Abstract

In our thesis, we focus on the cold start user problem in the field of recommender systems. We propose a mechanism called “User Rejuvenation” and combined it with the deep learning model Denoising Autoencoder to solve the cold start user problem. The “User Rejuvenation” first choose representative items for each group of users, after that we randomly set the dimensions corresponding to representative items in user vector to zero score with lower probability, and randomly set the dimensions corresponding to non-representative items in user vector to zero score with higher probability. The main purpose for “User Rejuvenation” is to turn the non-cold start user vectors back to cold start user vectors for each group of users. After getting the group-specific cold start user vectors generated from “User Rejuvenation”, we can use them to train a Denoising Autoencoder model for the user group. When the training process is complete, the model will have capacity for restoring the cold start user vectors to non-cold start user vectors, and the recommendation is made through the predicted non-cold start user vectors.

**Keywords：**Deep Learning、Recommendation System、User Cold Start Problem、Denoising Autoencoder、Representative items selection

1. Introduction
   1. The importance of the recommendation system

With the explosive growth of information on the Internet, the dizzying selection of goods or services in various service platforms on the Internet has already put users in a dilemma that they cannot cope with. At this time, the recommendation system has become an indispensable existence that can effectively filter out a large amount of unnecessary information for users, and help users focus on the information they are concerned about. In addition to allowing users to quickly and accurately obtain products that meet their needs, the recommendation system can also improve user satisfaction and increase user stickiness to the platform through accurate recommendations, which brings platform vendors considerable benefits. Therefore, applying the recommendation system to practice is prevalent in many e-commerce platforms (Amazon) or audio-visual streaming platform (Netflix), that help companies improve user satisfaction and increase platform revenue. Taking Netflix as an example, their chief product officer Hunt said that more than 80% of Netflix movie viewings are generated through their recommendation system. As Netflix’s Gomez-Uribe and Hunt mentioned in [5], “If we can increase the user’s stickiness on the platform by improving the quality of the Netflix recommendation system, then Netflix can reduce $1 billion losses every year caused by customer churn.” The point of reducing losses by maintaining customer loyalty coincides with the point made by Forbes columnist Larry Myler, “For retailers, the cost of retaining existing customers is often much lower than the cost of seeking for new customers, and having a stable customer base can enable retailers to obtain higher customer lifetime value, as well as the business revenue is much easier to predict.” Under the same circumstances, for the Netflix platform, it is easier to reduce customer churn than to develop new customers, because new customers often decide whether to join the platform after browsing less than 20 movie recommendations. As a result, Netflix can only predict the preferences of new customers with very little information and time. Compared with recommending videos that meet their expectations from past ratings information of existing customers, it is much more practical to improve the quality of the recommendation system. From the case of Netflix, we can realize that a well-designed recommendation system can bring benefits to both users and service platform.

* 1. The difficulty and importance of the cold start user problem

Generally speaking, when constructing a recommendation system, a commonly used method is “collaborative filtering recommendation”. The concept of collaborative filtering recommendation is using the user group with similar product preferences to the target user who you wants to recommend to predict the products that the target users may also prefer. For example, when a customer buys a book in Amazon online bookstore, the bottom of the website will display “customers who bought this book also bought”, this is a classic case of collaborative filtering recommendation. However, there is a problem with collaborative filtering recommendation. While recommending products for a new user, because there is only a little rating information about the new user, it is difficult for us to find the user group with similar preference to the new user, which makes it impossible to recommend using collaborative filtering recommendation. Such a problem is called “new user cold start problem”. In the past, various methods have been proposed to solve this problem, one of which is the “content-based method” [16]. The content-based method combines the user’s additional information, such as the user’s profile, age, or gender, etc., and we can replace the original user rating vector without rating information with the user representation vector constructed by these additional information. Then, the recommendation can be completed by calculating the vector similarity such as cosine similarity to find the products that the users who are highly similar to the new user vector like. However, the content-based recommendation method must be based on the premise that we can obtain additional information from users. But in practice, we may not be able to obtain this information. Therefore, there is another method to solve the cold start problem of new users, called “representative-based method” [8][4][12]. The core concept of this method is to find “representative items” in the recommendation system, and if we can obtain some ratings of representative items from new users, then we can indirectly infer the relationship between new users and other general items through the relationship between representative items and general items to make recommendations.

As a result, our goal is to propose a framework that can effectively solve the cold start problem of new users, so that we can recommend items that meet the needs of new users, which will not only enhance the adhesion of new users on the platform but also increase customer engagement on the platform. In this way, new users can gradually get rid of the cold start state and get more accurate item recommendations, and finally form a positive interaction cycle between the user and the platform.

* 1. Apply deep learning models to solve cold start problems

In recent years, with the application of deep learning in various fields such as text, images, and speech, it has gradually matured and it could even surpass human performance on many tasks. Therefore, no matter in the industry or in the academic world, more and more people are beginning to study how to apply deep learning technology to the recommendation system. We call such application “the recommendation method based on deep learning”. In the industry, Covington et.al propose a deep learning model used to construct a recommendation system for Youtube videos. Covington’s method first uses a deep neuron network model with three hidden layers to find candidate videos that the user may like through user browsing history, user searching history, demographic information and other information. After finding the candidate videos, the related information between the user and the candidate videos, such as the number of times the user view the candidate videos, the time since the user last viewed the candidate videos, etc., is used to learn user’s preference for these candidate videos with another three-layer deep neuron network model. Cheng et.al also use deep learning methods to construct Google Play’s APP recommendations. Cheng’s method generates the user representation vector by a deep neuron network which takes the continuous data of the user, such as user age, the number of apps installed by the user, etc. and the discrete data of the user, such as the type of the device used by the user, apps the user has installed, etc. as model input. Then, the model will consider both the user representation vector and the target APP related information to decide whether to recommend the target APP for the user or not. In academia, given that a large number of research articles using deep learning methods combined with recommendation systems have been proposed and published in succession, the ACM RecSys included a deep learning based recommendation system as one of the conference themes since 2016 in order to promote the development of deep learning technology on the recommendation system. Seeing that so many industry technicians or scholars are rushing to invest in this emerging research topic, and we can know that it is an inevitable trend to develop a deep learning based recommendation system. Therefore, our research is based on the research of these predecessors and constructs an effective deep learning model to effectively solve the cold start problem of new users in the recommendation system. After a thorough investigation of the deep learning based recommendation system, we found that the operation concept of the deep learning model Denoising Autoencoder is very suitable for simulating the cold start problem. Take images as an example of how Denoising Autoencoder works. Some pixels in the picture will be randomly set to 0 to generate noise for the original picture, and then the noise-generated picture will be input into the deep learning model framework of the encoder-decoder. In the encoder stage, the input image vector is passed through more than one hidden layer to reduce the dimensionality of the image. In the decoder stage, the image is restored to the original dimensionality of the input image through more than one hidden layer. The ultimate training goal of the Denoising Autoencoder is to hope that the image interfered by noise can be restored to the image without noise interference at the beginning after passing through the encoder-decoder architecture. Therefore, in the field of image processing, Denoising Autoencoder is often used to remove noise from image data or to reduce the dimensionality of image data.

