arXiv:2111.02740v1 [cs.IR] 4 Nov 2021

Sequential Movie Genre Prediction using Average Transition Probability with Clustering

Jihyeon Kim, Jinkyung Kim, Jaeyoung Choi*

School of Computing, Gachon University, 1342 Seongnamdaero, Sujeong-gu, Seongnam-si, Gyeongi-do 13120

Abstract

In recent movie recommendations, predicting the user's sequential behavior and suggesting the next movie to watch is one of the most important issues. However, capturing such sequential behavior is not easy because each user's short-term or long-term behavior must be taken into account. For this reason, many research results show that the performance of recommending a specific movie is not very high in a sequential recommendation. In this paper, we propose a cluster-based method for classifying users with similar movie purchase patterns and a movie genre prediction algorithm rather than the movie itself considering their short-term and long-term behaviors. The movie genre prediction does not recommend a specific movie, but it predicts the genre for the next movie to watch in consideration of each user's preference for the movie genre based on the genre included in the movie. Through this, it is possible to provide appropriate guidelines for recommending movies including the genre to users who tend to prefer a specific genre. In particular, in this paper, users with similar genre preferences are organized into clusters to recommend genres, and in clusters that do not have relatively specific tendencies, genre prediction is performed by appropriately trimming genres that are not necessary for recommendation in order to improve performance. We evaluate our method on well-known movie datasets, and qualitatively that it captures personalized dynamics and is able to make meaningful recommendations.

Keywords: Movie genre prediction, Sequential recommendation, Transition probability, User clustering.

1. Introduction

One of the most important parts of a recommendation system is to model the interactions between users and items, as well as the relationships amongst the

*Corresponding author, Tel.: +82-31-750-5829
Email addresses: `zizi39028@gmail.com` (Jihyeon Kim), `lxsz987@gmail.com` (Jinkyung Kim), `jychoi19@gachon.ac.kr` (Jaeyoung Choi)

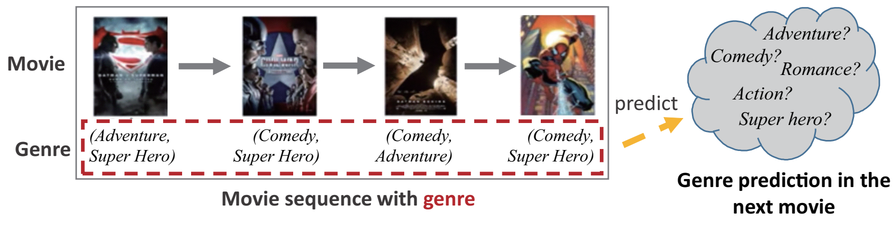


Figure 1: Movie Genre Prediction.

items themselves [1]. The former one is usually called item-to-user recommendation and the latter one is called item-to-item recommendation. In a sequential recommendation, user preferences and sequential patterns can be extracted by the above two kinds of interactions. However, it is not an easy task to learn the personalized sequential behavior from collaborative data since long and short term dynamics of the users have to be carefully considered for both personalization and sequential transitions. Especially, if the data is sparse, estimating parameters of learning models becomes very hard. For this reason, a main object is to design personalized models of user behavior using the users' recent purchase histories for the sequential recommendation systems [2]. To reflect the short-term dynamics of the user's behavior, Markov Chain (MC) approaches have been used, which assume that the next action depends on only the previous action. Since the last related item is in general the key factor affecting the user's next action, first-order MC based methods show strong performance, especially on sparse datasets [3]. For the long-term dynamics of user's behavior, Recurrent Neural Networks (RNNs) have been used to reflect all previous actions with a hidden state, which is used to predict the next action [4]. Both approaches, while strong in specific cases, are quite limited to certain types of data. MC-based approaches work well in high sparsity settings, but may be difficult to obtain the intricate dynamics of long-term complex scenarios. Different from this, RNNs work well in such long-term scenarios, which require large amounts of data. Recently, session-based recommendation systems (SBRSs) have been investigated [5] as a new approach to recommendation systems. Different from other Collaborative Filtering (CF)-based recommendation systems, which usually model long-term yet static user preferences, SBRSs try to capture short-term but dynamic user preference for more sensitive accurate recommendations to the evolution of their session contexts. Different from purchasing one specific item, the authors in [6] considered the sequential recommendation scenario that users purchase some similar items simultaneously. In this scenario, proposing each user a personalized "list of items" is the main object of the recommendation. Similar to this approach, one can consider an attribute prediction for movie data to the users. For instance, the genre/category of a movie can be an important attribute for user/item similarity in the recommendation systems. Moreover, this information is provided when new content is created. Based on this, the authors in [7] considered a movie recommendation system based on genre correlations, to improve the existing genre correlation algorithm, and compare the obtained results using the previous algorithm and the proposed algorithm. In [8], the authors introduced a recommender system using movie genre similarity and preferred genres. However, in these works, only a static situation was considered, not a sequentially changing recommendation systems.

In this paper, we first consider the prediction of movie genres included in preferred movies before recommending movies. The genre is one of the important features of a movie, which gives guidelines on which movies each user prefers. In the sequential movie recommendation system, we extract the genre included in the movie each user watched and studied the genre preference, and conducted a study on what movie genre the user will see next.

Our main contributions are described as follows:

- First, different from most prior researches of movie recommendation systems, we focus on the genres, which are included in a movie rather than the movie itself as a sequential prediction item. Although it cannot be used directly in sequential movie recommendation systems, it can show how well genre-based prediction works in learning user preferences.

- Second, for predicting the long-term user's preference behavior of movie genres, we use RNN-based learning models that show the best performance recently. Further, we consider an Average Transition Probability (ATV) between genres as a Markov chain to reflect the short-term behavior of the user's preference as in [2]. To see the effect of the average transition probability, we consider four kinds of training data with combining genre vectors.

- Third, we propose a clustering approach based on the k-means clustering, which has similar preferences for movie genre. In order to improve the prediction performance, we also propose a method for properly trimming genres that act badly on performance using the results obtained based on the RNN-based sequential learning models presented above.

- Finally, we evaluate our method on well-known movie datasets, and qualitatively that it captures personalized dynamics and is able to make meaningful recommendations for the movie genres. The results show that clustering with trimming improves the prediction performance whereas applying ATV has a very negligible performance improvement.

The remainder of this paper is organized as follows. Section 2 discusses related studies. In Section 2, our clustering and training methods are presented. In Section 4, the experiment results of our proposed methods are presented and some limitation and future works are given in Section 5. In Section 6, we conclude the paper.

2. Related Works

In recent sequential recommendation systems, most works focus on how to predict the short-term and long-term preference dynamic of users. As a short-term dynamic, the Markov chain approach has been studied. Zimdars *et al.*[9] described a sequential recommender based on the Markov chains. They studied how to catch sequential patterns to predict the next state with a standard predictor such as a decision tree. Rendle *et al.*[6] proposed a Factorizing Personalized Markov Chain (FPMC), where they modeled based on user-specific transition with Markov Chain, about the history of a basket. FPMC propagates information among users, items, transitions which has similar favor or patterns to extract the sequential pattern. Shani *et al.*[10] considered a recommendation system based on Markov decision processes (MDP). For this, they used the Maximum Likelihood Estimates (MLE) of the MC transition graphs and they suggested several heuristic approaches such as clustering and skipping. Mobasher *et al.*[11] adopted pattern mining methods to extract sequential patterns for generating recommendations. He *et al.*[3] proposed a Translation-based Recommendation (TransRec) with sequential data. The main approach is to consider items(movies) as a translation vector. Khorasani *et al.*[12] also used the MC to recommend courses that students taken. To do this, they estimated the transition probability of the MC from the record of courses students take based on MLE and enhanced MLE with skip-gram modeling. Konen *et al.*[13] considered temporal dynamics and they showed several results by using the evolution of users and items over time-based on Netflix data.

