KEDS Knowledge Engineering and Data Science (KEDS)
Vol 6, No 2, October 2023, pp. 199–214

pISSN 2597-4602
eISSN 2597-4637

Recurrent Session Approach to Generative Association Rule based Recommendation

Tubagus Arief Armanda [a,1], Ire Puspa Wardhani [a,2], Tubagus M. Akhriza [b,3,*], Tubagus M. Adrie Admira [a,4]

[a] *STMIK Jakarta STI&K*
Jl. Bri Radio Dalam No.17, Jakarta Selatan 12140, Indonesia
[b] *STMIK Pradnya Paramita (STIMATA)*
Jl. Laksda Adi Sucipto 249A, Malang 65126, Indonesia
[1] *tb_armanda@yahoo.com;* [2] *irepuspa@gmail.com;* [3] *akhriza@stimata.ac.id*;* [4] *adrie.admira@jak-stik.ac.id*
** corresponding author*

ARTICLE INFO

Article history:
Received 09 July 2023
Revised 29 July 2023
Accepted 30 October 2023
Published online 02 November 2023

Keywords:
Association Rules
Recommendation System
Recurrent Neural Network
Long-Short Term Memory
Session-Based Recommendation

ABSTRACT

This article introduces a generative association rule (AR)-based recommendation system (RS) using a recurrent neural network approach implemented when a user searches for an item in a browsing session. It is proposed to overcome the limitations of the traditional AR-based RS which implements query-based sessions that are not adaptive to input series, thus failing to generate recommendations. The dataset used is accurate retail transaction data from online stores in Europe. The contribution of the proposed method is a next-item prediction model using LSTM, but what is trained to develop the model is an associative rule string, not a string of items in a purchase transaction. The proposed model predicts the next item generatively, while the traditional method discriminatively. As a result, for an array of items that the user has viewed in a browsing session, the model can always recommend the following items when traditional methods cannot. In addition, the results of user-centered validation of several metrics show that although the level of accuracy (similarity) of recommended products and products seen by users is only 20%, other metrics reach above 70%, such as novelty, diversity, attractiveness and enjoyability.

I. Introduction

The Recommendation System (RS) has become a mandatory feature in e-commerce [1][2][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, an RS can be built 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 is similar to D_X. The similarity is measured by a distance formula $d(D_Y, D_X)$. Here D_X and D_Y are presented in feature vectors of the items X and Y, respectively. Popular distance calculation formulas include Cosine, Euclidean, Manhattan, and Jaccard coefficients. These collaborative and content filtering approaches are practical if the user has logged into the system, where SR then scans the database of transactions the user has made with items in the store.

https://doi.org/10.17977/um018v6i22023p199-214

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 as a bag of item IDs (itemID) or itemID vectors. In this article, itemID refers to an item with a unique identity code. Item X and Y are associated not because of the similarity of their descriptions or user-given ratings but on fulfilling two main interestingness metrics: Support and Confidence. Consequently, AR-based RS provides a variety of item recommendations.

Support of X, written as *Sup*(X) as in (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.

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

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

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

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.

The problem of AR-based RS is that it does not personalize recommendations to users, thus recommendations are general, monotonous, thus look unrelated to the item being browsed by U. To improve this limitation, session-based RS is proposed, where the session is a virtual time-space created when a user browses a web portal URL [11]–[16]. 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 [14]–[16]. Some methods use Markov chains [17]–[19], artificial neural networks [11], [20]–[22], and association rule learning approaches [23]–[28] to develop session-based RS.

Implementing a session approach to AR-based RS produces several approaches, as explained in. In the first approach, the rules database is generated from T. Items users have seen/purchased at recent sessions, 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 [23], [29].

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 $Q = \{q_1, q_2, \ldots, q_{|Q|}\}$ [27], [28]. 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 minimal [30]. On the other side, if the minsup is large, the resulting itemsets tend to be short and can result in no following 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 methods are not adaptive to a series of items the user visits, so recommendations look monotonous. Traditional methods also cannot generate recommendations

from a series of input items that are not frequent because they refer to the rule database, while rules are composed of frequent itemsets only.

This study was conducted with the objectives of building a generative model based on Recurrent Neural Network (RNN) and association rules which can predict the next item generatively from a series of items that the user has visited in a browsing session even though this series of items is not a frequent itemset.

Applying RNN to session and AR-based RS, this method is called the Recurrent-session approach to AR-based RS, or RS-ARRS. The model is built using Long-Short-Term Memory (LSTM), a type of layer in RNN, and dropout layers. The novelty of this model is that the dataset that is trained is not a series of items that customers have purchased but a series of rules that are arranged according to the Support and Confidence of the rules. The series of items visited by the user in a browsing session is considered an input prompt for the model, and the model responds by generatively predicting the items that will appear next.

The rest of the paper is structured as follows. In the Methods section, the proposed approach is explained, followed by a discussion of generating a training set for the model. After that, the flow of the model development cycle is explained, including the proposed model design. Experiments on model benchmarking were organized with the aim of testing and comparing the performance of the proposed model with traditional models. After that, the experimental results are discussed in the Results and Discussion section. The article concludes with conclusions and recommendations for future research.

