Recurrent Session-approach to Association Rule-based Recommendation

Tubagus Arief Armanda[1)], Ire Puspa Wardhani[1)], Tubagus M. Akhriza[2, *)], Tubagus M. Adrie Admira[1)]

[1)] *Information System Department, STMIK Jakarta STI&K*
Jl. Bri Radio Dalam No.17, Jakarta, Indonesia

[2)] *Information System Department, STMIK Pradnya Paramita (STIMATA)*
Jl LA Sucipto 249A, Malang, Indonesia

How to cite: T. A. Armanda, I. P. Wardhani, T. M. Akhriza and T. M. A. Admira, " Recurrent Session-approach to Association Rule-based Recommendation," *Jurnal Teknologi dan Sistem Komputer*, vol. x, no. x, pp. xx-xx, 2023. doi: 10.14710/jtsiskom.2022.xxxxx [Online].

Abstract –This article introduces an association rule-based recommendation system (RS) using a recurrent neural network approach that is applied when a user browses items in a browsing session. It is proposed to overcome the limitations of the traditional query-based session method which is not adaptive to input, thus is not generative in generating recommendations. Our contribution lies in the training set which is formed from a series of rules and fed to the model, so it can predict the next-items from the input itemID series. The proposed method can adaptively and generatively produce recommendations from a series of items that a user sees in a session. Compared to traditional method, our method can generate recommendations for 100% of item series browsed by user, which traditional methods are also capable of, including those that traditional methods are unable to produce.

Keywords *– association rules; recommendation system, recurrent neural network, long-short term memory*

I. INTRODUCTION

A. Recommender System

The recommendation system (RS) has become a mandatory feature in e-commerce[1]–[3]. This system principally filters large-scale transaction data to produce a list of items that e-commerce application users might like or even buy. An RS generates personalized recommendations for individual users; and this is effective if the user is logged in because the data regarding items that have been purchased or rated by the user personally has been recorded, so that the resulting recommendations can be relevant to user preferences.

For personalized recommendations, we can build a system with a collaborative approach by measuring the similarity of item features that users U like with those of other users[4], [5]; items that have never been rated by U, but rated by other users will be offered to U. The preferences of U are represented by the items vector, I_U which contains the rating value given by U to each item. The similarity of I_U with I_P, the items vector belonging to another user P, is calculated according to the distance formula $d(I_U, I_P)$

If there is no rating data, then the system utilizes the features of items that U once liked or bought. For example, descriptions of films or books[6], [7], or categories or ingredients in food menus[8], [9]. When U is looking for item X with a description of D_X, the system will look for other items, for example Y, with a description of D_Y that similar with D_X. The similarity is measured by a distance formula $d(D_Y, D_X)$. Here D_X and D_Y are presented in features vectors of the items X and Y, respectively. Popular distance calculation formulas include Cosine, Euclidean, Manhattan, and Jaccard coefficient.

B. Association Rule-based Recommender System

In case the application user is not logged in, then the association rule (AR)-based RS can be applied where recommendations are generated from rules X→Y, mined out from transactional data T [6], [10]. X is called the antecedent, and Y is the consequent of the rule and in practice, X and Y are presented in the form of a bag of item IDs (itemID) or itemID vectors. In this article, the term itemID refers to an item with a unique identity code.

Items X and Y are associated not because of the similarity of their descriptions or user-given ratings but on the fulfillment of two main interestingness metrics: the Support and Confidence. Consequently, AR-based RS provides a variety of item recommendations.

Support of X, written as *Sup*(X) as in Eq. (1), represents the number of transactions t_i in T that contain itemset X; and length X can be one or more items. $X \subseteq t \in T$ indicates that the itemset X is a subset of t, the records in T that in principle are also an itemset.

$$Sup(X) = \frac{|t_i \in T|}{|T|}; X \subseteq t \quad (1)$$

Confidence X→Y, written as *Conf*(XY) as in Eq. (2), represents the probability that if X appears in some transactions, then Y also appears.

$$Conf(XY) = \frac{Sup(XY)}{Sup(X)}; X, XY \subseteq t \in T \quad (2)$$

Rules are mined from T if the minimum support (*minsup*) and minimum confidence (*minconf*) thresholds set by the data miner are met. An itemset that satisfies *minsup* is called a *frequent itemset*, and from the explanation above, the itemsets X and Y that make up the rules must be frequent itemsets.

*) Corresponding author (Tubagus M. Akhriza)
Email: akhriza@stimata.ac.id

Submitted: 27 January 2021; Revised: 7 July 2021; Accepted: 10 August 2021; Published: 31 October 2021

C. *Session-approach to AR-based RS*

The problem of AR-based RS is that it does not personalize recommendations to users, thus recommendations are general, monotonous, thus looks unrelated to the item being browsed by U. To improve this limitation, session-based RS is proposed, where session is a virtual time space created when a User browses a web portal URL [11], [12]. Within this time space, the items that the user is or had been looking for, thus assumed as his/her preferences, can be temporarily recorded locally.