For the long-term dynamics, most recommendation usually relies on Matrix Factorization (MF) or other similarity-based approaches. In the prior work [14] using the MF, the authors considered the recommendation problem as a problem that infers missing values of a partially observed user–item matrix. Srebro *et al.*[15] proposed the maximum margin MF, which used low-norm instead of low-rank factorizations. Salakhutdinov *et al.*[16] considered a probabilistic MF (PMF) model that expresses the user preference matrix by multiplication of two lower-rank user and item matrices. The PMF approach was especially effective at making better predictions for sparse user rating data. He *et al.*[1] suggested an extended FPMC, called Fossil, to present the information of sequential patterns by considering high-order Markov chains and similarity models. As factorization machine-based sequential recommendation systems usually utilize the matrix or tensor factorization to factorize the observed user-item related data into latent factors of users and items for recommendations [4]. Specifically, some works [18, 19] have used the estimated latent representations as an input of a network to further calculate an interaction score between users and items, or successive users' actions.

Recently, deep learning technologies have been introduced in the sequential recommendation problem, such as RNNs [4, 20, 21], Long-Short Term Memory (LSTM) [22, 23], and Gated Recurrent Unit (GRU) [24]. These deep learning-based recommender systems have particularly shown a high performance for the sequential recommendation. In [4], the authors suggested new ranking loss

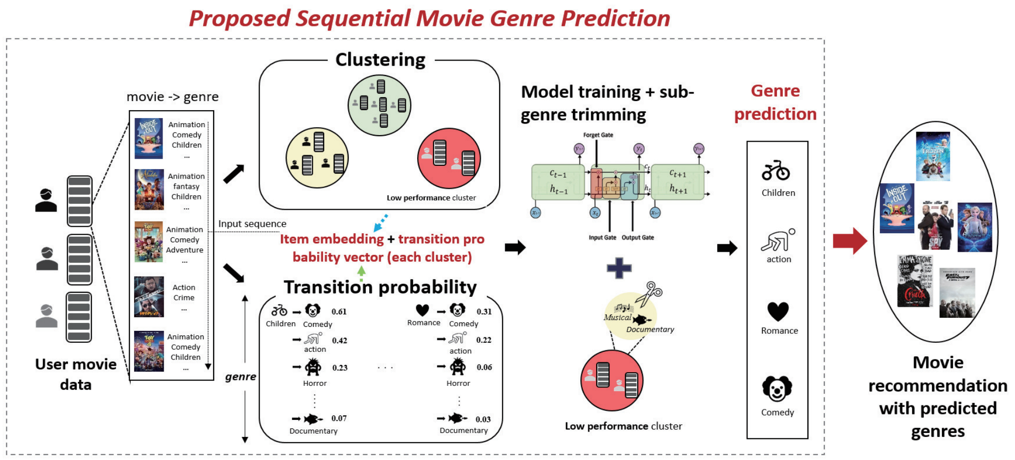

Proposed Sequential Movie Genre Prediction
movie -> genre
User movie data
Clustering
Low performance cluster
Input sequence
Item embedding + transition probability vector (each cluster)
Transition probability
genre
Model training + sub-genre trimming
Low performance cluster
Genre prediction
Children
action
Romance
Comedy
Movie recommendation with predicted genres

Figure 2: Overall view of our proposed sequential movie genre prediction.

functions corresponding to RNNs in the recommendation model. In [20] the authors designed a novel recommendation model named Recurrent Collaborative Filtering (RCF), which combines RNN and CF properly. In [21] the authors introduced an algorithm named Recurrent Translation-based Network (RTN). Their model reflected both short-term and long-term of a user's preference. In [22] the authors considered LSTM to extract the dependencies of both users and movies. Unlike the prior recommendation models, they considered a method of updating the state with recent operations as input. In [23] the authors introduced an LSIC model, Leveraging long and Short-term Information in Content-aware movie recommendation via adversarial training, which combined global behaviors from MF into the RNN for the top-N movies.

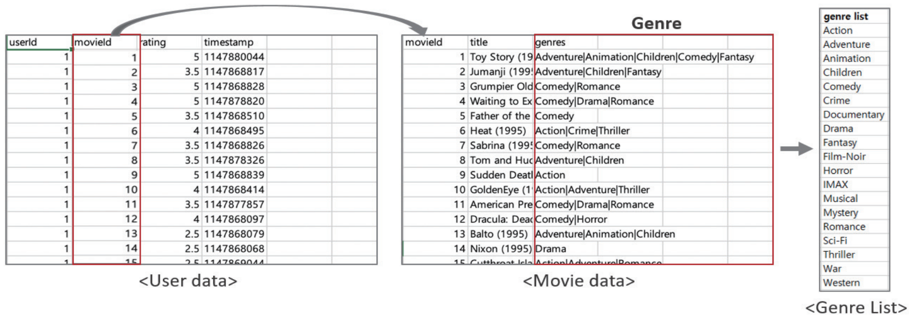
In [24] the authors considered a GRU-based RNN for session-based recommendations. Yuan *et al.*[25] suggested a Convolutional Neural Network (CNN) method that gives a sequence of user-item interactions. In their model, a CNN first puts user-item interactions data into a matrix, regarding such a matrix as an "image" in the time and latent spaces. Wu *et al.*[26] proposed a GNN to capture the sequential behavior of complex transitions over user-item interactions. Zhang *et al.*[27] adopted a self-attention mechanism to extract the item-item interaction from the user's historical interactions. Sachdeva *et al.*[28] considered a variational autoencoder to model a user's preference based on her historical sequential data, and combines latent variables with temporal dependencies for preference modeling. Similar to our works, Choi *et al.*[7] designed a movie recommendation algorithm based on genre correlations. For this, they assumed that movie genres are defined by some experts such as directors or producers to guarantee reliability. Then, they computed genre correlations and used them in a movie recommendation system. In [8], the authors also consider a movie genre similarity to provide related services in a mobile experimental environment.

3. Genre prediction Algorithm

In this section, we will propose a movie genre prediction algorithm. To do this, we first classify the genres included in the movie data watched by each user as shown in Figure 2. Next, we cluster the users into similar groups based on the ratings of the movies. Then, we estimate an average transition probability from genre to the genre for each cluster. Using this, we train some deep learning models. Since some sparse data of genres may cause poor performance in predicting the genres, we appropriate trim these after model training, and we finally predict a preferred genre for the group closest to the user. Based on the predicted genre, some suitable movies that contain that genre can be recommended. We describe all the above steps in detail as the following subsections.

3.1. Data Preprocessing

As a sequential movie genre prediction, we consider movie data by user ID and timestamp to extract each user's movie sequence in chronological order (left part of Figure 3). We drop user data with five or fewer movie viewing sequences and import user data with five or more movie viewing sequences in the pre-processing. At this time, the five most recent movies generated per user are arranged in chronological order. Next, information on genres included in movies that the user has watched is organized (middle part of Figure 3). One can see that a single movie contains several genres at the same time. We extract all kinds of genres that each film contains and set the data sequence to n-dimensions ($n > 0$) of one-hot vector, meaning each of the n genres (right part of Figure 3). We denote $\mathcal{G}$ the set of genres in the paper.



userId	movieId	rating	timestamp
1	1	5	1147880044
1	2	3.5	1147868817
1	3	5	1147868828
1	4	5	1147878820
1	5	3.5	1147868510
1	6	4	1147868495
1	7	3.5	1147868826
1	8	3.5	1147878326
1	9	5	1147868839
1	10	4	1147868414
1	11	3.5	1147877857
1	12	4	1147868097
1	13	2.5	1147868079
1	14	2.5	1147868068

movieId	title	genres
1	Toy Story (19	Adventure|Animation|Children|Comedy|Fantasy
2	Jumanji (199	Adventure|Children|Fantasy
3	Grumpier Old	Comedy|Romance
4	Waiting to Ex	Comedy|Drama|Romance
5	Father of the	Comedy
6	Heat (1995)	Action|Crime|Thriller
7	Sabrina (199	Comedy|Romance
8	Tom and Huc	Adventure|Children
9	Sudden Deatl	Action
10	GoldenEye (1	Action|Adventure|Thriller
11	American Pre	Comedy|Drama|Romance
12	Dracula: Deac	Comedy|Horror
13	Balto (1995)	Adventure|Animation|Children
14	Nixon (1995)	Drama

genre list
Action
Adventure
Animation
Children
Comedy
Crime
Documentary
Drama
Fantasy
Film-Noir
Horror
IMAX
Musical
Mystery
Romance
Sci-Fi
Thriller
War
Western

Figure 3: Example of data preprocessing for movie genre prediction ($n = 19$).