II. Methods

A. Research Framework

The framework of the proposed method is explained using Figure 1, which is divided into four main activities: a) generating the training dataset (trainDS), b) developing the proposed model, c) determining the top-K recommendations, d) benchmarking the model and e) validating the recommendation. Before explaining the steps for creating a training set, the basic idea of the proposed approach is explained first.

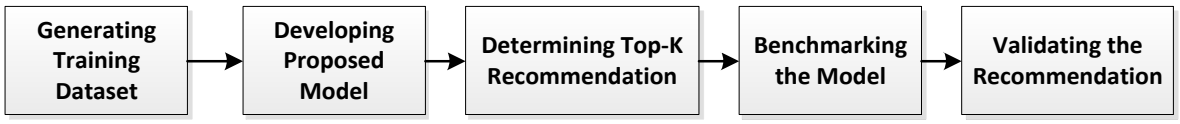

Fig. 1. Research framework

RNN is usually used to estimate a next-value in the future by learning time series of data in the past and present [17], [31], [32], so how does RNN predict the next term of a current sentence? Intuitively, a sentence 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. For example, large-scale textual paragraphs, such as a collection of scholarly publications on deep learning, are used as a training set for model building. Given an input prompt such as "recurrent neural net" to the model, the model predicts the appearance of the following letter or term, referring to all the text in the dataset of deep learning publication. The nature of the prediction is generative because the sentences formed are composed of new terms [33], [34]. While generative predictions are formed by modeling the probability distribution of the entire input data domain, a discriminative prediction aims to differentiate or classify input data into specific categories or labels [35]–[38]. Some examples include sentiment analysis or textual classification.

Several studies explain that RNN predicts the next item in a market basket. RNN that uses time series data can be used to predict the next item with the assumption that the user picks up item by item and puts it in the shopping cart following a particular time series [32], [33], [39], [40]. In another perspective, items viewed sequentially within a browsing session can also be considered time-series data [5], [17], [20], [41], [42]. However, the next-item prediction model that learns items that have been purchased still have weaknesses, namely that the process of recording items by the cashier (both

in an offline and online store) is carried out randomly and ignores the order in which the customer picks up the items. As a result, the time series nature of the items picked up by customers is lost.

In this study, as explained via Figure 2, a solution to this limitation is also included in the proposed model generation. Time-series training data is not created from item purchase transaction data but from associative rules mined from transaction data. The rules also form a predictive relationship via the confidence metrics that if an item 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 $x_1 \rightarrow x_2 \rightarrow x_3$. Similarly, a model can be built to predict the next item if this rule series is trained to the RNN. There is no percentage division between the training and testing set because it models the probability distribution of the entire input data domain to form a generative model. The model then produces the probability of all existing items as next-items with a total probability of one. By ranking these probabilities, top-k item recommendations are obtained.