Implementing session approach to AR-based RS produces several approaches, as explained in [12]. In the first approach, the rules database is generated from T. Items that users have seen/purchased at recent session, for example $q_U = \{x_1, x_2, x_3\}$ are used as a query to the rules database to find rules X→Y, where $X = \{x_1, x_2, x_3\}$. The items Y obtained are recommended items, if XY satisfied *minsup* and *minconf* thresholds [13].

In the second approach, the method used sequence itemsets that are mined not from T, but from Q i.e., a set of sessions q_i created by the user while browsing the items over some periods, thus $Q = \{q_1, q_2, \ldots, q_{|Q|}\}$ [14], [15]. From the mining, a set of sequence itemsets is obtained and stored in SI = $\{p_1, p_2, \ldots, p_{|SI|}\}$. Assumed, U is currently browsing the items x_i thus creates a session $q_U = \{x_1, x_2, \ldots, x_R\}$, and x_R is the item U saw most recently. If $x_R \in q_U$ and $x_R \in p$ in SI, thus $p = \{\ldots, x_R, x_S, x_{S+1}, \ldots\}$ then p contains the order of items relevant to U's preference. All items that appear after x_R, namely x_S, x_{S+1} and so on are candidate items to be recommended.

However, the traditional query-based session approach for AR-based RS still suffers from some problems. A large number of long frequent itemsets are required, since some subsets of these itemsets are expected to match q_U. Consequently, large enough memory is required to store long itemsets; because the amount is quite significant if the *minsup* threshold is very small [16]. On the other side, if the *minsup* is large, the resulting itemsets tend to be short, and can result in no next items to be recommended. Another problem, especially in the second approach, itemsets sequences are mined only from Q which does not cover all items contained in T; consequently, many items in T are not explored by U. In business, this situation is detrimental to e-commerce owners.

To sum up, traditional query-based session method is not *adaptive* to user input. It is also not *generative* – by means, recommendations are not generated from user search patterns, but are generated from database records, as seen from the monotonous recommendations.

Addressing the above problems, our study proposes a recurrent neural network (RNN)-based session approach to building an item series model, by learning from the rule series generated from T. Our contribution lies in using the rule series as a training set, instead of the item series that customers buy in T. The proposed model can predict next-items from item series patterns in q_U even though it is not a frequent itemset, so it is more adaptive to input. By assuming the series rules as time series data, the model can generatively recommend next-item after next-item that is predicted to appear in the future.

II. METHODS

A. *Recurrent-session approach to AR-based RS*

The proposed next-item prediction method refers to the next-term prediction technique, built using the RNN approach which is capable of handling time series data. A text or phrase is made up of terms, and a term is made up of letters, which are written or typed one letter at a time. As such, a text written can be assumed as time series data as well. Large-scale textual paragraphs, such as a collection of scholarly publications on 'deep learning', are used as a training set for model building. When we write several words as a trigger, for example "recurrent neural", the model can predict the letters that come after the phrase, even forming term by term separated by spaces.

Similarly, association rules mined from purchase transaction data also form a predictive relationship via the confidence metrics, that if product x_1 is purchased, then x_2 is purchased, if x_2 is purchased, then so is x_3, and so on. If it is sorted in such a way based on the highest support and confidence, then the confidence relationship of this rule also forms an item series, namely $x_1 \rightarrow x_2 \rightarrow x_3$, etc. That is, we can also build a model to predict the next item if this rule series is fed to the RNN as a training set. An illustration of model development from this series of rules is presented in Fig. 1.

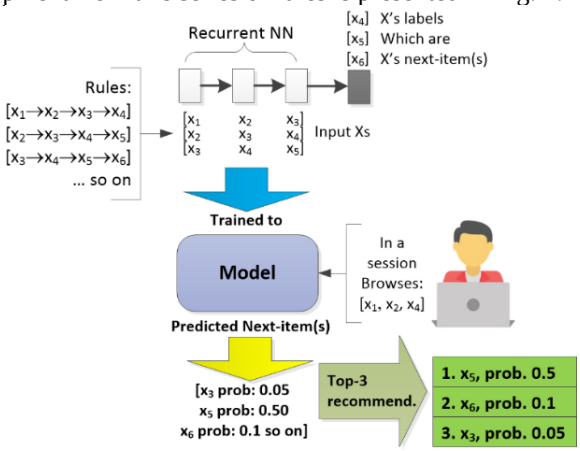

Fig. 1. Illustration of model development from series of rules

On the other hand, items browsed by a user in one session can also be treated as time series data as well. The next item that appears after the browsed item can also be determined, but in our case, this is not by querying the ruleDB but by predicting it via the built model. In Fig. 1, it is illustrated that the user is browsing items $[x_1, x_2, x_4]$ and the model will assign probability values to all items in the rules which become the training set. The total probability is 1, so we can determine the top-K recommendations by sorting the next-items from those with the highest probability ranking.