3.2. Clustering

To reflect user similarity, we consider a clustering approach. For this, we consider that each user scores a range between 0.5 and 5 rating for the most recent five movies they watched. Based on the rating sequences of each user, we obtain the average rating of each genre as shown in Figure 4. Let $\mathcal{U}$ be the set of users and we consider an average rating data is generated per one user so we have $|\mathcal{U}|$ by n rating matrix. Using this matrix, we apply a k-means clustering to obtain clusters. We let $\mathcal{C} := \{C_1, ..., C_k\}$ be a set of clusters C_l for $1 \leq l \leq k$ after performing the clustering.

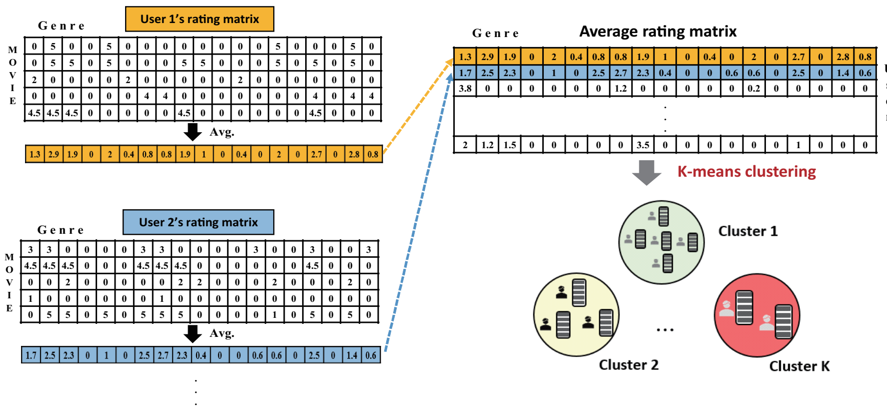


Figure 4: Illustration of example for clustering.

3.3. *Average Transition Probability of genres*

3.3.1. *Markov Chain for Set of Genres.*

In our genre prediction system, we use transition probabilities from genre to genre. It is known that many approaches for the sequential recommendation, the MC is used to reflect the short-term sequential behavior of a user. The MC assumes the next choice of item depends only on the current choice. Formally, it is described as follows. The transition probability matrix is generated for each cluster. To do this, we first consider the sequence of selected movies for each user in a cluster. However, as described before, a movie may contain multiple genres such as romance, action and comedy, simultaneously. We consider all genres included in the current and next movies and count them in the n by n matrix. We consider the transition matrix for each cluster, separately. Then, for all $C_l \in \mathcal{C}$, we summarize them for all user's chosen movies and normalize them to obtain the transition probability from genre to genre as shown in Figure 5.

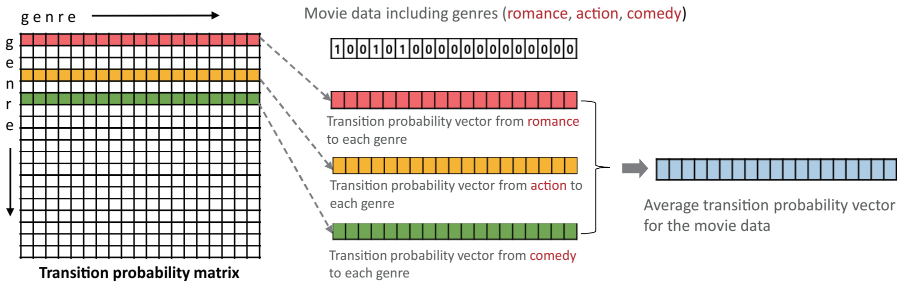

genre
genre
Transition probability matrix
Movie data including genres (romance, action, comedy)
1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
Transition probability vector from romance to each genre
Transition probability vector from action to each genre
Transition probability vector from comedy to each genre
Average transition probability vector for the movie data

Figure 5: Transition probability matrix and average transition probability vector.

To describe this, we let M_t^l and M_{t-1}^l be the selected movie sets for all user $u \in C_l$ at time t and $t-1$, respectively. Then, the transition probability of the first-order Markov chain for the movie selection for the cluster l is given by:

$$p(M_t^l | M_{t-1}^l). \tag{1}$$

However, in the genre prediction, we focus on the transition from genre (included in a movie) to genre. For this, we let $\mathcal{G}_t \subset \mathcal{G}$ be the set of genres which are contained in the movie M_t^l for all user $u \in C_l$ at time t. Consider two genres $i, j \in \mathcal{G}$, we model the genre transition probability in the cluster C_l as:

$$p_{ij} := p(j \in \mathcal{G}_t^l | i \in \mathcal{G}_{t-1}^l). \tag{2}$$

3.3.2. Estimation of Transition Probabilities.

To make predictions using the transition probability in (2), it needs to be estimated. To do this, we consider the following ratio:

$$\begin{align} \hat{p}_{ij} := \hat{p}(j \in \mathcal{G}_t^l | i \in \mathcal{G}_{t-1}^l) &= \frac{\hat{p}(j \in \mathcal{G}_t^l \wedge i \in \mathcal{G}_{t-1}^l)}{\hat{p}(i \in \mathcal{G}_{t-1}^l)} \tag{3} \\ &= \frac{|\{(\mathcal{G}_t^l, \mathcal{G}_{t-1}^l) : j \in \mathcal{G}_t^l \wedge i \in \mathcal{G}_{t-1}^l\}|}{|\{(\mathcal{G}_t^l, \mathcal{G}_{t-1}^l) : i \in \mathcal{G}_{t-1}^l\}|}, \tag{4} \end{align}$$

where the value of the denominator in (4) is the number of genre i at time $t-1$, and the numerator means the number of genre i at time $t-1$ and genre j at the next time point t. Hence, the estimated transition probability indicates that the ratio that the number of genre j, which is selected at time t among number of genres i at time $t-1$. However, since the user does not select a specific genre, but sequentially selects a movie including several genres, the transition probability between genres cannot be used by itself. Hence, using the movie data including several genres, we count the number of transitions to each genre. For example, if a movie contains three genres (romance, action and comedy) as in Figure 5, count all genres which are included in the next selected movie. Then we compute the ratio in (4) so that we have the estimated transition probabilities from romance to each genre, from action to each genre and from comedy to each genre, respectively. Next, we take an average for these tree transition probabilities and call it an Average Transition probability Vector (ATV) for each selected movie. The reason why we use the ATV is that there is no information about transition from a specific genre to another genre in actual data, only information about transition from a movie including these genres to another movie is given. Formally, the ATV can be presented by:

$$\hat{p}_j^{ATV} := p(j \in \mathcal{G}_t^l | \mathcal{G}_{t-1}^l) = \frac{1}{|\mathcal{G}_{t-1}^l|} \sum_{i \in \mathcal{G}_{t-1}^l} p(j \in \mathcal{G}_t^l | i \in \mathcal{G}_{t-1}^l), \tag{5}$$

for all $j \in \mathcal{G}$. We will use this ATV for training with each user's selected movie sequence.

3.4. Model Training

3.4.1. Training Data Types

As training data, we consider the following four types of training data during the model training: (1) Sum of transition vector and movie genre embedding, (2) Multiplication of transition vector and movie genre embedding (3) Successive transition vector and movie genre embedding, and (4) Movie genre only.