The correlation between Denoising Autoencoder and the cold start problem we want to solve is that we can imagine the original information-rich user input vector as a general user who has rated many items, and the process of generating noise is to simulate the gradual regression of general users to new users who have just entered the platform. However, the method proposed to generate noise for Denoising Autoencoder is usually applied to the image or speech data, and it is not directly applicable to the context of the recommendation system. Hence, we propose a method to generate noise for the input vector of the recommendation system. Below we will focus on the cold start problem of how to recommend items to cold start users, and briefly introduce our proposed method, which is mainly divided into two parts:

* 1. User Rejuvenation Stage

At this stage, we refer to the operating concept of Denoising Autoencoder to pass the user vector through our noise generation method, and then simulate the user vector back to the state when they just entered the platform. In order to get closer to the user cold start problem, we renamed the process of Denoising Autoencoder generating noise as the “User Rejuvenation Stage”. In the user rejuvenation stage, users will be grouped first, and then some representative items representing different groups of users will be selected. The selected representative items must have two characteristics: high popularity and large users’ rating entropy.

* 1. Encoder-Decoder Stage

Once the user vector used to simulate the cold start state in the rejuvenation stage can be obtained, the user vector is then passed through the encoder-decoder architecture to restore it to the original user input vector without noise generation. The whole process is similar to that we train a model which can restore cold start users to general users.

* 1. How we differ from current deep learning methods

At present, there are other studies that apply the Denoising Autoencoder model to the recommender system. In the current method, the method used when making noise interference to the input vector is nothing more than adding Gaussian noise, masking noise or salt-and-pepper noise. Although these noise interference methods can also make the trained model more robust in prediction and less susceptible to inference from input noise. However, the practical interpretability of these noise generation methods in the recommendation system is not so intuitive, because these methods are randomly cover some ratings in the user vector to zero. On the contrary, our method can retain the most important item ratings in the input vector by keeping the ratings of “representative items” when generating noise for input vector. Therefore, the practical interpretability of our method is more intuitive.

1. References

The reason why a large number of deep learning based recommendation methods have emerged and been studied in recent years is mainly due to the following two advantages of deep learning [18]: (i) Deep learning based methods can simulate more complex non-linear relationship between users and items through various non-linear activation function, such as sigmoid, tanh, and ReLU. Compared with methods that are not based on deep learning, such as matrix factorization can only simulate the simple linear relationship between users and items. Therefore, the method based on deep learning will obviously be closer to the reality. (ii) Deep learning based methods can effectively capture features from any type of input data, such as text data from user profiles, image data of items, or information from item advertising video. Having this advantage can save more time than using non-deep learning based methods, which requires manual feature engineering. However, using deep learning based methods can be supervised or unsupervised learning to automatically extract important information from original input, and it also enable us to combine various types of content information to make the recommendation results more precise.

In the view of the above two advantages, we investigated and organized the current recommendation methods based on deep learning. In addition, how to find representative items is also our research focus, so we will also organized some papers related to finding representative items at the end of this chapter. Lastly, all related papers are classified according to the following forms:

* 1. Recommendation system based on deep learning

The deep learning recommendation methods in this section is mainly divided into two stages. In the first stage, it is necessary to decide what form to represent the input user vector and item vector. In the second stage, a suitable neural network architecture must be designed so that the loss between the predicted rating and the truth rating can be as small as possible.

Just like in [3], two complementary methods are proposed to generate embedding vectors for users and items. One is called the Constraint Model (CM), which uses a set of users who have rated the same score on both item A and B to describe the relationship between them. Based on this idea, the correlation matrix R between all items can be obtained. Then, we can decompose the correlation matrix R to get two sub-matrices and . Lastly, we can form the embedding vector of item by means of taking out the i-th column vector of the sub-matrix and the i-th row vector of the sub-matrix respectively, and concatenate and together.

Another method of generating embedding vectors for users and items is the Rating Independent Model (RIM), which uses a set of users who have rated item A and item B but rated different scores to describe the relationship between item A and B. Based on this idea, the correlation matrix between all items can be obtained. Then, the embedding method for an item is the same as CM, and this approach will take into account that items will have different embedding vectors at different ratings.

[3] combines CM and RIM to generate item RIM vector, user RIM vector, item CM vector and user CM vector, respectively. These four vectors are called “View of current”. In addition, four vectors called “View of history” will be considered. In “View of history”, the average of the item vectors that the user has rated is used as the user vector (the item vector can be represented by CM or RIM), and the average of the user vectors who have rated the product is used as the item vector (the user vector can be represented by CM or RIM). After obtaining a total of eight vectors of “View of current” and “View of history”, CM embedding and RIM embedding will be passed through DNN and CNN respectively, and then all feature vectors outputted by DNN and CNN will be concatenated to form a input feature vector for the final deep neural network to predict user rating on items. The following is the neural network architecture diagram of [3]:

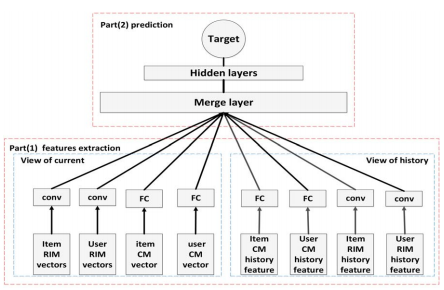


Figure 1. Neural Network Architecture Diagram of [3]

In another paper [6], the final result to be predicted by the recommendation system is not user ratings, but whether the user is interested in a certain item (if the prediction result is 1: the user is interested; otherwise: the user hasn’t noticed the item or the user is not interested in the item). When generating the user and item embedding, the one-hot encoding representation of each user and item is firstly passed through 4 sets of one-layer embedding layer to generate MF User Vector, MLP User Vector, MF Item Vector and MLP Item Vector. Then, these 4 vectors go through Generalized Matrix Factorization layer (GMF) and Multi-Layer Perceptron (MLP) layer respectively to learn the embedding of the linear and nonlinear relationship between user and item. Among them, the GMF layer multiplies the corresponding dimensions in the user and item vector, so as to simulate the operation of Matrix Factorization to capture the linear relationship between vectors. In the MLP layer, the user and item vectors are merged and undergo a series of nonlinear transformations to capture the nonlinear relationship between them. Finally, the model will consider the embedding of GMF and MLP at the same time to predict whether the user is interested in the item. The following is the neural network architecture diagram of [6]:

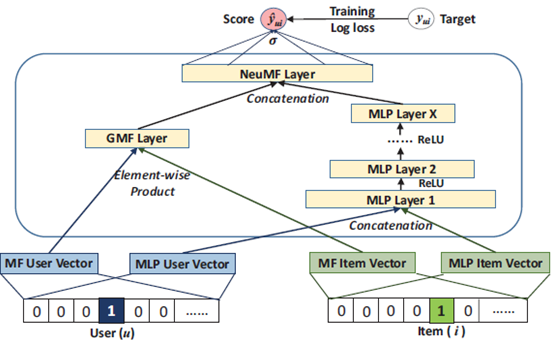
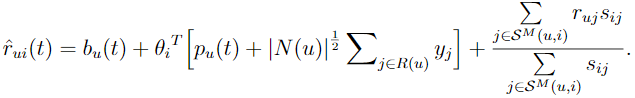


Figure 2. Neural Network Architecture Diagram of [6]

In addition to the above methods [3][6], another deep learning recommendation system is based on the Autoencoder architecture. This type of method can be divided into two categories: (i) Use Autoencoder to retrieve low-dimensional feature vectors of content information attached to users or items [16][14][9]. (ii) Use Autoencoder to predict user ratings on items [17].

In the application of the first type of Autoencoder, [16] proposed a framework suitable for predicting the ratings of two cold start items, complete cold start items (CCS) and incomplete cold start items (ICS). This framework is composed of deep learning model and timeSVD++. The deep learning model used is Stacked Denoising Autoencoder (SDAE). The SDAE model receives the textual description of the cold start item as input, and uses the hidden layer of SDAE as the item vector. Then, the item vector will be inputted into the timeSVD++ model, and take the popularity of the item over time and the rating pattern of the user over time into account, as well as the average rating of all the items that the user has rated to predict the user rating on the cold start item. The formula for predicting ratings of CCS items is as follows:



：user u’s rating on item i at time t

：the deviation of the rating given by the user u at time t

：the item vector generated after the textual description of item i passes through

the hidden layer of SDAE.

：the average vector of the items that user u has rated.

：the CCS item j and the top M non-cold-start items that are most

similar to it are weighted by the similarity of item i and item j according to

the rating of item given by user u.

For ICS items, the formula for predicting ratings of ICS items is as follows:



μ：average rating of all items

：the deviation of the rating received by item i at time t

：rating vector of ICS items

In the application of the second type of Autoencoder, [17] proposed the Collaborative Denoising Autoencoder (CDAE) model. CDAE receives the user’s preference value for each item (if the preference value is 1: the user likes the item; otherwise: the user doesn’t like the item or the user hasn’t noticed the item) and the user’s ID as model input. In the process of training the CDAE model, some dimensions of the input user vector are randomly set to 0, and the CDAE model needs to restore the input user vector disturbed by random noise to the original user preference vector. The following is the diagram of CDAE:

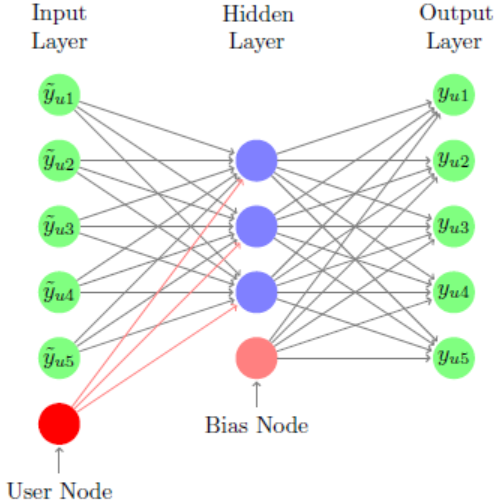


Figure 5. Neural Network Architecture Diagram of [17]

In addition, [19] proposed a new architecture using Autoencoder for recommendation, called Dual Autoencoder. The Dual Autoencoder will train the user vector and the item vector at the same time through two Autoencoder respectively, and use the inner product of the vector and the vector to predict the user rating on item v. The following is the diagram of Dual Autoencoder:

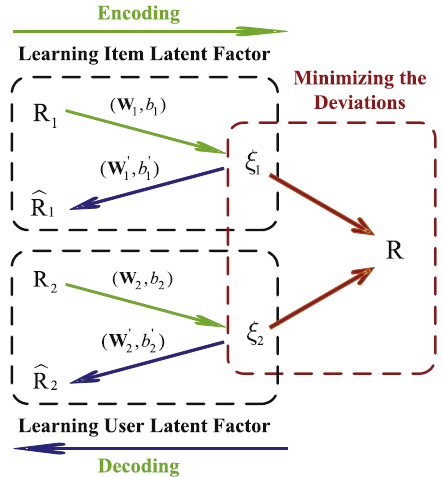


Figure 6. Neural Network Architecture Diagram of [19]

* 1. Cold start recommendation using deep learning

Although deep learning based recommendation methods are more capable than non-deep learning based methods, when new users or new items only have a small number of ratings, they still face the cold start problem. Therefore, there are many studies trying to use deep learning methods to solve the cold start problem. For example, [15] combines user preferences (ratings) and content information as model input, and use the dropout mechanism on the input vector. When training a model with cold start items or cold start users, because we do not have sufficient rating information about the items or users, so we set the corresponding item preferences or user preferences to zero vector. This approach is called “dropout”, the purpose is to hope that the model can use the complete information of content input to help model predict user ratings on items. The following is the diagram of [15]:

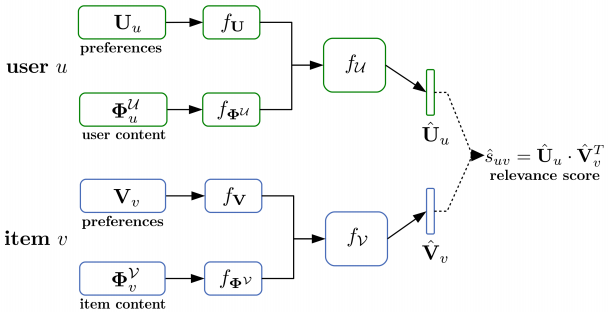


Figure 7. Neural Network Architecture Diagram of [15]

* 1. Representative items Mining

The deep learning methods used to solve the cold start problem in the previous section all use content-based recommendations. And there is another way to solve the cold start problem, which is based on the recommendation of representative items. Different from content-based recommendation need to collect additional information about users or items, the representative-based methods use the relationship between representative items and general items to predict the ratings of cold start users for general items.

