) \\
&= \sum_{u,i,j} ln\sigma(\hat{r}_{ui} - \hat{r}_{uj}) - \lambda_\Theta\|\Theta\|^2 \\
&= \sum_{u,i,j} ln\sigma(\hat{r}_{ui} - \hat{r}_{uj}) - \frac{\lambda_\Theta}{2}\|x_u\|^2 - \frac{\lambda_\Theta}{2}\|y_i\|^2 - \frac{\lambda_\Theta}{2}\|y_j\|^2 \\
&= \sum_{u,i,j} ln\sigma(x_u y_i^T - x_u y_j^T) - \frac{\lambda_\Theta}{2}\|x_u\|^2 - \frac{\lambda_\Theta}{2}\|y_i\|^2 - \frac{\lambda_\Theta}{2}\|y_j\|^2 \\
&= \sum_{u,i,j} ln\frac{1}{1 + e^{-(x_u y_i^T - x_u y_j^T)}} - \frac{\lambda_\Theta}{2}\|x_u\|^2 - \frac{\lambda_\Theta}{2}\|y_i\|^2 - \frac{\lambda_\Theta}{2}\|y_j\|^2
\end{aligned}$$

In general, machine learning will always minimise some parameters. Thus we flip all the signs and make the maximization into minimization.

$$\text{argmin} \sum_{u,i,j} -\ln\frac{1}{1 + e^{-\left(x_u y_i^T - x_u y_j^T\right)}} + \frac{\lambda_\Theta}{2}\|x_u\|^2 + \frac{\lambda_\Theta}{2}\|y_i\|^2 + \frac{\lambda_\Theta}{2}\|y_j\|^2$$

Finally, the objective function has been derived. Since above function is differentiable, gradient descent can applied to optimise this problem. However, considering the size of all the possible triplet (u,i,j) is too big, stochastic gradient descent algorithm will be a better choice to shorten the training time and still keep a relative good performance. The gradient can be calculated like the following:

$$\frac{\partial}{\partial x_u} = \frac{1}{1 + e^{\left(x_u y_i^T - x_u y_j^T\right)}} \cdot \left(y_j - y_i\right) + \lambda x_u$$

$$\frac{\partial}{\partial y_i} = \frac{1}{1 + e^{\left(x_u y_i^T - x_u y_j^T\right)}} \cdot -x_u + \lambda y_i$$
$$\frac{\partial}{\partial y_j} = \frac{1}{1 + e^{\left(x_u y_i^T - x_u y_j^T\right)}} \cdot x_u + \lambda y_j$$

3.5 Self-Attentive Sequential Recommendation model

Wang-Cheng Kang and Julian McAuley [3] developed a sequential recommendation model based on Transformer, which is a novel architecture that aims to solve sequence-to-sequence tasks (e.g. machine translation). The structure of the model is shown in the figure 7.

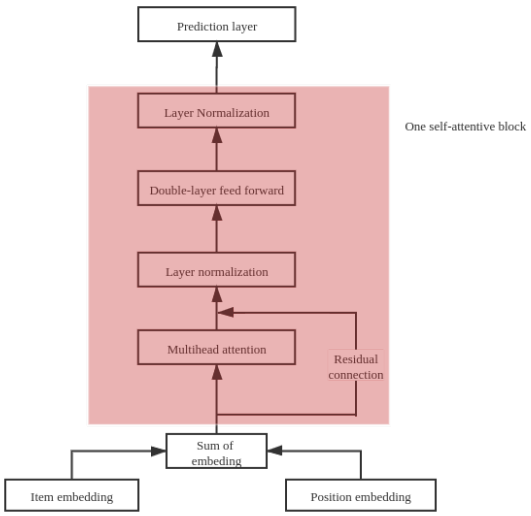

Figure 7: Structure of the model

3.5.1 Structure of the model. The self-attentive sequential recommendation model mainly consists of the following parts.

Fixed-length sequence: We should make sure that the sequence (i.e. a user's past courses selection) that are fed into the model should be one of fixed-length N. In other words, if a user has selected more than N courses, we only keep the most N recent courses he/she took in the sequence. On the other hand, if the user has selected less than N courses, a series of padding items (In our implementation, 0 is used as the padding item) and prepended to the sequence to ensure that the length of the sequence is N. To silence the padding (i.e. not involved in the calculation) in the model, we should also add masks on the paddings.