Submitted: 27 January 2021; Revised: 7 July 2021; Accepted: 10 August 2021; Published: 31 October 2021

The use of a series of sorted rules as a training dataset in building a model is one of the main contributions of the proposed method. Because it implements a recurrent neural network approach to predict the next-item of recurrent browsing sessions, our approach is called the recurrent-session approach to AR-based RS, or ***RS-ARRS***.

B. *Training Set Generation*

Our research is divided into four main activities: generating training dataset (***trainDS***), developing model, recommendation, determining the top-K recommendations, and benchmarking model. The trainDS generation flow is depicted in Fig. 2.

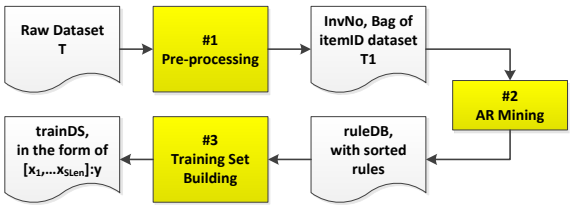

Fig. 2. Training dataset generation flow

Process #1 is pre-processing of raw transaction dataset T, including feature (column) selection which produces a dataset T1 consisting of two columns: invoice number (***invNo***) and itemIDs purchased according to that number. Process #1 also generates a data dictionary containing the itemID and item's description. Examples of records in T1 are as follows:

invNo; ***itemIDs***
000001; x_1, x_2, x_3, x_4
000002; x_1, x_3, x_4, x_5
Etc.

Process #2 is mining the association rules from the itemIDs column in T1, uses Apriori principles, with mining parameters: *minsup*, *minconf* and maximum rule length. The found rules are sorted based on the highest support and confidence, and then stored in the rule database (***ruleDB***).

Process #3, forming a training set from ruleDB. To help make explanations more effective, supposed the rules have been obtained and are sorted based on the highest support and confidence as follows:

$$\{x_1 \rightarrow x_2,\ x_1 \rightarrow x_3,\ x_2 \rightarrow x_4,\ x_3 \rightarrow x_2,\ x_4 \rightarrow x_5,\ x_5 \rightarrow x_6\}$$

After sorting, we make a series of rules with the following notes: 1) the consequence of the rule in the i-th term of the series, becomes the antecedent for the (i+1)-th term. Rules can only be used once to construct a series. An i-th series is made as long as possible by using as many rules as possible; after there are no more rules that can arrange the i-th series, then the (i+1)-th series is arranged in the same way using the rest of the rules. From the previous ordered rule example, the resulting rule series is as follows:

(1) $S_1 = x_1 \rightarrow x_2 \rightarrow x_4 \rightarrow x_5 \rightarrow x_6$, or simplified $S_1 = [x_1,\ x_2,\ x_4,\ x_5,\ x_6]$
(2) $S_2 = x_1 \rightarrow x_3 \rightarrow x_2$, or simplified $S_2 = [x_1,\ x_3,\ x_2]$

The proposed model architecture for next-item prediction applies the Long-short term memory (LSTM) layer, one of the RNN-based layers widely used in handling time series data. An illustration of the time series pattern handled by LSTM in the learning phase is given in Fig. 3.

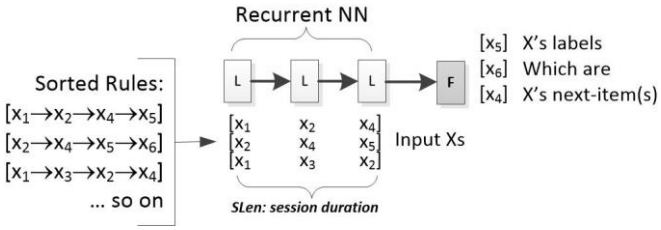

Fig. 3. Time series patterns learned by the model

The model learns the itemID flow pattern as arranged in the rule series in two parts: X and the label of X, namely y. X has a dimension, which is also called series or sequence length (SLen), while the y dimension is one. SLen represents the duration of a session that neurons can remember; in the example above if *SLen* = 3, then in the first session $[x_1, x_2, x_4]$ is X, and x_5 is the y which is the next-item of X. As the session moves forward, X is now $[x_2, x_4, x_5]$ and y is x_6, while x_1 is already out of session and will be forgotten by neurons. In the illustration, the L box represents the LSTM layer, and F box represents the output layer, which is fully connected to the total number of available next-items (labels) i.e., all itemIDs in ruleDB.

If X is stored in an array, or list in Python language, then the above explanation also implies that shifting the session forward (towards the right), algorithmically, pops up the itemID on the leftmost X, pushes itemID y to the rightmost X, and assigns a new next-item y as label for the new X. This algorithm also describes the mechanism for forming a training dataset (trainDS) from series of rules that have been built.