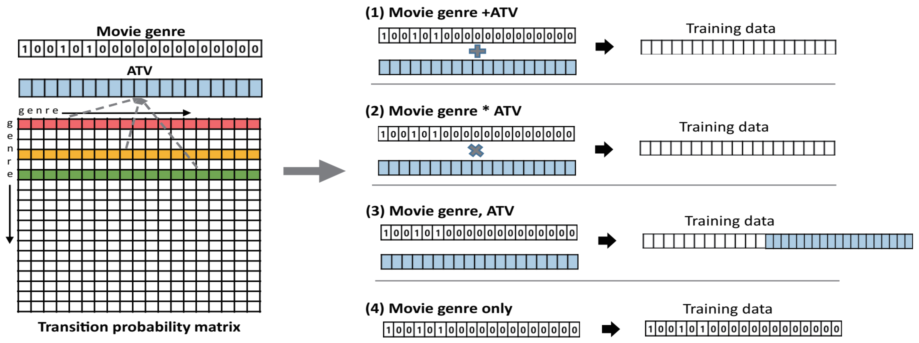


Figure 6: Four types of training data

First, the sum of ATV and movie genre embedding data is nothing but performing the summation of movie genre vector and ATV as shown in Figure 6. Second, the multiplication of ATV and movie genre embedding is the data after multiplying these two vectors by component-wise, which results in a new vector. Third, the successive ATV and movie genre embedding is the data that attaches the ATV at the end of the movie genre for the model training. Finally, the movie genre only is the data that consider only the movie genre vector. The reason we consider the training data types in this way is to check how much the ATV considered for short-term dynamics helps to improve the model performance. We will show the results for these four training data types in the experiment later.

3.4.2. Training Models

In our approach, we use RNN-based models to capture the long-term dynamics of sequential movie genre data such as RNN, LSTM and GRU. We will describe these methods in detail as follows.

(1) RNN. First, RNN is one of the deep learning models designed to be useful for sequential data processing. RNN is a recursive model that performs the same function on all input data and the output for the current input depends on past calculations. When the output data is generated, it is copied and sent back into the recurrent network. Based on the current input and the output that it has generated from the previous input, the RNN learns some sequential data and makes a decision.

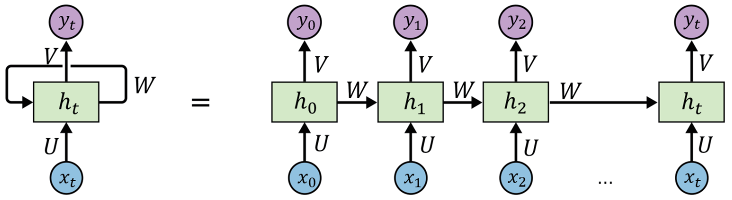


Figure 7: Recurrent Neural Network.

To formally describe, we let x_t be the input vector and y_t be the output vector at time t as shown in Figure 7. Then, a state value of hidden layer h_t at time t is given by:

$$h_t = \tanh(Ux_t + Wh_{t-1} + b), \tag{6}$$

where U, W are model parameter matrix and b is a constant vector. As a function of h_t, we consider a hypublic tangential function $tanh(\cdot)$. The output vector y_t is given by:

$$y_t = f(Vh_t + b), \tag{7}$$

where V is a model parameter and f is an activation function. RNN is optimized to approximate the function by capturing sequential patterns. However, if the length of the sequence input to the RNN is long, the effect of the elements at the beginning of the sequence will gradually loosing as the time step progresses and disappear after a certain period of time. This is because the constant value is multiplied equally in each cycle. This is called a long-term dependency problem that the RNN is useful for a short sequence of data.

(2) LSTM. To overcome the main disadvantage of RNN, one of the improved methods, LSTM has been introduced. LSTM is a kind of RNN that is capable of selectively remembering sequences for a long period of time. The main difference from the RNN is that LSTM introduces a "cell state" for each time t, which allows information to flow unaltered. In LSTM, the cell state is regarded as a long-term memory since the previous information is stored in it as a recursive nature of the cells. The forget gate is used to update the cell states. The forget gate outputs values saying which information to forget by multiplying 0 to a position in the matrix. If the output of the forget gate is 1, the information is kept in the cell. The input gates determine which information should enter the cell states. Finally, the output gate tells which information should be passed on to the next hidden state. Based on this fact, the LSTM addresses the long-term dependency problem of RNN. In general, the LSTM consists of the following four parts as shown in Figure 8:

(*i*) ***Forget Gate Layer.*** As a first part, the forget gate layer decides to filter some information from the cell state by using a sigmoid function. It obtains information at h_{t-1} and x_t, and outputs a number between 0 and 1 for each number in the cell state c_{t-1}. The number 1 implies "completely keep this" while 0 represents "completely drop this." The output of the forget gate vector t_t is given by:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \tag{8}$$

where σ is a sigmoid function and W_f and b_f are weight matrix and bias vector parameter.

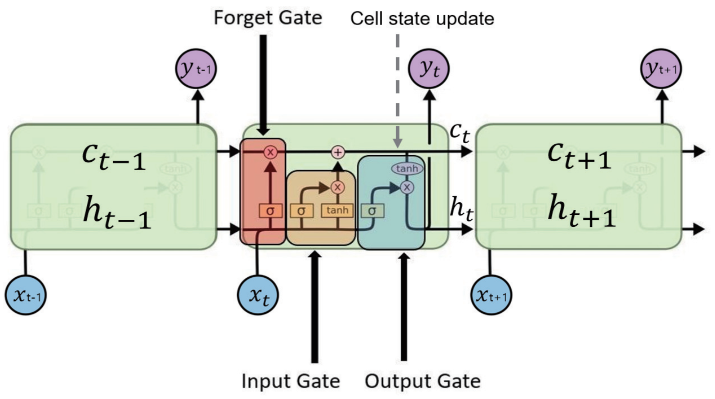


Figure 8: Long Short Term Memory [30].

(*ii*) ***Input Gate Layer.*** In the next step, LSTM decides whether new information to store or not in the cell state. For this, an "input gate layer" decides which values we'll update as a sigmoid gate. Next, a tanh gate generates a vector of new values, $\tilde{c}_t$, that could be added to the state. Then, these two layers are combined to create an update to the state. The input gate vector i_t is given by:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \tag{9}$$

where W_i and b_i are weight matrix and bias vector parameter. The cell input activation vector $\tilde{c}_t$ is computed by:

$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c), \tag{10}$$

where W_c and b_c are weight matrix and bias vector parameter and $\tanh(\cdot)$ is a hyperbolic tangential function as a sigmiod function.

(iii) ***Cell State Update.*** Next, LSTM performs a cell state update procedure to update the old cell state, c_{t-1}, into the new cell state c_t. The previous steps already decided what to do, it just needs to actually do it. Then, it multiplies the old state to f_t, forgetting the things it decided to forget earlier. Then it adds $i_t \cdot \tilde{c}_t$. This is the new candidate value, scaled by how much it decided to update each state value. The update of cell state $_t$ is computed by:

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t. \tag{11}$$

(iv) ***Output Gate Layer.*** Finally, in the output gate layer, LSTM decides what information going to be output. This output will be based on the cell state, but will be a filtered version. First, it runs a sigmoid layer which decides what parts of the cell state going to output. Then, it puts the cell state through tanh and multiplies it by the output of the sigmoid gate, so that it only output the parts it decided to. The output gate vector o_t is given by:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{12}$$

where W_o and b_o are weight matrix and bias vector parameter of the output gate layer. Here, h_t is computed by:

$$h_t = o_t * \tanh(c_t). \tag{13}$$

(3) GRU. Cho *et al.*[29] first introduced a slight variation on the LSTM, named GRU. It uses the forget and input gates as a single update gate. Further, it also combines the cell state and hidden state. It is known that the GRU is simpler than LSTM model.

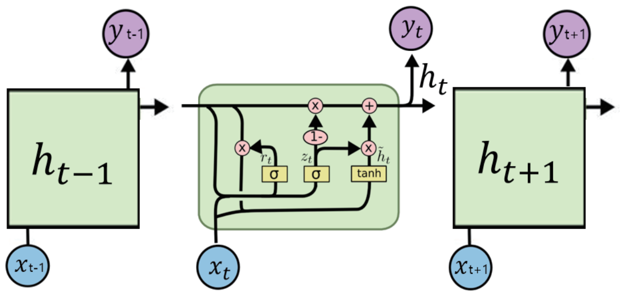


Figure 9: GRU [31].