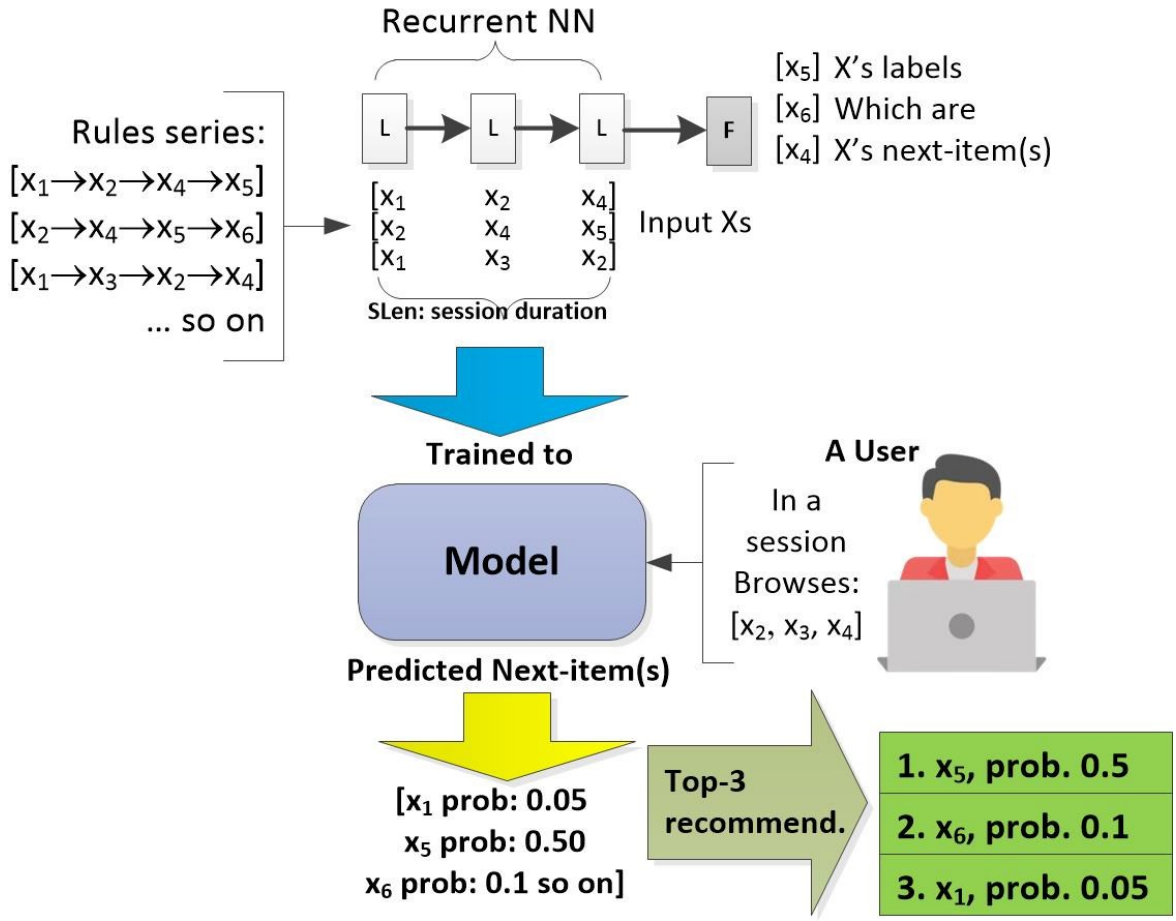

Fig. 2. Illustration of model development from series of rules

For illustration, as in Figure 2, the series of items visited by the user in a browsing session, e.g. $[x_1, x_2, x_4]$, is considered an input prompt for the model, and the model responds by generatively predicting the items that will appear next, similar to how generative text-generation works. All items have a certain probability of being the next-item, and a computer program will sort these probabilities to get, for example, the top 3 items that are the next-item recommended to the user.

B. Generating Training Dataset

Training dataset generation is described in Figure 3. 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.

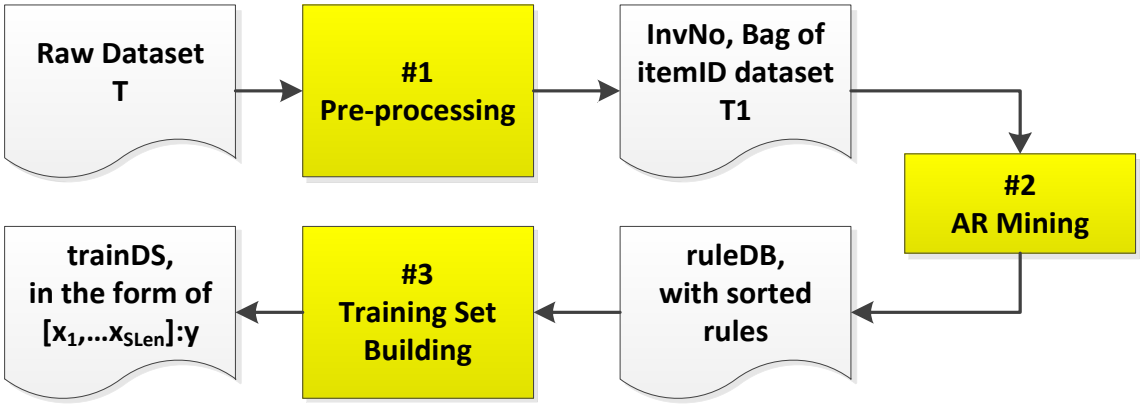

Fig. 3. Training dataset generation flow

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. 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, a series of rules are created with the following notes: 1) the consequence of the rule in the i-th term 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 no more rules can arrange the i-th series, the (i+1)-th series is the same way using the rest of the rules. From the previous ordered rule example, the resulting rule series is as in (3) and (4).

$S_1 = x_1 \rightarrow x_2 \rightarrow x_4 \rightarrow x_5 \rightarrow x_6$, or simplified $S_1 = [x_1,\ x_2,\ x_4,\ x_5,\ x_6]$ (3)

$S_2 = x_1 \rightarrow x_3 \rightarrow x_2$, or simplified $S_2 = [x_1,\ x_3,\ x_2]$ (4)

An illustration of the rule series pattern handled by LSTM in the learning phase is given in Figure 4. 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 a 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.

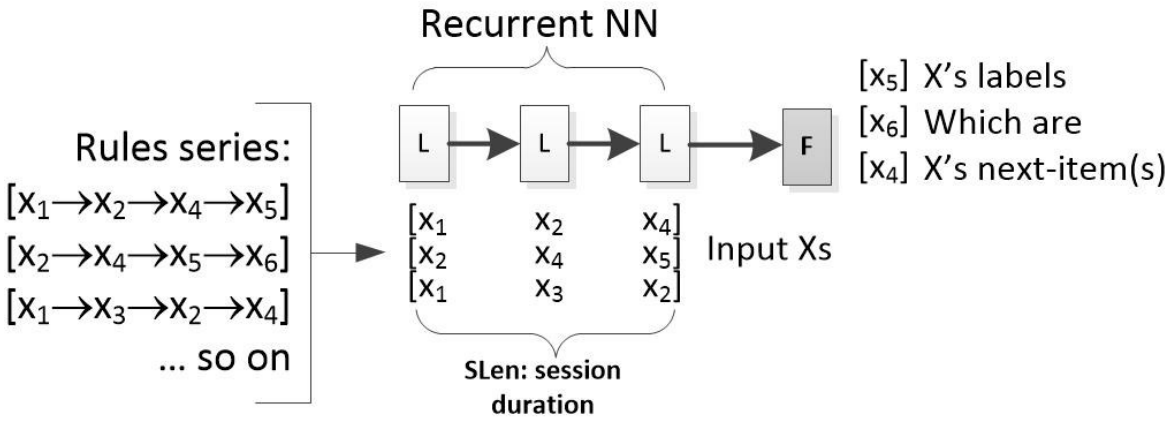

Fig. 4. Rule series patterns learned by the model

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

- 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
  1. X = S[0 : SLen] #Python's way to take S[0] to S[*SLen*-1] as X
  2. y = S[SLen] # set S[*SLen*] as y.
  3. idx = SLen # index last accessed from S
- Shifting: aims to generate the next record from the previous X by shifting the session forward:
  1. X = X.pop(0) # pop the leftmost X value
  2. X.append(y) # push y into the rightmost X
- Labelling: aims to labelling the new record X with y,
  1. idx +=1 # increase index of S
  2. y = S[idx] # set S[idx] as y
  3. 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), 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:

- Padding: aims to complete entry X so that it has the length SLen, and has a label y. The steps are as follows:
  1. 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($\acute{Y}$).
  2. If the length of X == SLen, then the formation of X is complete, and
     i. Continue searching 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. Developing Proposed Model

Like the text generator model, the proposed next-item prediction model is also generative – a model that can generate predictions for next items for several sessions in the future. The flow of model development in this study is given in Figure 5, which forms a cycle as described in [43]. The trainDS and existing reference models are materials for designing and tuning models. Models that have met the requirements regarding 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. The requirement is to have a model with a loss level < 0.5, and an accuracy > 80%.

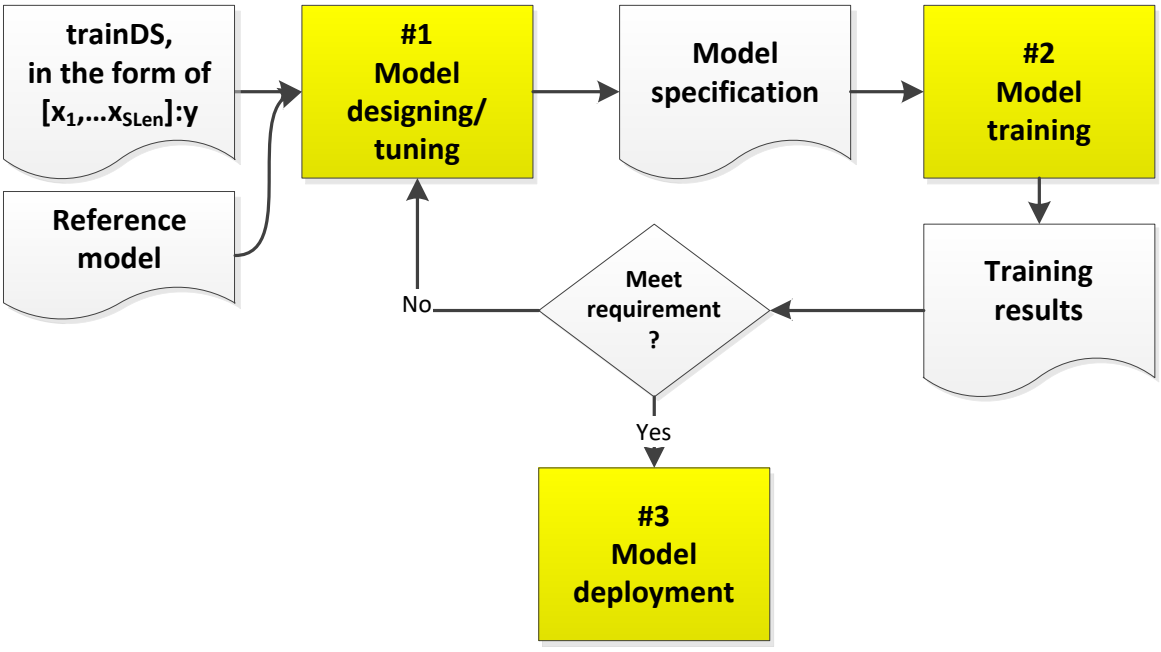

Fig. 5. Model development lifecycle flow

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

- Input layer with dimension (SLen, 1), with SLen = 3, which is the dimension of X, and 1 is the dimension of *y*.
- Hidden layers, for observation purposes, one to three LSTM layers are used in the experiments, where loss and accuracy are observed at each additional layer. Each LSTM layer is followed by a dropout layer, which removes cells that contribute to overfitting. The number of neurons is set to 256. The activation function applied is Tanh.
- Output layer, which uses the Dense layer after the LSTM layers. This layer is called the fully connected layer to the output, which in this case, the output dimension is the number of itemIDs as they are all potentially following items. The activation function used is Softmax.

LSTM is an RNN-type layer designed to handle time series data. LSTM has main components: cell, input gate, output gate and forget gate [31], [32]. Cells have the function of remembering past patterns in a series or sequence, 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 memory duration that the LSTM layer will remember is specified in the sequence or series length. LSTM can produce generative predictions, where the model can generate new samples from the same data distribution [36], [37]. For example, given a reading book as training data, a generative model for text-generator can generate several terms that will appear after a series of terms is given as a trigger so that composed sentences look new. To do this, the model requires the entire training data to be studied [34], [44].

The proposed model design is depicted in Figure 6, while a summary of the model using one layer of LSTM + Dropout + Dense is also given in Figure 7. A summary of models using two and three LTSM is not given, but intuitively it can be understood from such a figure. 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 a series of itemID in a whole trainDS. The design of this model is implemented with the Keras library using functional modeling. The layer

composition in each model is compiled by applying categorical cross-entropy loss, optimizer Adam. After compilation, the model is fitted to all vectors X and y, with 1000 epochs in 8 batches.