Embedding: We should create an embedding matrix for the input sequence, $E \in R^{I \times d}$, where the latent dimension is d and the cardinality of the course set is I. d is a hyperparameter that can be optimized. It is made up with two components: item embedding and position embedding.

- Item embedding: for the fixed-length sequence of each user, we retrieve the corresponding embedding and form a new matrix $\hat{E}_1 \in R^{N \times d}$.
- Position embedding: However, the matrix $\hat{E}_1$ does not contain any information on time series (i.e. the Chronological order of the courses taken). To solve this problem, the positions in the sequence are also transformed into embeddings and to be retrieved as $\hat{E}_2$ for each user.

In the end, we do the element-wise addition of the two matrices $E = \hat{E}_1 + \hat{E}_2$ then take the sum E as the input for the model.

Self-attentive layers: In the original Transformer model, the attention is calculated as

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d}})V \quad (1)$$

Where, $Q = EW^Q$, $K = EW^K$, $V = EW^V$ are the query, key, and values matrices; and matrices W^Q, W^K, and W^V are the learnable parameters. We use torch.nn.MultiheadAttention to simulate the self-attentive mechanism in the model.

Feed-forward neural network: Since self-attentive layers aggregate the embeddings with the adaptive weights, which is essentially a linear model. To endow the model with nonlinearity, a two-layer feed-forward network is applied to the output of the self-attentive layer, where ReLu is adopted for the non-linear transformation.

$$F = ReLu(S_i W^{(1)} + b^1)W^{(2)} + b^2 \quad (2)$$

Self-attentive blocks: In practice, the self-attentive layer and the feed-forward network together form a self-attentive block. To create a more complex model and thus improve its capability, we can add more self-attentive blocks. At the same time, we should pay attention to the overfitting problem while increasing the complexity and some techniques to prevent overfitting (e.g. dropout) should be adopted.

Layer normalization: Layer normalization is used to normalize the inputs across all features. It is applied to the outputs of both self-attentive layers.

Prediction layer: In the prediction layer, a multiplication of the output of self-attentive blocks and the embedding matrix of the items which have not been selected by the user is conducted to obtain the score of each item (i.e. the score of the item to be selected at the next time step). This step is essentially performing the calculation to find the most "similar" item which should be present at the next time step. The ranking of recommendations can be generated by sorting them in a descending order of scores.

3.5.2 Model Training. In the training, we adopt the binary cross entropy loss, of which the mathematical expression is

$$loss = \sum_{S^u \in S} \sum_{t \in [1,2,...,n]} [y_t log(\sigma(r_t)) + (1 - y_t) log(1 - \sigma(r_t))] \quad (3)$$

where r_t is the output of the rating for the items that have not been chosen before, n is the size of the item set for the user. It is similar to the cross-entropy loss where the binary cross entropy loss applies a sigmoid function to the rating prediction to transform it into probability.

The training is implemented in the following manner. For example, assume there are 4 items in the training set, in the first step, the item at time 1 is taken into the model and used to predict the rating of the item at time 2. For this purpose, we should prepare one

positive item(the ground truth) and one negative item (randomly selected from the items that are not chosen at time 2), calculate the ratings and feed them into the loss function. We do the same calculation at each step as the time progresses until we reach the end of the training sequence.

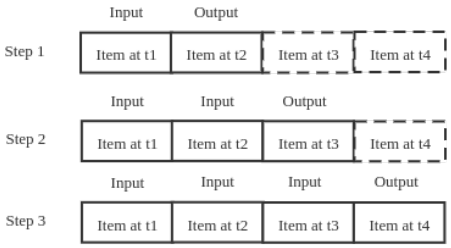

Figure 8: Implementation of the model training

4 EXPERIMENTAL EVALUATION

4.1 Evaluation metrics

Evaluation metrics for recommender systems have evolved with time. Initially, the accuracy of predicted ratings was used as an evaluation metric for recommender systems. Netflix even started a competition to find the algorithm which could improve upon their accuracy by 10 percent for one million dollars. But they later realized that the accuracy of predictions had no relation with whether someone will be interested in watching the predicted movie because a person wants to see new movies that they will like to see, so a list of top-n movies should be put in front of users and we should measure how they react to these recommended movies.