The detailed structure of GRU as shown in Figure 9 is in what follows:

(i) ***Update Gate.*** In GRU, it first begins with computing the update gate z_t for time step t by:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]), \tag{14}$$

where W_z is a weight matrix. When the input x_t is generated into the network, it is multiplied by its own weight W_z. The previous h_{t-1} also multiplied by the current input x_t. As an activation function, a sigmoid is commonly used. The update gate is used to determine how much of the past information (from previous time steps) needs to be passed along to the next. The most useful fact is that the model can control to copy all the information from the past and eliminate the risk of vanishing gradient problems.

(ii) ***Reset Gate.*** Next, a reset gate is applied from the model to decide how much of the past information to forget by:

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]), \tag{15}$$

The difference between this from the update gate is the weights and the gate's usage. As similar steps in the update gate, it plugs in h_{t-1} and x_t, multiply them with their corresponding weights, sum the results and apply the sigmoid function.

(iii) ***Current memory content.*** The current memory content is then used for the reset gate to store the relevant information from the past. It is computed as:

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t]), \tag{16}$$

where W is a weight matrix and the operator $*$ denotes the Hadamard element-wise product. Then the result determines what to remove from the previous time steps. In this step, it uses a tanh as the nonlinear activation function.

(iv) ***Final memory at current time step.*** As the last step, the network needs to calculate h_t, which is a vector that holds information for the current unit and passes it down to the network. In order to do that the update gate is needed. It determines what to collect from the current memory content $\tilde{h}_t$ and what from the previous steps h_{t-1} by weighting the update gate value:

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t. \tag{17}$$

3.4.3. Sub-genre trimming

Finally, using the evaluation results of the trained models, we perform a sub-genre trimming process based on a pre-defined threshold of the evaluation metric scores for each cluster. To do this, we first select clusters that do not satisfy the criteria of evaluation metrics.

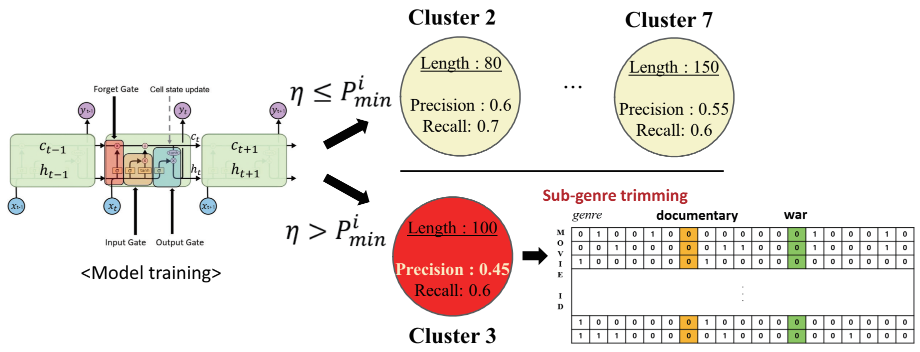


Figure 10: Sub-genre trimming. In this example, we set η=0.5 and there are two kinds of performance metrics such as precision and recall. We see that the minimum value of evaluation metrics $P^2_{min} = 0.6$ and $P^7_{min} = 0.55$ for the cluster 2 and cluster 7, respectively. Hence, these clusters are not regarded as the trimming clusters, However, we have $P^3_{min} = 0.45 < 0.5$ for cluster 3 so it is regarded as a trimming cluster.

```
Algorithm 1 Sub-genre Trimming
Input: Set of movie-genre matrices $\mathcal{M} := \{\mathcal{M}_i\}_{i=1}^k$ for each cluster $C_i$ with each evaluated value $P^i_e$, threshold parameters $\eta$ and $\theta$.
Output: Sub-genre trimmed matrices set $\mathcal{M}' := \{\mathcal{M}'_1, ..., \mathcal{M}'_k\}$.
Set $C'_i = \emptyset$ for all $1 \leq i \leq k$;
for $1 \leq i \leq k$ do
    Set $|C_i|$ the length (number of movies) of a cluster $C_i$ for each $i$ and set;
        $$P^i_{min} := \min_{e \in \mathcal{E}} P^i_e. \qquad (18)$$
    if $P_{min} < \eta$ then
        for $1 \leq j \leq n$ do
            Count the number of genre $j$ in a column of movie-genre matrix $\mathcal{M}_i$ and set it by $g_j = \sum_{u \in C_i} c_{uj}$. If $g_j < 100 \times \theta_i$, replace all values of the column $j$ by zero;
        end for
    end if
    $\mathcal{M}'_i \leftarrow \mathcal{M}_i$;
end for
Return $\mathcal{M}' := \{\mathcal{M}'_1, ..., \mathcal{M}'_k\}$;
```

More precisely, we let P^i_e be the value of evaluation for a performance metric $e \in \mathcal{E}$ of the cluster i, where $\mathcal{E}$ is a set of performance metrics such as

$\mathcal{E} = \{Precision, Recall, Accuracy\}$. Next, we let $P^i_{min} := \min_{e\in\mathcal{E}} P^i_e$ be the minimum evaluated value of P_e for all $e \in \mathcal{E}$. Then, we check the value P^i_{min} for each cluster and if there exists an evaluation value less than a pre-defined threshold $\eta > 0$, $i.e. P^i_{min} < \eta$, then we choose the cluster as the target cluster for the sub-genre trimming. For example, if we consider the evaluation metrics as precision and recall and the threshold η is given by 0.5, the cluster 3 does not satisfies this as shown in Figure 10. Hence, it is regarded as a target cluster for the sub-genre trimming. After selecting target clusters, we find sub-genres that are less than θ_i percent of the total length (number of movies) of each target cluster i to trim. To do this, we let $\mathcal{M}_i = [c_{uj}]_{u\in C_i, 0\le j\le n}$ be the movie-genre matrix for the cluster i and let $\mathcal{M} := \{\mathcal{M}_i\}^k_{i=1}$. Then, using the matrix, we find the genres that the number of total sum is less than $100 \times \theta_i$, $i.e. \sum_{u\in C_i} c_{uj} < 100 \times \theta_i$. To minimize the data loss, we replace the values as zero rather than deleting thm. The reason for performing this is that it will increase the accuracy of evaluation of clusters that do not have explicit preference. In the example of Figure 12, the length of the cluster is 100 and $\theta = 0.1$, then we have $100 \times 0.1 = 10$. We choose genres that do not have more than 10 data in the cluster such as documentary and war. After these procedures, we finally obtain the sub-genre trimmed matrices set $\mathcal{M}' := \{\mathcal{M}'_1, ..., \mathcal{M}'_k\}$. We will use this matrices to obtain the performance results.

4. Experiment Results

In this section, we will show our experiment results. For this, we first use a well-known movie dataset and performance metrics of the evaluation as follows.

4.1. Data

In this section, we present our experimential results. For the simulation, we use a movielens data set(ml-25m), where 25 million ratings and one million tag applications applied to 62,000 movies by 162,000 users [32]. For the sequential recommendation, we sort the data by 'userId' and 'timestamp' (to extract each user's movie sequence in chronological order) as shown in Figure 3. We drop user data with 5 or fewer movie viewing sequences and import user data with 5 or more movie viewing sequences to configure the dataset for the experiment. At this time, five movie data generated per user are arranged in chronological order. To train the model, we convert the data sequence to 19 dimensions of a one-hot vector, meaning each of the 19 genres: $\{Action, Adventure, Animation, Children, Comedy, Crime, Documentary, Drama, Fantasy, Film{-}Noir, Horror, IMAX, Musical, Mystery, Romance, Sci{-}Fi, Thriller, War, Western\}$.