For given a series of rules S, Session duration *SLen* = 3, and ruleDB, do the following stages:

1. Initialization: aims to create an initial record in the form X:y, with X's length = *SLen* and y's length = 1. The following steps are performed
   a. X = S[0 : SLen] #Python's way to take S[0] to S[*SLen*-1] as X
   b. y = S[SLen] # set S[*SLen*] as y.
   c. idx = SLen # index last accessed from S
2. Shifting: aims to generate the next record from the previous X by shifting the session forward:
   a. X = X.pop(0) # pop the leftmost X value
   b. X.append(y) # push y into the rightmost X
3. Labelling: aims to labelling the new record X with y,
   a. idx +=1 # increase index of S
   b. y = S[idx] # set S[idx] as y

Submitted: 27 January 2021; Revised: 7 July 2021; Accepted: 10 August 2021; Published: 31 October 2021

c. S = S[idx:] # trim S starting from index 0 to idx

S is trimmed so that the shifting step can be repeated. However, if all entries in S have been used, so that S becomes empty, then the formation of training data from a rule series S also ends.

The training data of a rule series is said to be complete if all Xs with length *SLen* and its label y have been developed. However, in practice, because the value of *SLen* can vary (depending on the needs of model development), then all S entries have been accessed even though the length of X has not yet reached SLen. An example of this case is $S_2 = [x_1, x_3, x_2]$, where all entries in S_2 can only form X, but it does not yet have a label y. When this case arises, the fourth stage must be done as follows:

4. Padding: aims to complete entry X so that it has the length SLen, and has a label y. The steps are as follow:
   a. While the length of X < SLen:
      1 Search for rules $\acute{X} \rightarrow \acute{Y}$ in ruleDB, where $\acute{X}$ = X[-1].
      2 If found: X.append($\acute{Y}$).
   b. If the length of X == SLen, then the formation of X is complete, and
      i. Continue search from the last position for the rule $\acute{X} \rightarrow \acute{Y}$, if $\acute{X}$ = X[-1] then set label y = $\acute{Y}$

The result of the padding step for S_2 is $X = [x_1, x_3, x_2]$ and y = x_4.

C. *Model Development*

Like the text generator model, the proposed next-item prediction model is also a generative model – a model that can generate predictions for next-items for several sessions in the future.

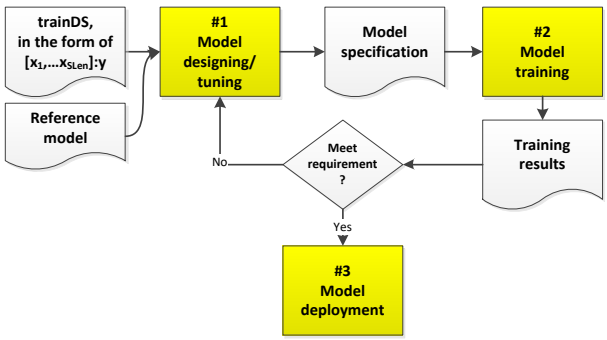

Fig. 4. Model development lifecycle flow

The flow of model development in our study is given in Fig. 4, which forms a cycle as described in [17]. The trainDS and existing reference models are materials for designing and tuning models. Models that have met the requirements in terms of loss and accuracy will be deployed to an implementable recommendation system. If it does not meet the requirements, then the model will be redesigned which includes the composition of the layers and neuron cells, as well as the number of epochs and batches in the training process. Our requirement is to have a model that has level of loss < 0.5, and accuracy > 80%.

The neural network layers that make up the model are divided into three parts where the term is used in reference to the Keras library for Python:

1. Input layer with dimension (*SLen*, 1), with *SLen* = 3 which is the dimension of X, and 1 is the dimension of *y*.
2. Hidden layers, for observation purposes, we use one to three LSTM layers, and what is observed is the loss and accuracy values for the addition of each layer. Each LSTM layer is followed by a dropout layer which serves to remove cells that contribute to overfitting. The number of neurons is set to 256. The activation function applied is *Tanh*.
3. Output layer, which uses the Dense layer after the LSTM layers. This layer is called the fully connected layer to the output, which in our case, the output dimension is the number of itemIDs as they are all potentially next items. The activation function used is Softmax.

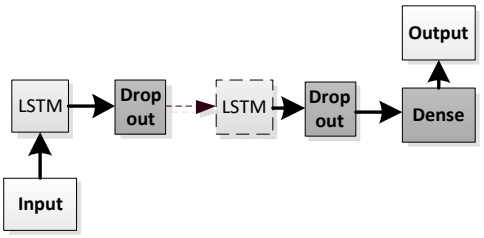

Fig. 5. Proposed model design

LSTM is an RNN type layer designed to handle time series data. LSTM has main components: cell, input gate, output gate and forget gate [18], [19]. Cells that function to remember past patterns in a series or sequence. This is useful for remembering contexts that appeared in the past, to be combined with current information in order to forecast patterns that will occur in the future. The past memory duration that will be remembered by the LSTM layer is specified in the sequence or series length. LSTM can produce generative predictions, where the model is able to generate new samples from the same data distribution [20], [21]. For example, given a reading book as training data, a generative model for text-generator can generate several words that will appear after a series of words is given as a trigger, so that sentences that are composed look new. To be able to do this, the model requires the entire training data to be studied [22], [23].