To evaluate the performances of these models, we adopted the following two metrics to evaluate the performance of the two models.

- Hit Rate (HR):
  The first concept is hit rate. The basic idea is that we generate a list of top n recommendations for a user and compare them the test courses. If they match then increase the hit rate by 1, do this for the complete training set to get the hit rate.
- Normalized discounted cumulative gain (NDCG):
  The second evaluation metric that we adopt is normalized discounted cumulative gain(NDCG). It's a feature to measure the quality of recommendations, which takes the position of recommendation courses into account. Because we have learned this concept in MLBD class, we don't use too many words to explain it here.

Consistent with intuition, the bigger cutoff we choose the greater the model performance is. However, from a practical and scientific angle, the cutoff is chosen as 10 for evaluation in this project. Because 10 courses are the common size for displaying on the online MOOC platform. To be precise, we also compare the performance varying different cutoff values in the parameters-fining part.

4.2 Experiment with SASRec

In this section, we conducted plenty of numerical experiments with the self-attentive sequential recommendation model. Hyper-parameters with optimal performance are selected.

4.2.1 Learning curve. As a starter, we ran the model and plot the learning curve with a batch size of 256. The result of our model's loss varying epochs is as shown in the Fig.9. It can be observed that when epochs reach 50, the learning curve begins to flatten; when epochs reach 150, we consider it convergence. In the next few experiments, We tune hyper-parameters using the validation set, with the epoch of 150.

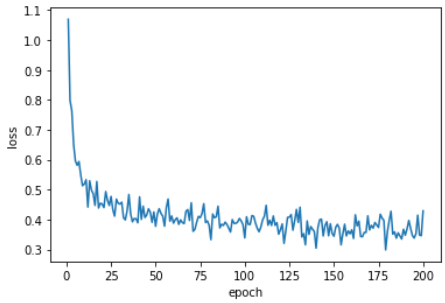

Figure 9: The learning curve

4.2.2 Length of training sequence. The first hyper-parameter that comes to our mind is the length of the training sequence. It takes a crucial role in two concerns. Firstly, to keep a fixed length N for each training sequence, we prepend 0 as the paddings to the users who have selected less than N items. These paddings can be properly handled by adding masks to the paddings to make sure that they play no effect on the model. Secondly, the more items as the prior knowledge we feed into the model, the better performance we would achieve. However, a quite long sequence may lead to that the model may consider out-of-date information, which means it considers the courses selected too long ago. We test the model varying the length of training sequence, shown in Fig. 10.

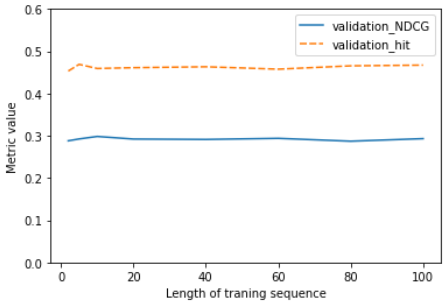

Figure 10: The NDCG and Hit rate varying with the length of training sequence

From this experiment, we can draw the following conclusions.

- From the figure above, we can tell the model's NDCG achieves the best performance, when the length of the training sequence is 10. The main reason is that in our MOOC dataset, the mean length of users' course sequences is 5.55, and the 75% quantile is 6. So that a longer training sequence never contributes. However, the optimal performance doesn't excel.
- When the training sequence is created in an unreasonably large length, the performance does not deteriorate. We can conclude that the added paddings barely affect our results after the properly handled masks.

4.2.3 Number of blocks. From Fig. 11, we define the self-attentive block as a combination of a multi-head attention layer and a feed-forward network, which indicates the complexity of the model. The evaluation scores under the different number of blocks are shown in figure 11.