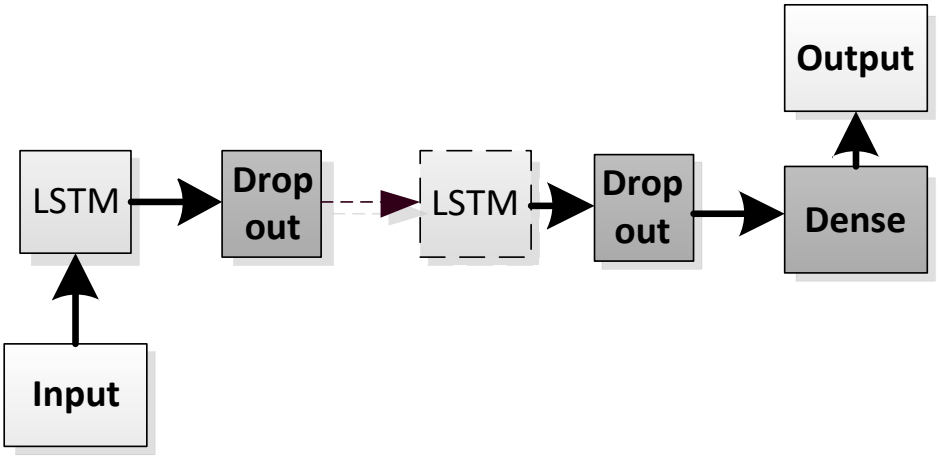

Fig. 6. Proposed model design

```
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. 7. Summary of model with one LSTM layer

D. Determining Top-K Recommendation

Briefly, the procedure carried out to generate next-items predictions is shown in Pseudocode 1.

PSEUDOCODE 1. LSTM and GRU Stack

```
1   Prepare the inputs:
    a) matrix Xs of all arrays X in trainDS in which X has a 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 arrangement of the layers 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, such as "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 shown in Pseudocode 2.