A summary of the model using one layer of LSTM + Dropout + Dense is given in Fig. 6. A summary of models using two and three LTSM is not given, but intuitively it can be understood from Fig. 6. All itemID series in trainDS are used as training data, without a test dataset because the model built is a generative model that must learn the probability of each itemID in series of itemID in a whole trainDS. The design of this model is implemented with the Keras library using a functional modelling. The layer composition in each model is

Submitted: 27 January 2021; Revised: 7 July 2021; Accepted: 10 August 2021; Published: 31 October 2021

compiled by applying categorical_crossentropy loss, optimizer Adam. After compilation, the model is fitted to all vectors X and y, with 1000 epochs in 8 batches.

```
Model: "Model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 INPUT (InputLayer)          [(None, 3, 1)]            0

 LSTM_1 (LSTM)               (None, 256)               264192

 DROP-OUT (Dropout)          (None, 256)               0

 OUTPUT (Dense)              (None, 208)               53456

=================================================================
Total params: 317,648
Trainable params: 317,648
Non-trainable params: 0
```

Fig. 6. Summary of Model with one LSTM layer

D. *Top-K Recommendation Determination*

Briefly, the procedure carried out to generate next-items predictions is as follows:

1 Prepare the inputs:
   a) matrix Xs of all arrays X in trainDS in which X has with shape (*SLen*, 1). If the number of records in trainDS is N, then the Xs dimension is (N, SLen, 1)
   b) Matrix Y i.e., X's label with shape (N, 1)
2 Compile the layers arrangement into a model M
3 Perform training by fitting data Xs to data Y with M, usually written M.fit(Xs, Y, number of epochs, number of batches), save the fitting with the lowest loss (along with accuracy) into ***M_best*** or an external file M.hdf5
   #M_best is now ready to predict any series of itemID as input
4 INPUT "enter a series of itemID as input"
5 PREDICTION = *M_best.predict*(INPUT)
6 PREDICTION is obtained in the form of a matrix containing the probability predictions for each itemID to become the next-items

The process of determining the top-K recommendations from PREDICTION is given as follows.

1 Determine the value of K;
2 Sort the probabilities in the PREDICTION array from the highest value, noting that each element in the array represents an itemID index in the itemID-Description data dictionary.
3 Get the first K index of itemID in the array,
4 Print item descriptions in itemID order
5 K recommendation items obtained

E. *Model Benchmarking*

The activity flow of benchmarking the model is given in Fig. 7, which shows that the proposed model is compared with the query-based session method. The aspect being compared is the ability of the model to always be able to get predictions of the probabilities of all itemIDs to become the next-item with respect to the items that the user is looking for in a session. We run two test scenarios to examine both methods in terms of their adaptability in generating next-item recommendations

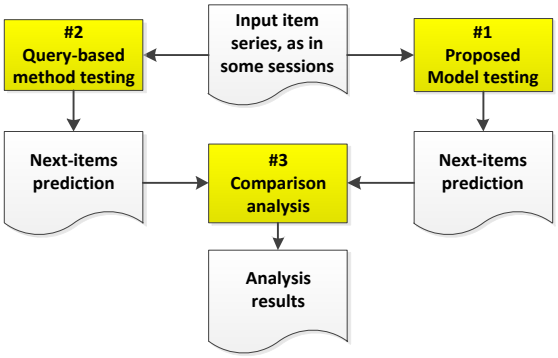

Fig. 7. Model benchmarking flow

Test #1: the rule that produces next-item in the query-based method is tested on the proposed method. The steps are as follow:

7 Generate all rules X→Y with |X| = 3 and |Y| = 1,
8 Each X in the rules has at least one next-item Y
9 Enter all the Xs as the input for the proposed method, and get the top-10 recommendations.
10 Count the number of X that have top-10 recommendations
11 If all X have top-10 recommendations then the proposed model is adaptive to all query-based method inputs

Test #2: combining several items that can produce recommendations through proposed items, then used as a query to find recommendations in the traditional method.:

6 Simulate one 3-item series as input for the query-based method to find rules X→Y, where X is equal to respective 3-item series
7 If traditional query-based method cannot produce recommendations, then it is not an adaptive method.

III. Results and Discussions

F. *Dataset and Settings*

The dataset used is Online retail data available on the UCI web portal. The number of records initially was 541,909, but after grouping by invoice number, the number of records became 22,106, consisting of 4059 unique items.

In order to make the proposed method can be compared fairly with the query-based session method, the rules are mined with *minsup* = 1% and *minconf* = 50%. Using a lower *minsup* and *minconf* such as 0.1% and 10% respectively, results in an explosion of the number of rules to more than 2M rules, which is not effective for demonstrating the features and functionality of the proposed method and of the compared traditional method as well.