[8] proposed the representative-based matrix factorization (RBMF), which decompose the rating matrix Y∈ describing relationship between users and items into two sub-matrices. The two decomposed sub-matrices are 、, that is , where the sub-matrix represents: the relationship between all n users and k representative items; the sub-matrix X represents: the relationship between k representative items and all m items. As for how to select k representative items, it is obtained through the maximal volume algorithm, which is used in the task of approximating the original matrix from the decomposed sub-matrices. The maximal volume algorithm can select the k columns that can best restore the original matrix. Therefore, we consider that the k items corresponding to these k columns are representative items. The following figure is a schematic diagram of RBMF selecting representative items:

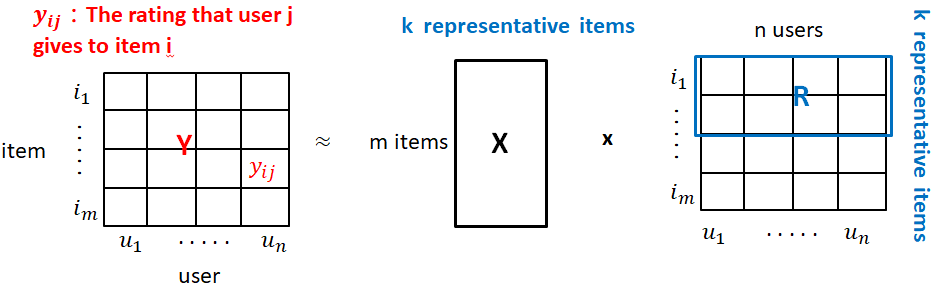


Figure 10. The RBMF of [8]

[12] wants to improve the RBMF method proposed in [8] to capture more precise new user preferences. Compared with RBMF only uses a set of representative items to represents all users, the LRBMF proposed in [12] will dynamically group users according to different “global representative items”, and then select “local representative items” according to each group.

1. Method

In this chapter, we will first introduce the operating concept of Denoising Autoencoder (3-1), and then explain how Denoising Autoencoder is applied to the problem of cold start recommendation for new users (3-2), after that we will introduce the method of this paper (3-3): how to simulate the user back to when the user just entered the platform through the “user rejuvenation” mechanism. Finally, we will explain how the Denoising Autoencoder is trained and how we use the trained Denoising Autoencoder to generate recommendations to users.

* 1. Denoising Autoencoder

Before understanding Denoising Autoencoder, we must first have a certain understanding of Autoencoder. Autoencoder is composed of two parts: encoder and decoder. In the encoder, a fully connected neural network is used as the function . maps the input vector to a lower-dimensional vector . The is called the encoder and it is represented by the following formula:

= .

The function parameter of is , is a matrix (), and is a bias vector with dimension equal to k, and is the sigmoid function.

In the decoder, a fully connected neural network is used as the function , and will map the low-dimension vector back to the with the same dimension as the input vector . The function is called the decoder and it is represented by the following formula:

=

The function parameter of is , is a matrix (), and is a bias vector with dimension equal to n, and is the sigmoid function.

The final loss function to be optimized for the entire model is the sum of errors between the input vector and the output vector , which can be expressed by the following formula:

.

Where is a function to calculate the square error of two variables. represents the input vector. represents the output vector of the input vector passed through the Autoencoder.

Denoising Autoencoder is a variant of Autoencoder. The training goal of the model is still to minimize the error between the input vector and the output vector. However, Denoising Autoencoder will generate noise in part of the dimensions of the input vector, and then input the vector with noise into Autoencoder to restore the input vector. The operation process of the entire Denoising Autoencoder is as follows:

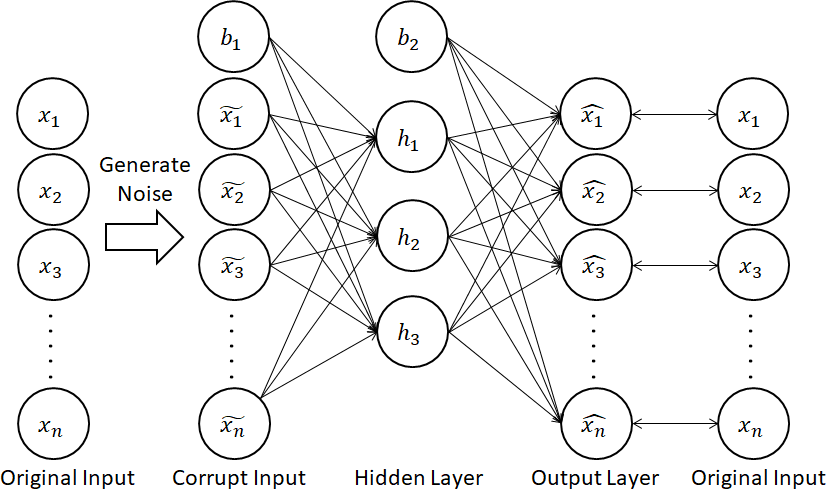


Figure. Denoising Autoencoder Model

There are three common noise generation methods: the first method is Gaussian noise. Gaussian noise is suitable when the dimension value of the input vector is a real number. It will be generated according to a normal distribution with the average is 0 and the variance is 1, and add the generated noise value to the dimension value in the input vector. The second method is masking noise. Masking noise will randomly set a part of the dimension value in the input vector to 0. The third method is salt-and-pepper noise. Salt-and-pepper noise randomly sets a part of the dimension value in the input vector to the maximum or minimum allowable value of the dimensionality. Take the application of Denoising Autoencoder in image processing as an example: each pixel of a color photo corresponds to a dimension value in a vector, and the allowable dimension value is between 0 and 255.

By adding the noise, the subsequent Autoencoder model can extract higher-level feature representations from the input vector during training, so that the model is not easy to change the prediction result when the input vector is interfered by a little noise.

Therefore, we summarize the training process of the Denoising Autoencoder below:

1. We will transform the input vector to the vector interfered by noise through the above-mentioned three noise generation methods or the noise generation method designed by ourselves. This process can be expressed as:

Where is the function generate noise to the input vector .

1. The vector interfered by noise is used as the input of Autoencoder, and the encoder will map to low dimensional vector . This process can be expressed as:

=

1. Use the decoder to map the vector back to the same dimension of the input vector to generate the output vector . This process can be expressed as:

=

The optimization process of the entire Denoising Autoencoder parameters and can use the general backpropagation. By gradually transferring the loss between the input layer and the output layer from the model output layer to the model input layer, the model parameters and can be updated.