PSEUDOCODE 2. Generate next-items predictions

```
1. Determine the value of K;
2. Sort the probabilities in the PREDICTION array from the highest value,
   noting that each element 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. Benchmarking the Model

The activity flow of benchmarking the model is given in Figure 8, which shows that the proposed model is compared with the query-based session method. The aspect being compared is the ability of the model always to be able to get predictions of the probabilities of all itemIDs to become the next-item concerning the items that the user is looking for in a session. Two test scenarios were run to examine both methods in terms of their adaptability in generating next-item recommendations.

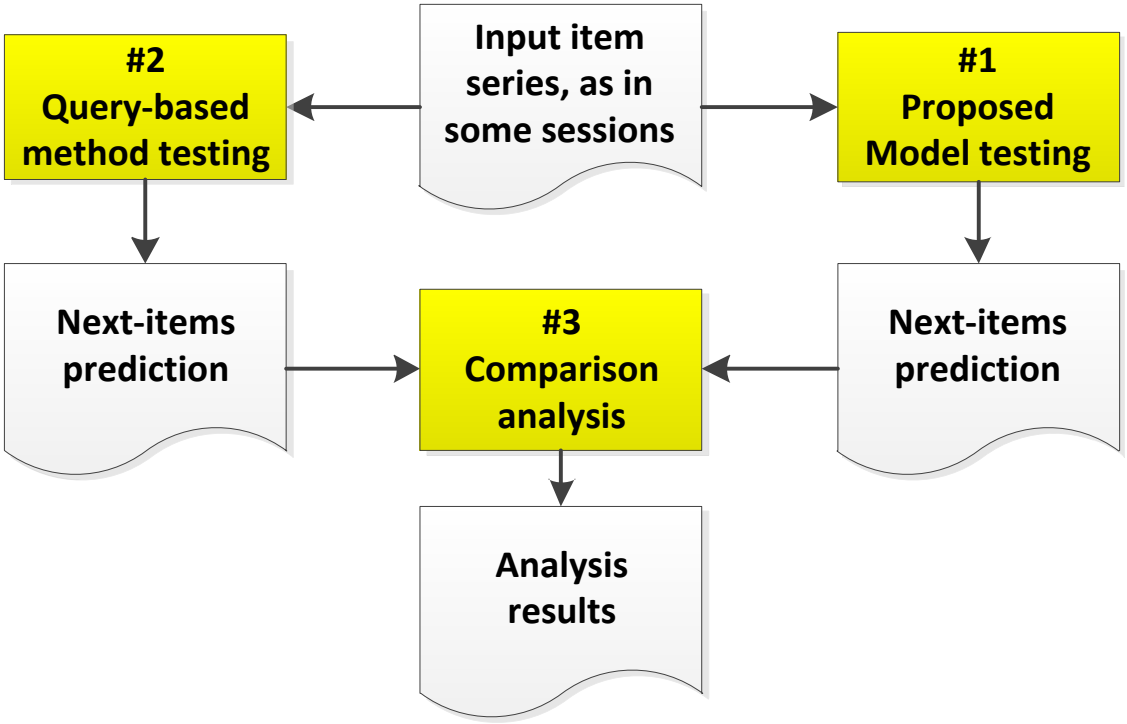

Fig. 8. Model benchmarking flow

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

1. Generate all rules X→Y with |X| = 3 and |Y| = 1,
2. Each X in the rules has at least one next-item Y
3. Enter all the Xs as the input for the proposed method and get the top-10 recommendations.
4. Count the number of X that have top-10 recommendations
5. 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.:

1. Simulate one 3-item series as input for the query-based method to find rules X→Y, where X is equal to the respective 3-item series

2. If the traditional query-based method cannot produce recommendations, then it is not an adaptive method.

F. Validating the Recommendation

The validity of the recommendation list can be confirmed through two approaches: system- or user-centered validity [45]–[50]. The recommendation results are matched with a set of items generated by the system, and here the validation results are objective. In the second approach, which is the one used in this study, recommendations are validated based on the user's perspective because in the end, users are expected to take action after seeing the contents of the recommendations. These perspective metrics include accuracy, familiarity, attractiveness, enjoyability, novelty, diversity, and context compatibility.

The number of 25 users were asked to evaluate seven metrics through one related question as follows [49], [50]:

1. Accuracy: the recommended items match my interests and vice versa
2. Familiarity: some recommended items are familiar to me and vice versa.
3. Attractiveness: some recommended item to me is attractive and vice versa
4. Enjoyability: I enjoy the items recommended and vice versa
5. Novelty: the RS helps me discover new items and vice versa
6. Diversity: the items recommended to me are varied and vice versa
7. Context compatibility: recommended items take into account my personal context and vice versa

For each question, users give a rating of 1 to 3, where 1 means the user strongly agrees with the question asked, 2 means a neutral perception, and 3 shows the user strongly disagrees. Users are uniformly asked to rate the ten 3-items in the generated rules and are assumed to be viewed by the user. The user validation table is described in Table 1.

Table 1. The user validation table

Description of items seen by user previously	**Illustration of item**							
Jumbo Bag Red Retro spot, Jumbo Bag Woodland Animals, Jumbo Storage Bag Suki								
Top-10 recommendation by proposed method		User's validation rate (1, 2, or 3) on seven Metrics						
Items recommended	Illustration	1	2	3	4	5	6	7
3 Birds Canvas Screen								
36 Doilies Vintage Christmas								
Advent Calendar Gingham Sack								
Antique Glass Heart Decoration								
Assorted Color T-Light Holder								

III. Results and Discussions

The dataset used is Online retail data available on the UCI web portal. The number of records initially was 541,909 lines, but after grouping by invoice number, the number of records became 22,106, consisting of 4059 unique items. As explained in the UCI web portal, this transnational dataset contains all transactions between 01/12/2010 and 09/12/2011 (almost one year) for UK-based and registered non-store online retailers. This company primarily sells unique gifts for any occasion.

In order to make the proposed method 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 2 million rules, which is not adequate for demonstrating the features and functionality of the proposed method and of the compared traditional method as well.

The difference between the proposed approach and traditional AR-based RS methods is that only rules with X and Y lengths of precisely one item were mined out, or $|X| = 1$ and $|Y| = 1$; whereas to the traditional method $0 < |X| \leq 3$ and $|Y| = 1$ were applied. These approaches are carried out with the following considerations: first, with short rules, the number of rules that must be maintained in memory is less than long rules [51], [52].

In the proposed approach, the number of rules generated is 194 rules which are then arranged as a series of rules that are used as the training dataset. The size of the 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.

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 with 1 LSTM layer is given in Figure 9.

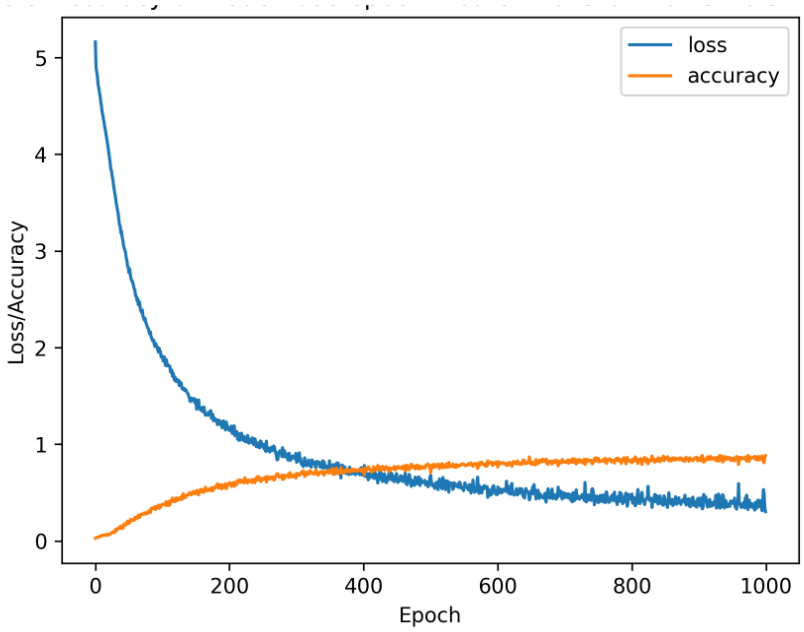

Fig. 9. Loss and accuracy of model with 1 LSTM + Dropout + Dense layers