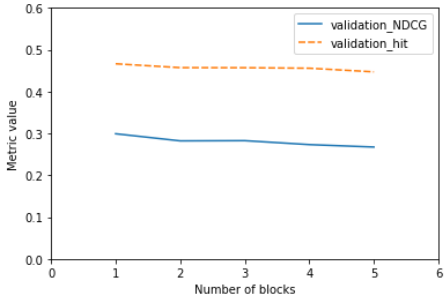

Figure 11: The NDCG and Hit rate varying with the number of blocks

It is clearly shown that as the number of blocks increases, the performance declines slightly. It could likely be a sign of over-fitting that a too complex model may "learn" not only the pattern but also the noise in the training set. From this figure, we can conclude that with one block the model can achieve the best performance. The observation is also in line with the fact that the majority of the users have relatively a short item sequence.

4.2.4 Number of heads in the multihead layer. As mentioned by Wang and Julian [3], the embedding space can be decomposed into several sub-spaces in the multihead layer to extract the features respectively, which enhances the model's power. This is what the number of head works on. We experimented with different number of heads and see how it affects the model. The result is shown in figure 12.

It shows an insignificant impact under different number of heads, which is consistent with the observation by Wang and Julian [3], where they apply the model in a number of settings, for example game recommendations. This might owe to the small dimensions of hidden units in our problem (d = 512 in Transformer), which is not suitable for decomposition into smaller subspaces. And this will be discussed later in the model comparison part.

After the above experiments, we selected the most possible optimal hyperparameter of our SASRec model. To be specific, we

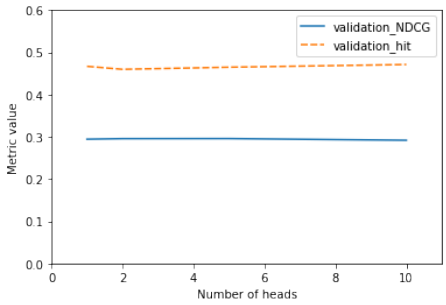

Figure 12: The NDCG and Hit rate varying with the number of heads

choose 150 epochs to compare the performances during the parameter fining steps. And we use the length of the training sequence as 10; the number of blocks as 1; the number of heads as 1. In the next part, we compare the performance between our three suggested models.

5 DISCUSSION AND IMPLICATIONS

As mentioned in the previous study, the SASRec model is good at sequential recommendation with long item sequences. We have discussed the selection of main hyperparameters and found that when we increase the complexity of the model, we will not obtain a better prediction. That is mainly because the sequence in the dataset is rather short, where 75% quantile is 6. We expect that if the model is applied to a problem which has longer sequences, a more complex model will be suitable.

Furthermore, BPR model is hard to catch the chronological order into the model. We have tested that, if we do not select the most recent course as the test while take one test course randomly, it can achieve higher Hit rate and NDCG.The main difference between the two models lies in the ability to capture the time series feature.

Here we conduct the model comparison among our three models. First we investigate the relationship between the hit rate (NDCG) and the number of recommended courses(cutoff). From Figure 13 and 14, it can be seen that the hit rate and NDCG will increase when the number of recommended courses is getting larger. The Reason is that there is a higher probability to hit as the number of recommended courses increases. SASRec model has a very good performance while BPR has a similar hit rate with baseline. In terms of NDCG, BRP is better than baseline.

Then we also compare the influence of latent dimension of BPR and SASRec model to hit rate and NDCG [10]. The latent dimension of BPR model is the latent factor used by matrix factorization and the latent dimension of SASRec model is the hidden unit of each layer. The hit rate and NDCG will first increase then decrease when the latent dimension goes up. From the comparison, BPR can achievement a good performance when latent dimension is 30, while for SASRec is around 50.