4.2. Performance Metrics

As performance metrics, we consider (i) Recall, (ii) Precision, (iii) Accuracy and (iv) F1-score. To formally explain these metrics, we denote True Positive (TP) as the number of correctly predicted positive values which is the actual value is yes and the predicted value is also yes. True Negatives (TN) indicates the number of correctly predicted negative values which is the actual value is no and the predicted value is also no. False Positives (FP) is the number of actual value is no and the predicted value is yes. False Negatives (FN) is the number of the actual value is yes but the predicted value is no. Then, the three metrics are described as follow:

(i) ***Precision:*** A Precision is the ratio of correctly predicted positive answers to the total predicted positive answers.

$$Precision := \frac{TP}{TP + FP}. \tag{19}$$

(ii) ***Recall:*** A Recall is the ratio of correctly predicted positive answers to all answers in the actual class of answers.

$$Recall := \frac{TP}{TP + FN}. \tag{20}$$

(iii) ***Accuracy:*** An Accuracy is a ratio of correctly predicted answers to the total answers.

$$Accuracy := \frac{TP + TN}{TP + FP + FN + TN}. \tag{21}$$

The accuracy is one of good measures when the values of false positive and false negatives of the datasets are almost same.

(iv) ***F1-Score:*** This metric is a weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account.

$$F1\text{-}score := \frac{2(Recall * Precision)}{Recall + Precision}. \tag{22}$$

F1-score is usually more useful than accuracy when the values of false positive and false negatives of the datasets are quite different.

Using the previously described data and performance metrics, we obtain various experimential results of the movie genre prediction in the following subsection.

4.3. Results

In the result, we obtain that how much the prediction performances are affected for (1) Clustering, (2) Sub-genre trimming, and (3) ATV. To see the clustering effect, we obtain the results as before and after clustering. We consider seven clusters after applying the kNN during the clustering step and the results show the mean performance of all clusters and best and worst performance of clusters among them, respectively. To select the trimming clusters, we set $\eta = 0.5$ and $\theta_i = 0.1$ for all cluster i.

4.3.1. Effects of Clustering

Table 1: Four performance results of RNN with the training data type [Genre*ATV]

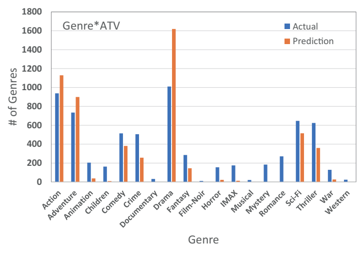
Clustering	Recall	Precision	Accuracy	F1-score
BC	0.44	0.77	0.85	0.56
AC (best)	0.73	0.80	0.91	0.76
AC (worst)	0.29	0.87	0.85	0.43
AC (mean)	0.52	0.77	0.86	0.58

Table 2: Four performance results of LSTM with the training data type [Genre*ATV]

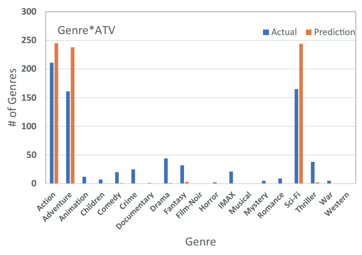
Clustering	Recall	Precision	Accuracy	F1-score
BC	0.45	0.76	0.85	0.57
AC (best)	0.75	0.79	0.91	0.77
AC (worst)	0.27	0.84	0.84	0.41
AC (mean)	0.51	0.75	0.86	0.59

Table 3: Four performance results of GRU with the training data type [Genre*ATV]

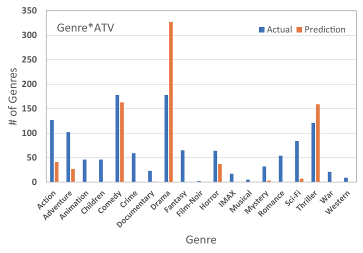
Clustering	Recall	Precision	Accuracy	F1-score
BC	0.49	0.70	0.83	0.58
AC (best)	0.73	0.75	0.89	0.74
AC (worst)	0.35	0.69	0.79	0.46
AC (mean)	0.53	0.69	0.84	0.59



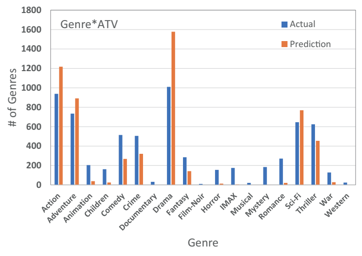
(a) (RNN) Result without clustering.



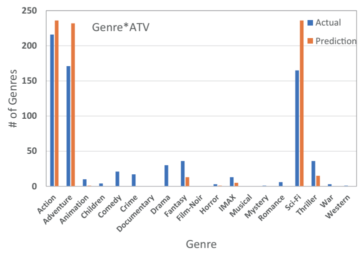
(b) (RNN) Best result among all clusters after clustering.



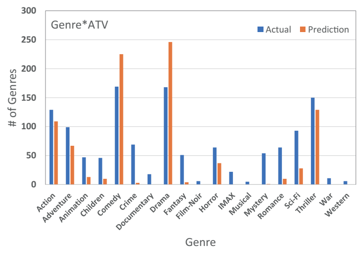
(c) (RNN) Worst result among all clusters after clustering.



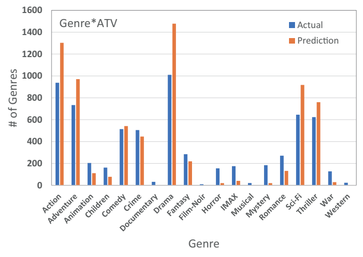
(d) (LSTM) Result without clustering.



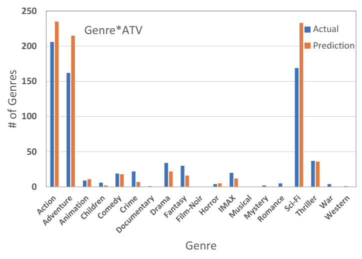
(e) (LSTM) Best result among all clusters after clustering.



(f) (LSTM) Worst result among all clusters after clustering.



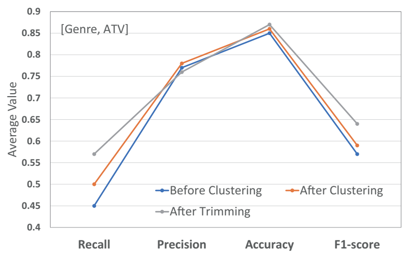
(g) (GRU) Result without clustering.



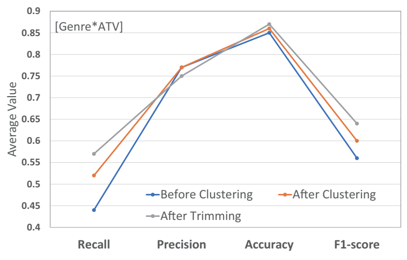
(h) (GRU) Best result among all clusters after clustering.

Figure 11: Result of Clustering for RNN ((a), (b), (c)), LSTM ((d), (e), (f)), and GRU ((g), (h), (i)), respectively.

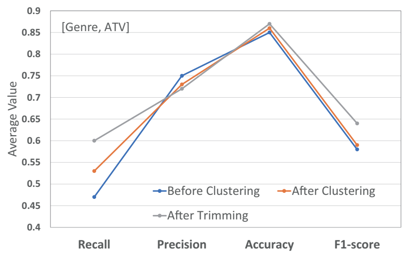
As a first result, we obtain the performances of each model before the clustering and after clustering in Figure 11. Without clustering (Figure 11(a), Figure 11(d) and Figure 11(g)), the performances are measured in consideration of all users without distinguishing users based on any criteria. This is because the data is not classified, it is difficult for the models (RNN, LSTM and GRU) to grasp the data itself, and it is difficult to extract any information. Therefore, all three models show relatively poor performance. In order to improve performance, we apply clustering for users with similar preferences. After the clustering (Figure 11(b), Figure 11(e) and Figure 11(h)), we see that the performance is quite improved, and among all stages of our experiment, the range of performance increase is large. In this case, users with similar preferences are grouped together, so that in the case of a group in which preferences are well expressed, the range of values between preferred and non-preferred genres is very large. In other words, the number of data from genres with clear preferences is overwhelmingly large. It can be said that this helped make the process of recommending movies that the group would like to be easier for the model. However, even after doing this, there were occasionally (1 or 2) clusters where the preference was not clearly evident (Figure 11(c), Figure 11(f) and Figure **??**). In Table 1-3, we obtain the results of four performance metrics (Recall, Precision, Accuracy and F1-score) with respect to three training models, respectively. In the experiment, we consider the training data as [Genre*ATV] as a representative one. Here, the abbreviations BC means before clustering and AC means after clustering. AC (best) and AC (worst) indicate the best result and worst result among clusters. Finally, AC (mean) present a mean of all result of clusters. As a result, we see that the performances of AC for four metrics are improved compared to BC except for the worst case. Further, among them, we also check that the accuracy has the highest value since false positives and false negatives of the datasets are not quite different.