An essential note during experiments is that if the dropout layer is not applied, 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 [39]. However, in some literature regarding text generators, no comparison was found between the results of the dropout and non-dropout models[31], [44], [53]. In addition, because of its generative nature, the text-generator

method results in the formation of new sentences from new term arrangements so that the 'accuracy' of terms that should appear after the previous term intuitively does not result from applying the dropout layer only, but also the richness of vocabulary and sentences available in the training set.

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 the next-items. In contrast, the query-based method cannot generate top-10 recommendations for all X, but only 2 items, as shown in Table 2 (left-side), is because not all X which is the antecedent of the rules has 10 consequent items Y. This is an advantage offered by the proposed method.

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

Table 2. Top-10 recommendations produced for items seen by the user, which is a frequent itemset (left-side), and is not a frequent itemset (right-side)

itemID	Description	itemID	Description
Items seen by user is frequent itemset		**Not a frequent itemset**	
85099B	Jumbo Bag Red Retrospot	85099B	Jumbo Bag Red Retrospot
20712	Jumbo Bag Woodland Animals	20711	Jumbo Bag Toys
21931	Jumbo Storage Bag Suki	20712	Jumbo Bag Woodland Animals
Top-10 Recommendation by proposed method			
84731	3 Birds Canvas Screen	22282	12 Egg House Painted Wood
22950	36 Doilies Vintage Christmas	84559B	3d Sheet of Cat Stickers
90199B	5 Strand Glass Necklace Amethyst	72801C	4 Rose Pink Dinner Candles
22580	Advent Calendar Gingham Sack	22371	Airline Bag Vintage Tokyo 78
21143	Antique Glass Heart Decoration	23068	Aluminum Stamped Heart
17164B	Ass Col Small Sand Gecko Weight	90183A	Amber Drop Earrings W Long Beads
47421	Assorted Color Lizard Suction Hook	84879	Assorted Color Bird Ornament
20749	Assorted Color Mini Cases	20749	Assorted Color Mini Cases
47420	Assorted Color Suction Cup Hook	47420	Assorted Color Suction Cup Hook
84950	Assorted Color T-Light Holder	84950	Assorted Color T-Light Holder
Top Recommendation by traditional method			
22386	Jumbo Bag Pink Polka Dot		
22411	Jumbo Shopper Vintage Red Paisley	No result	

Next, the proposed method's ability to generatively find recommendations for each input given in a session is demonstrated with the following step: 1) get the top-K recommendations from the itemIDs series, called X1, 2) the itemID in the first position of recommendation is assumed to be clicked by the user, so it goes into X1, and simultaneously pushes out a product from X1, 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, 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 Color 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 Colored Party Balloons, 22150: 3 Stripey Mice Felt craft, 84559a: 3D Sheet of Dog Stickers.

In this simulation, it can be understood that whatever order of items the user sees, the system can always generate a new list of recommendations, and with this ability, the recommendation system is said to be generative in generating recommendations.

The results of the user-centric validity test on the list of recommendations produced by the proposed model are shown in Figure 10, with the metrics being measured as accuracy, familiarity, attractiveness, enjoyability, novelty, diversity, and context compatibility, which are captured from the user's perspective. As seen, users feel that the recommended items are less accurate than those the user has seen. However, in other metrics, users give the opposite response. In terms of familiarity, even though it is inaccurate, as many as 56% of users feel familiar with the recommended item. Furthermore, as many as 72% of users agree that recommended items are attractive, 76% of users enjoy the list of recommended items, and they also feel that they just found out that the recommended items are related to items they have previously viewed. 80% of users agree that the list of recommended items is diverse, and 56% of users also agree that the items are related to the context of the items they have seen. On the other hand, although it appears that many users have a neutral opinion, it can be said that few users disagree with the questions asked regarding the metrics being measured.

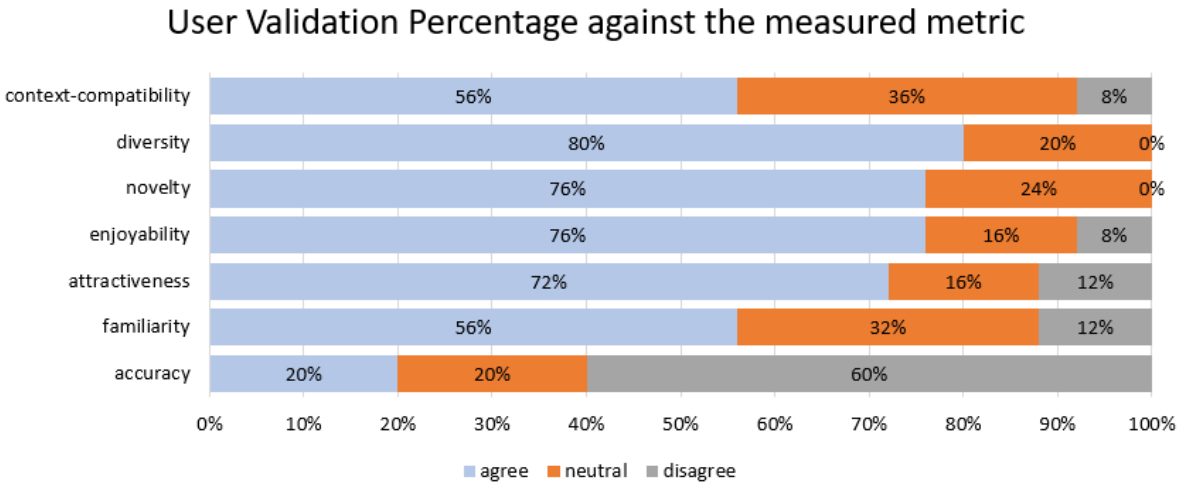

Fig. 10. User validation of measured metrics

An interesting thing to note is that 20% of users who have a neutral perception of accuracy think that the recommended product still has something to do with the product they have seen, namely that it has elements of animal shapes or something related to Christmas, such as the color red, and ornaments to decorate Christmas or New Year celebrations.