* 1. Reasons for adopting Denoising Autoencoder model

The correlation between the Denoising Autoencoder model and the cold start recommendation is: masking noise will randomly set a part of the dimension value of the input vector to 0. This kind of noise generation method corresponds to the context of the recommendation system, which is to erase some item ratings from the user vector. Our goal is to propose a noise generation method suitable for the context of the recommendation system, thereby simulating the user vector as a cold start user vector that has just entered the platform, and the goal of subsequent Autoencoder model is used to fast forward the cold start user vector into the original user vector before the noise generation. In this way, the trained Autoencoder model can be regarded as a prediction model that can predict which items cold start users will prefer in the future.

Below we will use a diagram to illustrate the Denoising Autoencoder and the cold start user recommendation process. In order to get closer to the cold start problem of new users, we renamed the noise generation stage during Denoising Autoencoder training to “user rejuvenation stage”, and the details of the “user rejuvenation stage” will be introduced in the next section.

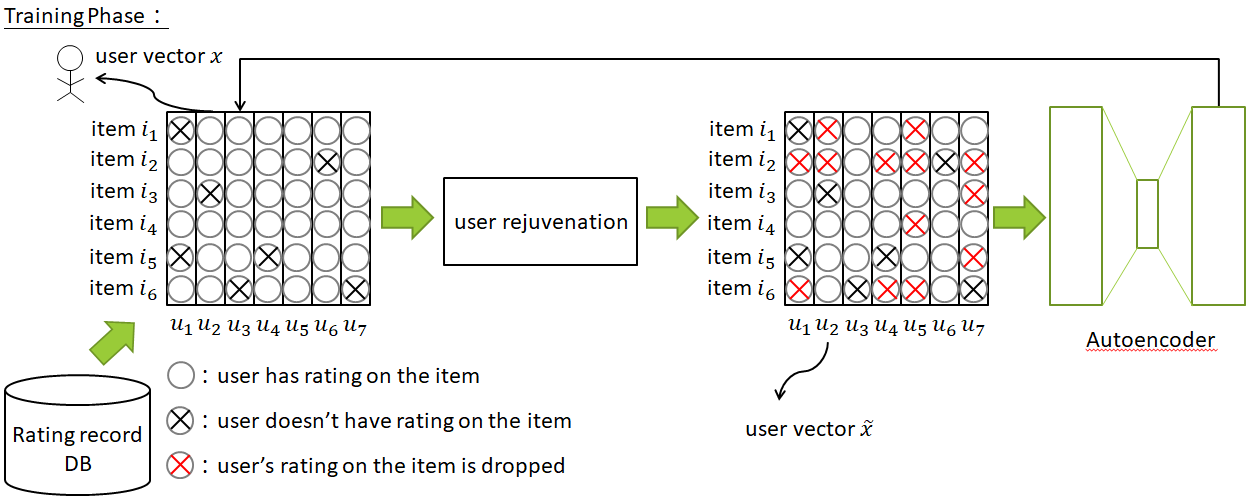


Figure . The training process of the Denoising Autoencoder

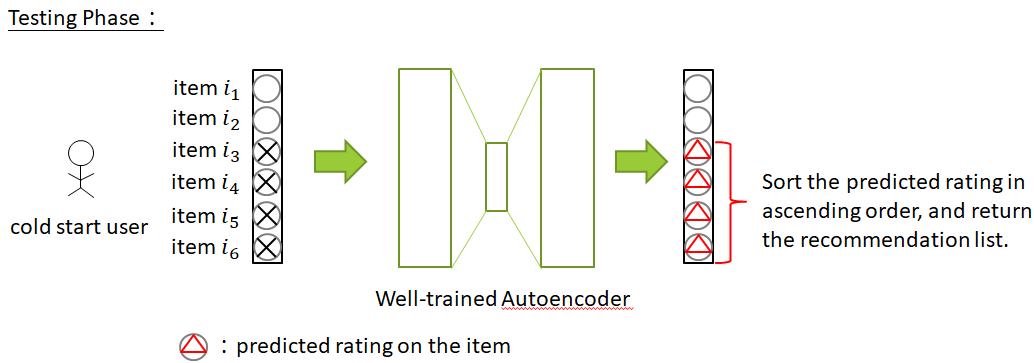


Figure . The test process of the Denoising Autoencoder

* 1. User Rejuvenation stage

In the view of the fact that the original Denoising Autoencoder randomly erases the rating information of some items in the user vector, the sparse user vector generated by this approach may not accurately reflect the cold start user state. Therefore, in our “user rejuvenation stage” when we restore the user to the cold start user state, we will not randomly erase the rating information of items. Instead, we will find some representative items, and remove the rating information of non-representative items as much as possible, so that the ratings of representative items are retained in the cold start user vector. Because we believe that when new users enter a service platform, they usually start to contact with representative items on the platform, and then they will explore non-representative items over item. The representative items we hope to select must have two characteristics: First, the representative items must be rated by most users (popularity), in other words, most users must be familiar enough with the item. In addition, the rating distribution of representative items must be sufficiently dispersed (rating entropy), which means that the more inconsistent the ratings of the items, then we can distinguish different types of users through the item.

Below we will illustrate the operation process of selecting representative items:

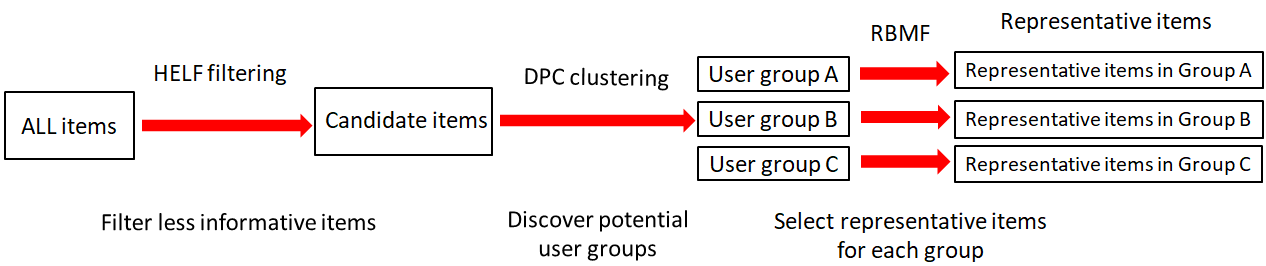


Figure. Flow chart of selecting representative items

It can be seen from the above flow chart that all items will pass through a processing path to form the final representative collection of items. In the following subsections, the processing operation between section will be explained.

* Harmonic mean of Entropy and Logarithm of Frequency (HELF)

When selecting candidate representative items, we use the HELF proposed in [10] as an indicator for initial screening of items. The HELF indicator takes two factors into account: one is the popularity of the items, which is how many users have rated the item; second is the rating entropy of the item, which means the dispersion of users’ opinion on an item. For example: if all user give a rating of 5 scores to an item, the entropy of the item is the minimum value of 0; on the contrary, if the ratings of all users are evenly distributed between 1 and 5, then the item can get the maximum entropy value. By integrating the harmonic average of popularity and rating entropy, we can obtain the HELF.