The difference between our approach and traditional AR-based RS methods is that we only mine rules with X and Y lengths of exactly one item, or |X| = 1 and |Y| = 1; whereas the traditional method we apply 0 < |X| ≤ 3 and |Y| = 1. We take this approach with the following considerations, First, with short rules, the number of rules

Submitted: 27 January 2021; Revised: 7 July 2021; Accepted: 10 August 2021; Published: 31 October 2021

that must be maintained in memory is less than long rules. For our approach, the number of rules generated is 194 rules which are then arranged as a series of rules which are used as the training dataset. Size of training dataset becomes 824 records. For the traditional method approach, the resulting rules are 194 rules, of which 40 rules have |X| = 3 and |Y| = 1 which is used for Test 1. Mining results for this traditional method are stored in ***ruleDB-trad***.

G. *Results and Discussion*

The results of applying 1 to 3 layers of LSTM show no significant difference between loss and accuracy. The lowest loss values for each application, respectively are 0.2234, 0.2163 and 0.3118 with an accuracy of 84.2%, 83.8% and 84.4%. Charts of changes in loss and accuracy for each epoch for these three treatments are given in Fig. 8, but this is only for models with 1 LSTM layer, while other models can be understood from this figure.

The results of test #1 show that the proposed method can predict next-items and produce top-10 recommendations for all 40 three-item X series where the query-based method can generate next-items. In contrast, the query-based method is not able to generate top-10 recommendations for all X, but only 2 items, as shown in Table 1. This is because not all X which is the antecedent of the rules has 10 consequent items Y. This is an advantage offered by our proposed method.

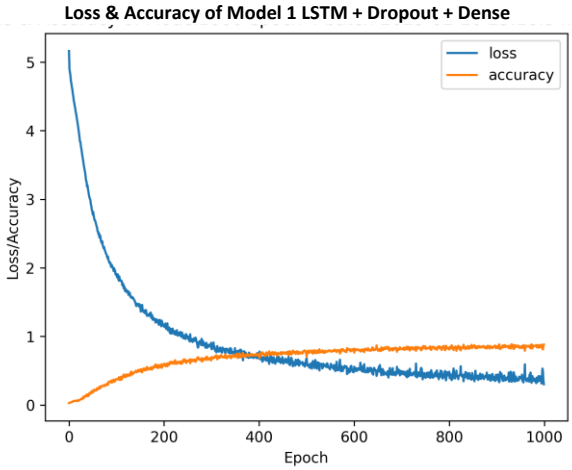

Fig. 8. Loss and accuracy of Model with 1 LSTM + Dropout + Dense layers

Table 1. Top-10 recommendation produced by two compared methods

itemID	Description
Items seen by User	
85099B	Jumbo Bag Red Retrospot
20712	Jumbo Bag Woodland Animals
21931	Jumbo Storage Bag Suki
Top-10 Recommendation by proposed method	
84731	3 Birds Canvas Screen
22950	36 Doilies Vintage Christmas
90199B	5 Strand Glass Necklace Amethyst
22580	Advent Calendar Gingham Sack
21143	Antique Glass Heart Decoration
17164B	Ass Col Small Sand Gecko P'Weight
47421	Assorted Colour Lizard Suction Hook
20749	Assorted Colour Mini Cases
47420	Assorted Colour Suction Cup Hook
84950	Assorted Colour T-Light Holder
Top-2 Recommendation by traditional method	
22386	Jumbo Bag Pink Polkadot
22411	Jumbo Shopper Vintage Red Paisley

For test #2, through manual inspection we found several item combinations that are not in the *ruleDB-trad* database. These items are fed to the developed model, to seek recommendations. One of the results is given in Table 2, where the proposed method produces top-10 recommendations, and the traditional method does not find any items. This means that traditional query-based methods are not adaptive in generating recommendations for any input itemIDs entered.

Table 2. Top-10 recommendation produced by proposed method

itemID	Description
Items seen by User	
85099B	Jumbo Bag Red Retrospot
20711	Jumbo Bag Toys
20712	Jumbo Bag Woodland Animals
Top-10 Recommendation by Proposed Method	
22282	12 Egg House Painted Wood
84559B	3d Sheet of Cat Stickers
72801C	4 Rose Pink Dinner Candles
22371	Airline Bag Vintage Tokyo 78
23068	Aluminium Stamped Heart
90183A	Amber Drop Earrings W Long Beads
84879	Assorted Colour Bird Ornament
20749	Assorted Colour Mini Cases
47420	Assorted Colour Suction Cup Hook
84950	Assorted Colour T-Light Holder
Recommendation by traditional method	
-	Null

Next, we demonstrate the proposed method's ability to generatively find recommendations for each input given in a session. The mechanism is 1) get the top-K recommendations from the itemIDs series, called X1, 2) the itemID in the top recommendation position is assumed to be clicked by the user, so it enters the list of items the user sees, and then this series becomes X2. 3) The second step is repeated until X5 is obtained, then the results are analyzed.