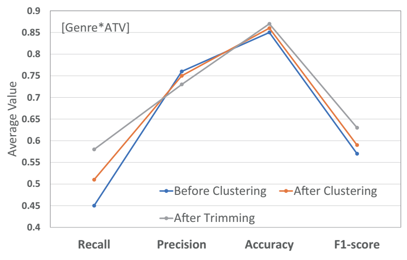
(a) (RNN) Result after trimming.



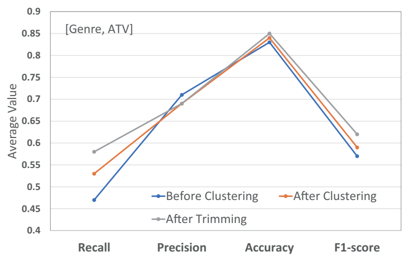
(b) (RNN) Result after trimming.



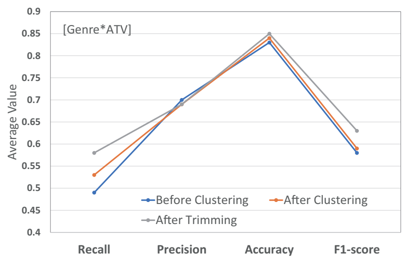
(c) (LSTM) Result after trimming.



(d) (LSTM) Result after trimming.



(e) (GRU) Result after trimming.



(f) (GRU) Result after trimming.

Figure 12: Result of Sub-Genre Trimming for RNN ((a), (b)), LSTM ((c), (d)), and GRU ((e), (f)), respectively.

Table 4: Four performance results of RNN with the training data type [Genre*ATV]

Trimming	Recall	Precision	Accuracy	F1-score
BT (mean)	0.52	0.77	0.86	0.60
BT (worst)	0.29	0.87	0.85	0.43
AT (worst)	0.37	0.87	0.85	0.52
AT (mean)	0.57	0.75	0.87	0.64

Table 5: Four performance results of LSTM with the training data type [Genre*ATV]

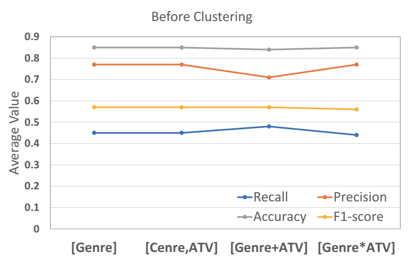
Clustering	Recall	Precision	Accuracy	F1-score
BT (mean)	0.51	0.75	0.86	0.59
BT (worst)	0.27	0.84	0.84	0.41
AT (worst)	0.35	0.84	0.86	0.49
AT (mean)	0.58	0.73	0.87	0.63

Table 6: Four performance results of GRU with the training data type [Genre*ATV]

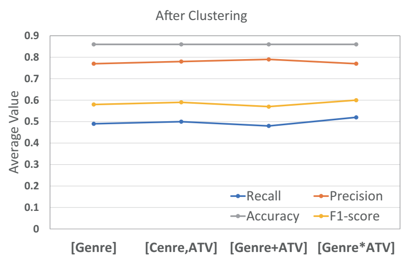
Clustering	Recall	Precision	Accuracy	F1-score
BT (mean)	0.53	0.69	0.84	0.59
BT (worst)	0.35	0.69	0.79	0.46
AT (worst)	0.43	0.70	0.82	0.53
AT (mean)	0.58	0.69	0.85	0.63

4.3.2. Effects of Sub-genre Trimming

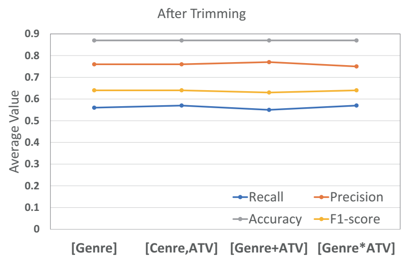
In order to maximize the advantages of clustering, we come up with a method of trimming the sub-genres that are not preferred in the cluster. To see this, we set the threshold η by 0.5 and if there is a cluster that does not exceed 0.5 at any one of Recall, Precision, Accuracy and F1-score, it is subject to trimming. For the result, we consider the following three cases: before clustering, after clustering, and after trimming. We use two kinds of training data types such as [Genre, ATV] and [Genre*ATV] for each model. As shown in Figure 12, we see that most of the metrics for the after trimming case has larger value than others except precision. This is because the precision considers the ratio of correctly predicted positive observations to the total predicted positive observations. Further, we also check that the accuracy is the highest values for all three models. In Table 4-6, we obtain the results of four performance metrics (Recall, Precision, Accuracy and F1-score) with respect to three training models, respectively. In the experiment, we consider the training data as [Genre*ATV] as a representative one. Here, the abbreviations BT means before trimming, and AT means after trimming. AT (best) and AT (worst) indicate the best result and worst result among clusters. Finally AT (mean) present a mean of all result of clusters. As a result, we also check that the performances of AT for four metrics are improved.



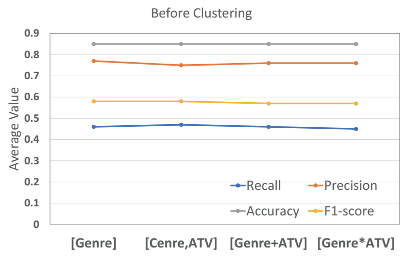
(a) (RNN) Result of ATV.



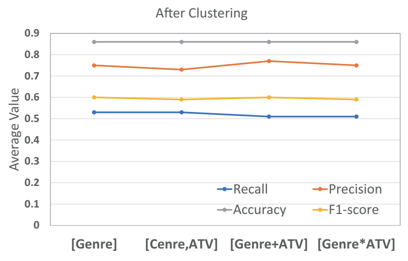
(b) (RNN) Result of ATV.



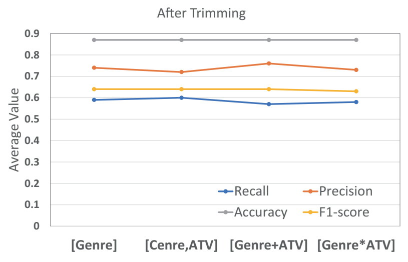
(c) (RNN) Result of ATV.



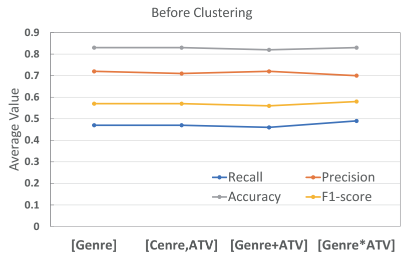
(d) (LSTM) Result of ATV.



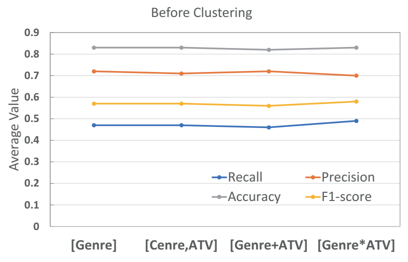
(e) (LSTM) Result of ATV.



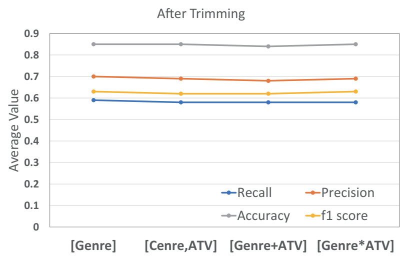
(f) (LSTM) Result of ATV.



(g) (GRU) Result of ATV.