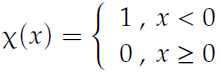
：the number of ratings received for the item , take the logarithm and divide by the number of users.

：the rating distribution received by the item and divide it by 5 scores.

We hope that the selected candidate representative items need to have a higher HELF value. In other words, the items need to be rated by most people, and everyone’s opinions on the item are more divergent. Therefore, we will first filter out items with a smaller HELF value to generate a set of candidate representative items.

* Density Peaks Clustering (DPC)

DPC [11] is a clustering algorithm based on regional density. By setting a predefined cutoff distance , we can calculate the number of other data points contained within for a data point . That is the local density of data points . Corresponding to the application context of the recommendation system, because our goal is to find potential user groups, each data point is a user vector, and each dimension in the user vector is represented by candidate representative items selected by the HELF. The calculation formula for the area density of user is as follows:



: the distance between user and user

: predefined cutoff distance

: If the distance between user and user is within , the local density will increase by 1.

is calculated as follows:

M：a collection of items rated by user and user

m：an item in the item collection M

：user i’s rating for item m

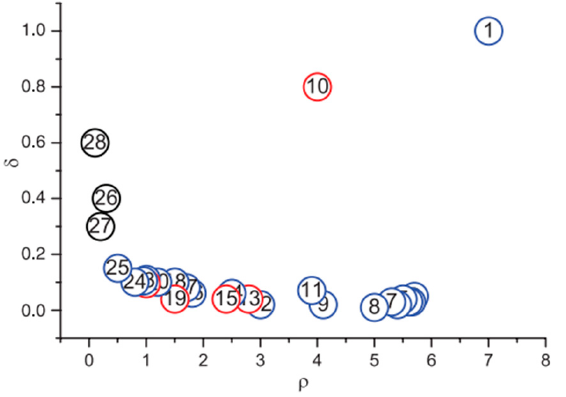
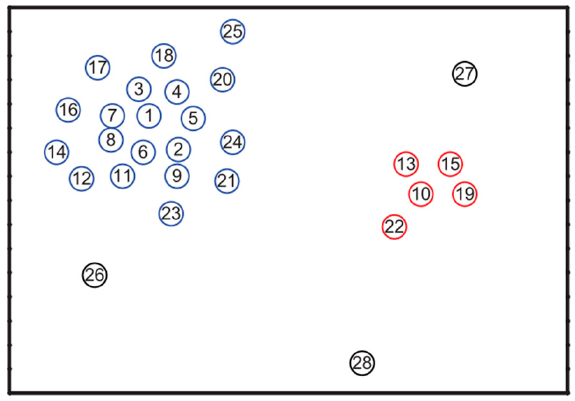
：user j’s rating for item m

After obtaining the local density , we can calculate the relative distance of the cluster center based on . The calculation method is: for user with the largest local density, its relative distance will be equal to the distance of the user farthest from user . For user whose local density is not the largest, the relative distance of user will be equal to the distance of the user with the smallest distance from user in the user set whose local density is greater than user . The calculation formula of can be expressed as the following two formulas based on whether is the maximum local density:

if is the user with largest local density.

if is not the user with largest local density

The DPC algorithm will use the local density of each user and relative distance of each user as the coordinate axes of the two-dimensional plane to generate a decision diagram. As shown below:



The figure on the left shows the distribution of 28 users in space, and the numbers on each user are numbered in descending order of local density, which means number 1 has the largest local density and number 28 has the smallest local density. The figure on the right that is the decision diagram plots the respective and of the 28 users on the coordinate plane where the x-axis is the local density and the y-axis is the relative distance. According to the decision diagram, we will choose the users who have large local density and large relative distance as cluster centers. In this example, user 1 and user 10 will be chosen as the cluster centers. After using the decision diagram to select cluster centers, we can assign other users to the cluster center which has closest distance to the user to complete the whole user clustering task.

* Representative-based matrix factorization (RBMF)

Suppose that using the DPC algorithm in our recommendation system, all users can be divided into three groups: A, B and C, which means that we can divide the rating matrix ∈ into three rating sub-matrices ∈ 、 ∈ 、 ∈ , where n represents the number of users in the recommendation system, m represents the number of candidate items that have been preliminarily screened by the HELF, and , , represent the number of users in different groups. We can use Maximal Volume Algorithm to find k columns (k representative items) in the three rating sub-matrices respectively, so that the disassembled sub-matrix can mostly restore the original matrix. The items selected through RBMF for each user group are called “representative items”. We will give the “representative items” a higher probability than general items in the “user rejuvenation stage” to be retained.

* 1. Cold start user recommendation

When making recommendations, we can regard the trained Denoising Autoencoder as a model can deduce the user’s future rating based on the user vector with only a few rating information. It means that we only need to input the user vector in the cold start state into the Denoising Autoencoder model, and the output vector of the Denoising Autoencoder is our predicted rating for each item. Therefore, we can recommend items to users from high to low according to the predicted ratings.

1. Experiments and Analysis
   1. Experimental dataset and evaluation metrics

* Experimental dataset

The MovieLens 1M dataset is a set of movie rating data provided by MovieLens users from the late 1990s to the beginning of the 21st century. The dataset includes user ratings for movies, movie genres, movie years, and demographic data about users. However, since the method of our paper focuses on user rating for movies, we only collect and organize the information of user ratings for movies in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Num of ratings | Num of movies | Num of users |
| MovieLens 1M | 1,000,209 | 3,900 | 6,040 |

Table. Statistics table of MovieLens 1M dataset

* Evaluation Metrics

In the performance evaluation stage of the recommendation system, we select the commonly used metrics to evaluate the ranking performance NDCG@k, precision@k, recall@k to review the performance of our proposed model, and introduce how these metrics are calculated in detail below.

* NDCG@k

The full name of NDCG [7] is Normalized Discounted Cumulative Gain, which is a measure of the quality of search engine ranking results. The concept behind NDCG is: if you can sort the more relevant results at the top, you can get a higher score. Therefore, applying the concept of NDCG to the context of the recommendation system, we would hope that the recommendation system can sort the users’ favorite movies in the front of the recommendation list. The following shows the calculation method of NDCG@k, where IDCG is the maximum DCG value that can be obtained when the recommendation system can perfectly sort the recommendation results:

k：top k sorted results

：the rating of the movie in the recommendation list

* precision@k, recall@k

In the context of the recommendation system, we will divide the movie into two categories according to the rating given by the user. If the rating of a movie is higher than the average rating of the user, we will regard the movie as a favorite movie; otherwise, if the rating of a movie is lower than the average rating of the user, we regard the movie as a movie that the user doesn’t like. At this time, precision@k represents when we recommend k movies to a user, how many movies out of these k movies are liked by the user. And recall@k represents how many of the user’s favorite movies appear in the first k positions in the recommendation list. The following shows the calculation methods of precision@k and recall@k:

* 1. Training process and experimental parameter setting
* Training process

In MovieLens 1M, users with more than 25 ratings will be regard as training users, and users with less than 25 ratings will be regard as test users, the division results obtained by this method are as follows: it can be seen that there are a total of 5549 users in the training set, and these 5549 users have overrated 3702 movies. In the test set, there were 491 users, and they only rated 1995 movies.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Training set** | | **Test set** | |
| Split methods | Num of users | Num of movies | Num of users | Num of movies |
| Cold Start split | 5549 | 3702 | 491 | 1995 |

Table 2.