Using K = 3, the result is shown as follows:

X1:
85099b, Jumbo Bag Red Retrospot
20711, Jumbo Bag Toys
20712, Jumbo Bag Woodland Animals
Top-3 recommendations:
23697, A Pretty Thank You Card (**Clicked**)
85161, Acrylic Geometric Lamp

Submitted: 27 January 2021; Revised: 7 July 2021; Accepted: 10 August 2021; Published: 31 October 2021

22915, Assorted Bottle Top,Magnets

X2:
23697, A Pretty Thank You Card
85099b, Jumbo Bag Red Retrospot
20712 ,Jumbo Bag Woodland Animals
Top-3 recommendations:
22282, 12 Egg House Painted Wood (**Clicked**)
22374, Airline Bag Vintage Jet Set Red
22915, Assorted Bottle Top Magnets

X3
22282, 12 Egg House Painted Wood
23697, A Pretty Thank You Card
20712, Jumbo Bag Woodland Animals
Top-3 recommendations:
84558a, 3d Dog Picture Playing Cards (**Clicked**)
22915, Assorted Bottle Top Magnets
84879, Assorted Colour Bird Ornament

X4
22282, 12 Egg House Painted Wood
84558a, 3d Dog Picture Playing Cards
23697, A Pretty Thank You Card
Top-3 recommendations:
21448, 12 Daisy Pegs In Wood Box (**Clicked**)
23442, 12 Hanging Eggs Hand Painted
22906, 12 Message Cards With Envelopes

X5
21448, 12 Daisy Pegs In Wood Box
22282, 12 Egg House Painted Wood
84558a, 3d Dog Picture Playing Cards
Top-3 recommendations:
22436, 12 Coloured Party Balloons
22150, 3 Stripey Mice Feltcraft
84559a, 3d Sheet Of Dog Stickers

Here we can see the ability of the proposed RNN-based session method to generatively and adaptively produces recommendation after recommendation from a series of items viewed by a user in a session. Traditional query-based methods do not have the capability for this because next-item recommendations are not generated from the learning process but instead rely on rules. As a result, when the item array that a user is looking for in a session is not a frequent itemset, then the traditional method fails to find the next-item, hence also recommendations.

Another important note found in the experiment is that if the dropout layer is not applied, then there is an improvement in loss, which is an average of 0.07 and an average accuracy of 93.7%. It is explained that Dropout can avoid overfitting by deleting cells randomly[24]. However, in some literature regarding text-generators, no comparison was found between the results of the dropout and non-dropout models[18], [23], [25]. New words are generated from the predicted next-characters, but many of them seem meaningless, or at first glance like typos. In addition, because of its generative nature, the text-generator method results in the formation of new sentences from new word arrangements, so that the 'accuracy' of words that should appear after the previous word, intuitively does not result from applying the dropout layer only, but also the richness of vocabulary and sentences available in the training set.

In line with the challenge of 'accuracy' in the text generator, because the next itemIDs series is generated by the proposed model through a learning process, the recommended items are more varied, besides the appearance of one itemID after another refers to the relationship pattern of the Confidence metric between itemIDs that builds the training dataset.

IV. CONCLUSIONS

This paper demonstrates the ability of the proposed RNN-based session approach to AR-based RS to find item recommendations in a generative and adaptive manner. It can find recommendations from all item arrays where traditional query-based methods can find it, including those that cannot be found by the traditional method. The resulting recommendations appear to vary, but follow the pattern of items arranged in a series of rules based on the Confidence and Support values between items

REFERENCES

[1] T. Silveira, M. Zhang, X. Lin, Y. Liu, and S. Ma, "How good your recommender system is? A survey on evaluations in recommendation," *Int. J. Mach. Learn. Cybern.*, vol. 10, no. 5, 2019, doi: 10.1007/s13042-017-0762-9.

[2] J. Ben Schafer, J. A. Konstan, and J. Riedl, "E-Commerce Recommendation Applications," in *Applications of Data Mining to Electronic Commerce*, 2011.

[3] F. O. Isinkaye, Y. O. Folajimi, and B. A. Ojokoh, "Recommendation systems : Principles , methods and evaluation," *Egypt. Informatics J.*, vol. 16, no. 3, pp. 261–273, 2015, doi: 10.1016/j.eij.2015.06.005.

[4] Q. Y. Shambour, M. M. Abu-Alhaj, and M. M. Al-Tahrawi, "A hybrid collaborative filtering recommendation algorithm for requirements elicitation," *Int. J. Comput. Appl. Technol.*, vol. 63, no. 1–2, 2020, doi: 10.1504/IJCAT.2020.107908.

[5] W. Jiang *et al.*, "A new time-aware collaborative filtering intelligent recommendation system," *Comput. Mater. Contin.*, vol. 61, no. 2, pp. 849–859, 2019, doi: 10.32604/cmc.2019.05932.