(h) (GRU) Result of ATV.



(i) (GRU) Result of ATV.

4.3.3. Effects of Average Transition Probability

Finally, we obtain recommended performance for the four data types described in the model to examine the effect of the ATV application. In the result, we consider the average evaluation values of all clusters. As result, in Figure 13, we see that there was no significant difference in performance before and after clustering when the transition probability was applied and when the transition probability was not applied. Before clustering, there are no distinct characteristics to consider for all users, so it is seen as an environment that is not good for generating a transition probability that can contain the user's preference. After clustering, users with similar preferences are grouped together, so the ATV would have a good effect, but contrary to expectations, the effect is insignificant. Rather, except for RNN, the performance is slightly higher when the transition probability is excluded. Since RNN learns with more weight on recent information due to the characteristics of the model, it is expected that it will effectively represent the transition probability compared to other models. However, the results show that the effects of clustering and trimming are still quite large.

5. Discussion

A method for genre prediction has been examined as a pre-step for sequential movie recommendation in this work. However, we did not specifically deal with how to make sequential movie recommendations based on this. In fact, in order to solve problems such as cold start due to the limitation of movie data in movie recommendation, [7] also proposed a recommendation system based on correlation information on movie genres. However, this study does not suggest a recommendation method for movies with sequential dynamics. In the sequential movie recommendation system, it is necessary to study how to use the information on the genre of a movie to show the prediction performance well. In addition, it is necessary to design how to select and recommend movies including recommended genres based on the results of sequential movie genre prediction shown in our study. As a learning method for short-term dynamics, we used ATV in Markov Chain. However, the reason why it did not have much effect on the performance is because the RNN-based deep learning we used actually learns some short-term dynamics. Therefore, we will examine whether this short-term dynamic is better estimated by additionally using a higher-order MC that uses information from the past better than the first-order MC that uses the estimation of the next step with the result of the previous step also need. We remain these as our future work.

6. Conclusion

In this paper, we proposed a sequential movie genre prediction algorithm based on the MC for the short-term behavior and RNN for the long-term behavior of user preference. The movie genre prediction does not recommend a specific movie, but it recommends the genre for the next movie to watch in consideration of each user's preference for the movie genre based on the genre included in the movie. For this, we considered that users with similar genre preferences are organized into clusters to recommend genres, and in clusters that do not have relatively specific tendencies, genre prediction has been performed by appropriately trimming genres that are not necessary for recommendation in order to improve performance. We have performed various experiments using our method on well-known movie datasets, and the results showed that clustering and sub-genre trimming worked, but the AVT was not that great.

References

References

[1] R. He and J. McAuley. Fusing similarity models with Markov chains for sparse sequential recommendation. In *Proc. ICDM*, 2016.

[2] W. Kang and J. McAuley. Self-Attentive Sequential Recommendation. In *Proc. ICDM*, 2018.

[3] R. He, W. Kang, J. McAuley. Translation-based Recommendation: A Scalable Method for Modeling Sequential Behavior. In *Proc. IJCAI*, 2018.

[4] B. Hidasi and A. Karatzoglou Recurrent neural networks with Top-k gains for session-based recommendations. In *Proc. ACM Int. Conf. Inf. Knowl. Manage.*, 2018.

[5] S. Wang, L. Cao, Y. Wang, Q. Z. Sheng, M. A. Orgun, and D. Lian. A Survey on Session-based Recommender Systems. In *ACM Computing Surveys*, Vol. 9, No. 4, May 2021.

[6] S. Rendle, C. Freudenthaler, and L. S. Thieme Factorizing personalized Markov chains for next-basket recommendation. In *Proc. WWW*, 2010.

[7] S. Choi, S. Ko and Y. Han. A movie recommendation algorithm based on genre correlations. In *Expert Systems with Applications*, Volume 39, Issue 9, Pages 8079-8085, July 2012.

[8] K. Kim and N. Moon. Recommender system design using movie genre similarity and preferred genres in SmartPhone. In *Multimed Tools Appl*, Volume 61, Issue 1, Pages 87-104, 2012.

[9] A. Zimdars, D. M. Chickering, and C. Meek. Using temporal data for making recommendations. In *Proc. 17th Conference in Uncertainty in Artificial Intelligence,*, 2001.

[10] G. Shani, D. Heckerman, and R. I. Brafman. An mdp-based recommender system. In *Journal of Machine Learning Research,*, 6:1265–1295, 2005.

[11] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Using sequential and non-sequential patterns in predictive web usage mining tasks.. In *Proc. ICDM*, 2002.

[12] E. S. Khorasani, Z. Zhenge, J. Champaign. A Markov chain collaborative filtering model for course enrollment recommendations. In *Proc. Big Data*, 2016.

[13] Y. Koren Collaborative filtering with temporal dynamics. In *Proc. Communications of the ACM*, 2010.

[14] Y. Koren, R. Bell, and C. Volinsky, Matrix factorization techniques for recommender systems. In *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[15] N. Srebro, J. D. M. Rennie, and T. S. Jaakkola Maximum-margin matrix factorization. In *Proc. Adv. Neural Inf. Process. Syst.*, 2005.

[16] R. Salakhutdinov and A. Mnih Probabilistic matrix factorization. In *Proc. Adv. Neural Inf. Process. Syst.*, 2008.

[17] Ba. Hidasi and D. Tikk. General factorization framework for context-aware recommendations. In *Data Mining and Knowledge Discovery*, 30(2):342–371, 2016.

[18] P. Wang, J. Guo, Yanyan Lan, Jun Xu, ShengxianWan, and Xueqi Cheng Learning hierarchical representation model for next basket recommendation. In *Proc. ACM SIGIR Conference on Research and Development in Information Retrieval*, 2015.

[19] S. Wang, L.Hu, Longbing Cao, Xiaoshui Huang, Defu Lian, andWei Liu Attention-based transactional context embedding for next-item recommendation. In *Proc. AAAI*, 2018.

[20] D. Dong, X. Zheng, R. Zhang, and Y. Wang Recurrent collaborative filtering for unifying general and sequential recommender. In *Proc. Int. Joint Conf. Artif. Intell.,*, 2018.

[21] N. Chairatanakul, T. Murata, and X. Liu. Recurrent translation-based network for Top-N sparse sequential recommendation. In *IEEE Access,*, 2019.

[22] C. Wu, A. Ahmed, A. Beutel, Alexander J Smola, and How Jing. Recurrent recommender networks. In *Proc. Web Search and Data Mining*, 2017.

[23] W. Zhao, B. Wang, M. Yang, J. Ye, Z. Zhao, X. Chen and Y. Shen. Leveraging Long and Short-Term Information in Content-Aware Movie Recommendation via Adversarial Training. In *IEEE Transactions on Cybernetics*, Volume: 50, Issue: 11, Nov. 2020.

[24] B. Hidasi, Al. Karatzoglou Session-based recommendations with recurrent neural networks. In *Proc. Learning Representations*, 2016.

[25] F. Yuan, A. Karatzoglou, I. Arapakis, J. M. Jose, and X. He A simple convolutional generative network for next item recommendation. In *Proc.Web Search and Data Mining*, 2019.

[26] S. Wu, Y. Tang, Session-based recommendation with graph neural networks. In *Proc. AAAI*, 2019.

[27] S. Zhang, Y. Tay, L. Yao, A. Sun, and J. An. Next item recommendation with self-attentive metric learning). In *Proc. AAAI*, 2019.

[28] N. Sachdeva, G. Manco, E. Ritacco, and V. Pudi. Sequential Variational Autoencoders for Collaborative Filtering. In *Proc. WSDM*, 2019.

[29] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk and Y. Bengio. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proc. EMNLP*, 2014.

[30] LSTM `https://medium.com/nerd-for-tech/what-is-lstm-peephole-lstm-and-gru-77470d84954b`, Released x/201x.

[31] GRU `https://primo.ai/index.php?title=Gated_Recurrent_Unit_(GRU)`, Released x/201x.

[32] Movielens (ml-25m) `https://grouplens.org/datasets/movielens/25m`, Released 12/2019.