6 CONCLUSIONS AND FUTURE WORKS

In this report, the main task is to predict the next course to recommend to each student, based on their needs or interests. Three

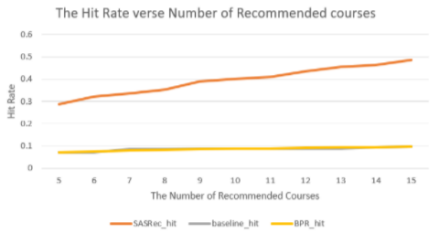

Figure 13: HR over number of recommended courses

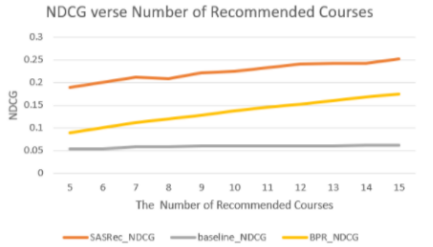

Figure 14: NDCG over number of recommended courses

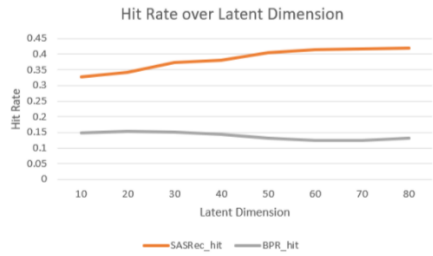

Figure 15: HR over latent dimension

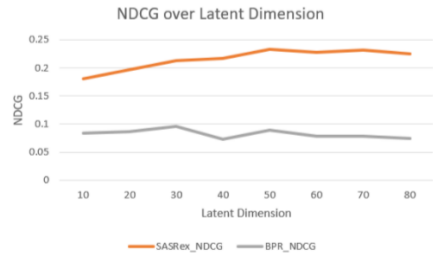

Figure 16: NDCG over latent dimension

models are selected: Non-Personalized Recommendation Model, Bayesian Personal Ranking (BPR) Model, The Self-attentive Sequential Recommendation Model(SASRec). The SASRec model achieve the best performance among the three models. Besides, we discussed the hyperparameter selection of the SASRec model and found out the optimal values for the several main ones.

For future work, we can add more models based on RNN to compare the performance with SASRec model as chronological information plays a vital role in such problems. Furthermore, we can also create more embeddings as the input for the SASRec, which contain additional information and thus contribute to the model. For this purpose, we may choose to do the summation for all the embbedings (the original operation) or design a new structure which can extract the useful information and combine them together.

REFERENCES

[1] B. Shapira F. Ricci, L. Rokach and P. Kantor. 2011. *Recommender systems handbook*. Springer US.
[2] Wang-Cheng Kang, Chen Fang, Zhaowen Wang, and Julian J. McAuley. 2017. Visually-Aware Fashion Recommendation and Design with Generative Image Models. *CoRR* abs/1711.02231 (2017). arXiv:1711.02231 http://arxiv.org/abs/1711.02231
[3] Wang-Cheng Kang and Julian McAuley. 2018. Self-Attentive Sequential Recommendation. arXiv:1808.09781 [cs.IR]
[4] Y. Koren and R. Bell. 2011. "Advances in collaborative filtering" in Recommender systems handbook.
[5] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian Personalized Ranking from Implicit Feedback. arXiv:1205.2618 [cs.IR]
[6] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. 2010. Factorizing personalized Markov chains for next-basket recommendation. In *WWW '10*.
[7] L. Yao S. Zhang and A. Sun. 2017. "Deep learning based recommender system: A survey and new perspective. arXiv:1707.07435 [abs]
[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. arXiv:1706.03762 [cs.CL]
[9] Suhang Wang, Yilin Wang, Jiliang Tang, Kai Shu, Suhas Ranganath, and Huan Liu. 2017. What Your Images Reveal: Exploiting Visual Contents for Point-of-Interest Recommendation. In *Proceedings of the 26th International Conference on World Wide Web* (Perth, Australia) *(WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 391–400. https://doi.org/10.1145/3038912.3052638
[10] Jing Zhang, Bowen Hao, Bo Chen, Cuiping Li, Hong Chen, and Jimeng Sun. 2019. Hierarchical Reinforcement Learning for Course Recommendation in MOOCs. *Proceedings of the AAAI Conference on Artificial Intelligence* 33, 01 (Jul. 2019), 435–442. https://doi.org/10.1609/aaai.v33i01.3301435