When training the model, we use the training set to train the model. During the training process, the loss function to be optimized for our Denoising Autoencoder model is masked RMSE. The calculation method of masked RMSE is to calculate the error between the predicted ratings and the real ratings for the items that the user has actually rated. The schematic diagram and formula are as follows:

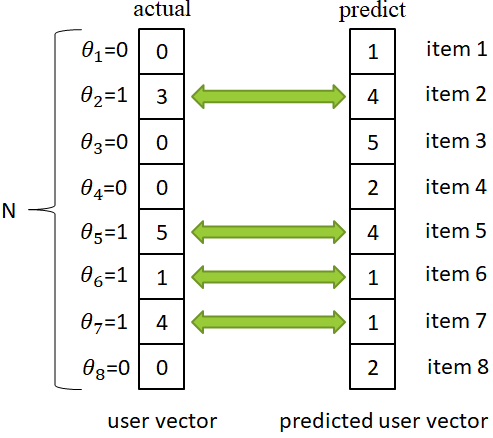


Figure. The schematic diagram of masked RMSE

,

Where N represents the number of all items, represents the user’s actual rating for item , represents the predicted rating for item generated by Denoising Autoencoder model. And , only when , = 1. Otherwise, = 0. Finally, we will update the parameters of the model through the commonly used optimization algorithm SGD (Stochastic Gradient Descent) for training neural networks. The following figure shows relation between training error versus training epochs while training Denoising Autoencoder model. It can be found that the training error reaches convergence at about 150 epochs, so we set the epochs of training to 150 times in subsequent experiments.

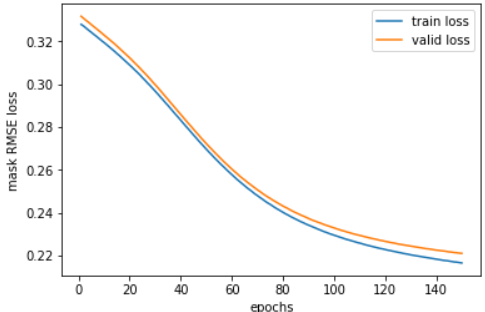


Figure . Diagram of training error and training epochs

When evaluating the performance of model training, we will randomly cover 20% of the movies’ ratings that each test user likes to 0 as the ground truth to be verified when calculating NDCG@k, precision@k and recall@k. The schematic diagram is as follows:

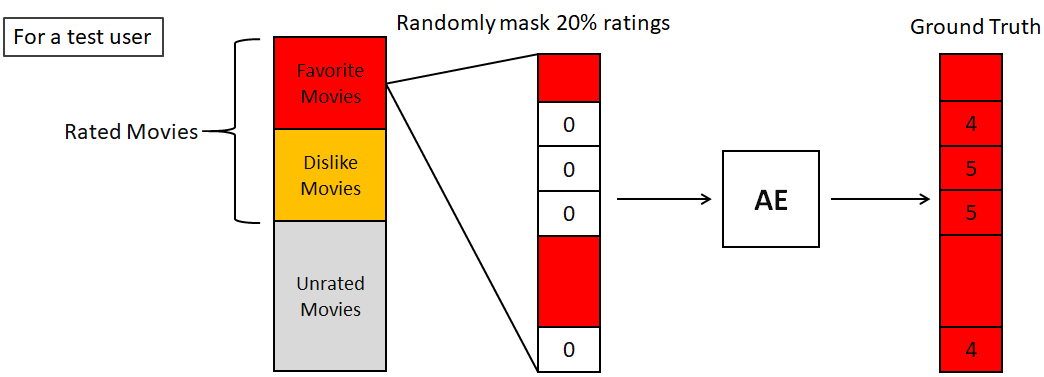


Figure.

* Hyper-parameters setting

Our model has three parts of hyper-parameters that need to be determined:

The first part is to cluster users into groups. We will filter out movies with insufficient information, which means movies with a small HELF value, and we set the HELF threshold to 0.65. Then we will use the filtered movies as the feature of the user vector to calculate the distance between users for subsequent clustering. We need to decide the appropriate number of groups according to the decision diagram.

The second part needs to determine the architecture of the Autoencoder, and the dropout ratio of representative items and non-representative items. After experiments, we set the architecture of the Autoencoder model to [N, 1024, N], where N represents the dimensions of the input user vector, and 1024 represents the user vector compressed into 1024 dimensions through a fully-connected neural network. And we set the dropout ratio of representative and non-representative items to 0.4 and 0.8 respectively. The following are the experimental results of the Autoencoder model architecture. It can be found that the best performance is when the number of units in the hidden layer of Autoencoder is 1024.

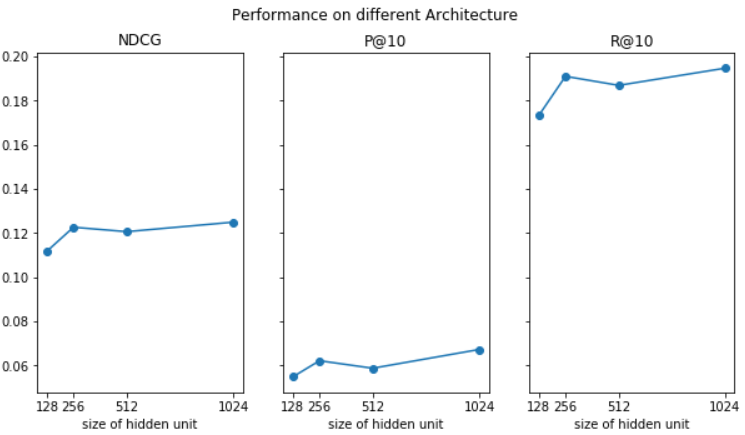


Figure.

The third part of the hyper-parameters is to determine the optimization algorithm, the activation function, and the initial method of model parameters used when training the model. After experiments, we found that using the SGD optimization algorithm with the initial method of lecun normal parameters and the selu activation function can get the best performance.