[6] K. Yi, T. Chen, and G. Cong, "Library personalized recommendation service method based on improved association rules," *Libr. Hi Tech*, vol. 36, no. 3, pp. 443–457, 2018, doi: 10.1108/LHT-06-2017-0120.

[7] Y. Tian, B. Zheng, Y. Wang, Y. Zhang, and Q. Wu, "College library personalized recommendation system based on hybrid recommendation algorithm," in *Procedia CIRP*, 2019, vol. 83, doi: 10.1016/j.procir.2019.04.126.

[8] M. Ge, F. Ricci, and D. Massimo, "Health-aware food recommender system," 2015, doi: 10.1145/2792838.2796554.

[9] X. Li *et al.*, "Application of intelligent recommendation techniques for consumers' food choices in restaurants," *Front. Psychiatry*, 2018, doi: 10.3389/fpsyt.2018.00415.

[10] L. D. Adistia, T. M. Akhriza, and S. Jatmiko, "Sistem Rekomendasi Buku untuk Perpustakaan Perguruan Tinggi Berbasis Association Rule," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 3, no. 2, 2019, doi: 10.29207/resti.v3i2.971.

[11] T. K. Dang, Q. P. Nguyen, and V. S. Nguyen, "A Study of Deep Learning-Based Approaches for Session-Based Recommendation Systems," *SN Computer Science*, vol. 1, no. 4. 2020, doi: 10.1007/s42979-020-00222-y.

[12] S. Wang, L. Cao, Y. Wang, Q. Z. Sheng, M. A. Orgun, and D. Lian, "A Survey on Session-based Recommender Systems," *ACM Comput. Surv.*, vol. 54, no. 7, 2022, doi: 10.1145/3465401.

[13] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa, "Effective

Submitted: 27 January 2021; Revised: 7 July 2021; Accepted: 10 August 2021; Published: 31 October 2021

Personalization Based on Association Rule Discovery from Web Usage Data," 2001, doi: 10.1145/502932.502935.
[14] G. E. Yap, X. L. Li, and P. S. Yu, "Effective next-items recommendation via personalized sequential pattern mining," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2012, vol. 7239 LNCS, no. PART 2, doi: 10.1007/978-3-642-29035-0_4.
[15] U. Niranjan, R. B. V. Subramanyam, and V. Khanaa, "Developing a Web Recommendation System Based on Closed Sequential Patterns," in *Communications in Computer and Information Science*, 2010, vol. 101, doi: 10.1007/978-3-642-15766-0_25.
[16] T. M. Akhriza, Y. Ma, and J. Li, "Revealing the Gap Between Skills of Students and the Evolving Skills Required by the Industry of Information and Communication Technology," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 27, no. 05, pp. 675–698, 2017, doi: 10.1142/s0218194017500255.
[17] H. Miao, A. Li, L. S. Davis, and A. Deshpande, "Towards unified data and lifecycle management for deep learning," 2017, doi: 10.1109/ICDE.2017.112.
[18] M. Lippi, M. A. Montemurro, M. Degli Esposti, and G. Cristadoro, "Natural Language Statistical Features of LSTM-Generated Texts," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 30, no. 11, 2019, doi: 10.1109/TNNLS.2019.2890970.
[19] A. Paszke, "LSTM implementation explained," *Web Page*, 2015.
[20] C. L. P. Chen and S. Feng, "Generative and Discriminative Fuzzy Restricted Boltzmann Machine Learning for Text and Image Classification," *IEEE Trans. Cybern.*, vol. 50, no. 5, 2020, doi: 10.1109/TCYB.2018.2869902.
[21] A. Fujino, N. Ueda, and K. Saito, "A hybrid generative/discriminative approach to text classification with additional information," *Inf. Process. Manag.*, vol. 43, no. 2, 2007, doi: 10.1016/j.ipm.2006.07.013.
[22] J. Brownlee, "Text Generation With LSTM Recurrent Neural Networks in Python with Keras," *Machine learning mastery*, 2018. https://machinelearningmastery.com/text-generation-lstm-recurrent-neural-networks-python-keras/.
[23] T. Iqbal and S. Qureshi, "The survey: Text generation models in deep learning," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 6. 2022, doi: 10.1016/j.jksuci.2020.04.001.
[24] A. Carta, "Building a RNN Recommendation Engine with TensorFlow," *Medium.com2*, 2021. https://medium.com/decathlontechnology/building-a-rnn-recommendation-engine-with-tensorflow-505644aa9ff3 (accessed Jan. 06, 2022).
[25] R. Patel and S. Patel, "Deep Learning for Natural Language Processing," in *Lecture Notes in Networks and Systems*, vol. 190, 2021, pp. 523–533.

Submitted: 27 January 2021; Revised: 7 July 2021; Accepted: 10 August 2021; Published: 31 October 2021