* Baseline models

In order to check whether our proposed “rejuvenation” mechanism is effective, we compare the performance of our proposed model to two baseline models.

* + - 1. Autoencoder: Autoencoder model with the neural network architecture [N, 1024, N].
      2. Random noise + Autoencoder: Autoencoder model with the neural network architecture [N, 1024, N] and randomly dropout some ratings during training.
* How cluster number affect model performance

When clustering users in MovieLens 1M, we need to select the appropriate group number according to the decision diagram. The following diagram shows the decision diagram, and each point on the diagram represents a user, and each ellipse represents the group center corresponding to the number of groups when users are divided into 1, 4 and 7 clusters. In the subsequent clustering experiment, the reason why we chose to divide all users into 1, 4 and 7 clusters is that the users represent cluster center meet the characteristics in the DPC algorithm: the local density of the cluster center should be large enough, and the relative distance between the cluster center and the cluster center should be large enough.

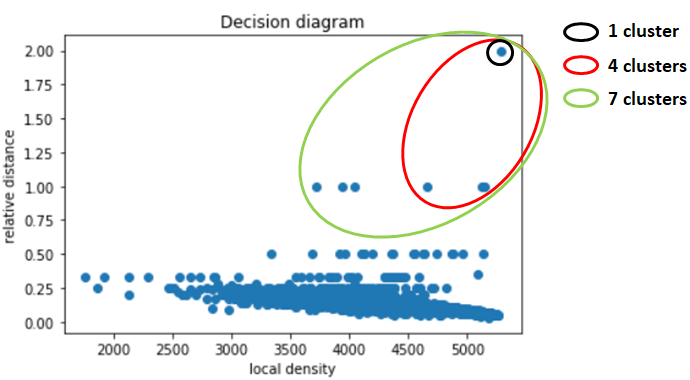


Figure. Decision diagram of DPC in our experiments

We compare the differences between our model and other benchmark models in the three performance metrics under different cluster numbers. And we can find that as the number of clusters increase, the performance of our method gradually increase.

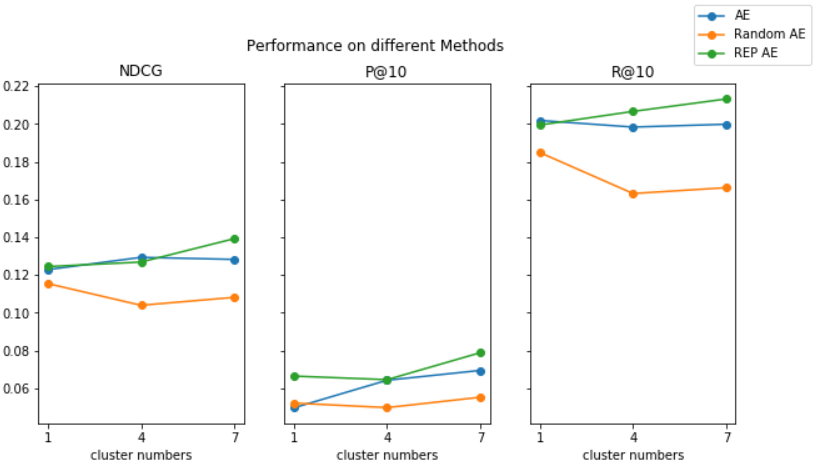


Figure.

* 1. Performance comparison of all models

In this section, we will compare our method with the benchmark models mentioned in the previous section and other current models [17][19] on the test set, and set the significance level to 0.05. Then, each method is sampled 40 times on three evaluation metrics, and their t-test statistics are calculated to test whether our method has significant greater than other methods. Below we will first introduce the models [17][19], and then organize the performance of each method in the table.

* CDAE [17]: As introduced in the literature review, CDAE receives the user’s preference value for each item (if the value is 1: the user likes the item; if the value is 0: the user doesn’t like the item or the user hasn’t seen the item) and the user ID as model input. Here we change the user’s preference value for each item to the user’s rating for the item, and randomly set some dimensions of input vector to 0 with a probability of 0.4. The model architecture of CDAE is a three-layers neural network with [N, 1024, N].
* Dual Autoencoder [19]:

As introduced in the literature review, Dual Autoencoder will train the vector representation of the user and the item at the same time. Here we set the compressed dimension of the user and the item vector to 1024, and use the compressed inner product of the user vector and item vector as the prediction result of the model.

|  |  |  |  |
| --- | --- | --- | --- |
| Models | NDCG@10 | P@10 | R@10 |
| Autoencoder |  | 0.0573 |  |
| Random Autoencoder |  |  |  |
| CDAE |  |  |  |
| Dual Autoencoder |  |  |  |
| Our Method | 0.1321 | 0.0689 | 0.2029 |

Table. Comparison table of all model performance

From the above table, we can see that the performance of our proposed method is statistically significant better than other methods. The possible reason is that the amount of information contained in the representative items we selected is helpful to restore users in the cold-start state to general user state, and this phenomenon will increase when the number of cluster increase, because the effect of clustering will group users with similar characteristics into the same group. For user in the same group, we can more precisely select representative items that represent this group of users. As for the performance of the CDAE model is not as good as our model, the possible reason is that the CDAE model randomly dropout some items in the user vector to generate noise, which makes subsequent Autoencoder unable to train as effectively as our method. The reason for the worst performance of the Dual Autoencoder model may be that although the architecture can train both the low dimension vector of the user and the item simultaneously. However, the simple inner product of user vector and item vector may not be able to capture the complex nonlinear relationship between the user and the item.

1. Conclusion

This paper mainly focuses on solving the cold start user problem in recommendation systems. We have proposed a mechanism called “user rejuvenation”. The mechanism first selects representative items for different user groups, and randomly dropout most of the ratings of non-representative items in the user vector, and randomly dropout a small part of the ratings of representative items in the user vector, so as to simulate the cold start state of the user. After that we use the cold start users restored by the “user rejuvenation” mechanism to train a deep learning model Denoising Autoencoder for each group of users. When the model training is completed, the model will have the ability to restore the input cold start user to the user with rich rating information and thus we can make recommendations.

In summary, our paper has two major contributions. First, we are the first to propose noise generation and combine the concept of Denoising Autoencoder to solve the cold start problem in the recommendation system. Second, the “user rejuvenation” mechanism we proposed can effectively help Denoising Autoencoder to train.

In the future, in addition to selecting representative items for each group of users based on the amount of item information, we may also consider the time sequence in which the items are rated to further simulate cold start user state. As well as we hope that instead of using Denoising Autoencoder with only one hidden layer, we can continue to improve the model architecture, and try to make the model architecture more complex to increase the ability of model learning. At the end, we should also conduct experiments on other different datasets to prove the generality of our method.