This result is in line with the results of previous studies, which show that accuracy versus novelty and diversity are inverse metrics [54]–[58]. If accuracy is essential, recommendation results tend to be uniform because accuracy is associated with the degree of similarity between the recommended product and those the user has seen or purchased. Diversity, on the other hand, brings a list of recommended products that are not similar to any products the user has ever seen. Novelty is closely related to diversity because the user's new understanding of the product usually arises when they are presented with products that are not similar to those previously visited.

Another important note is that AR-based RS does not produce recommended item Y with high similarity to a series of items X that the user has visited or purchased. The pair (X, Y) is formed from the Support and Confidence metrics, so if the results from the traditional method show that Y and X

look similar, it is because (X, Y) were purchased together, not because the product descriptions are similar.

IV. Conclusions

The ability of the proposed RNN-based session method to generatively and adaptively produce recommendations after recommendations from a series of items viewed by a user in a session has been demonstrated. Traditional query-based methods are incapable of 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. The results of user-centered validation of several matrices toward the proposed method show that although the level of accuracy of recommended products and products seen by users is only 20%, other metrics reach above 70%, such as novelty, diversity, attractiveness and enjoyability. As a suggestion for future development, the model can be built by adding several layers of attention to remember a more extended sequence of rules, such as in the transformers model.

Declarations

Author contribution

All authors contributed equally as the main contributor of this paper. All authors read and approved the final paper.

Funding statement

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Conflict of interest

The authors declare no known conflict of financial interest or personal relationships that could have appeared to influence the work reported in this paper.

Additional information

Reprints and permission information are available at http://journal2.um.ac.id/index.php/keds.

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.

[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.

[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.

[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.

[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.

[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.

[8] M. Ge, F. Ricci, and D. Massimo, "Health-aware food recommender system," in RecSys 2015 - Proceedings of the 9th ACM Conference on Recommender Systems, 2015.

[9] X. Li et al., "Application of intelligent recommendation techniques for consumers' food choices in restaurants," Front. Psychiatry, 2018.

[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.

[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.
[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.
[13] M. Ludewig and D. Jannach, "Evaluation of session-based recommendation algorithms," User Model. User-adapt. Interact., vol. 28, no. 4–5, 2018.
[14] S. Latifi, N. Mauro, and D. Jannach, "Session-aware recommendation: A surprising quest for the state-of-the-art," Inf. Sci. (Ny)., vol. 573, 2021.
[15] M. Ludewig, N. Mauro, S. Latifi, and D. Jannach, "Empirical analysis of session-based recommendation algorithms," User Model. User-adapt. Interact., vol. 31, no. 1, 2021.
[16] M. Maher et al., "Comprehensive Empirical Evaluation of Deep Learning Approaches for Session-Based Recommendation in E-Commerce," Entropy, vol. 24, no. 11, 2022.
[17] D. Wang, D. Xu, D. Yu, and G. Xu, "Time-aware sequence model for next-item recommendation," Appl. Intell., vol. 51, no. 2, 2021.
[18] G. M. Harshvardhan, M. K. Gourisaria, S. S. Rautaray, and M. Pandey, "UBMTR: Unsupervised Boltzmann machine-based time-aware recommendation system," J. King Saud Univ. - Comput. Inf. Sci., 2021.
[19] J. Li, Y. Wang, and J. McAuley, "Time interval aware self-attention for sequential recommendation," in WSDM 2020 - Proceedings of the 13th International Conference on Web Search and Data Mining, 2020.
[20] Y. Guo, Y. Ling, and H. Chen, "A time-aware graph neural network for session-based recommendation," IEEE Access, vol. 8, 2020.
[21] T. M. Phuong, T. C. Thanh, and N. X. Bach, "Neural session-aware recommendation," IEEE Access, vol. 7, 2019.
[22] J. Zhao et al., "DCFGAN: An adversarial deep reinforcement learning framework with improved negative sampling for session-based recommender systems," Inf. Sci. (Ny)., vol. 596, 2022.
[23] X. Huang, Y. He, B. Yan, and W. Zeng, "Fusing frequent sub-sequences in the session-based recommender system[Formula presented]," Expert Syst. Appl., vol. 206, 2022.
[24] L. van Maasakkers, D. Fok, and B. Donkers, "Next-basket prediction in a high-dimensional setting using gated recurrent units," Expert Syst. Appl., vol. 212, 2023.
[25] T. Liu, X. Yin, and W. Ni, "Next Basket Recommendation Model Based on Attribute-Aware Multi-Level Attention," IEEE Access, vol. 8, 2020.
[26] B. Peng, Z. Ren, S. Parthasarathy, and X. Ning, "M2: Mixed Models With Preferences, Popularities and Transitions for Next-Basket Recommendation," IEEE Trans. Knowl. Data Eng., vol. 35, no. 4, 2023.
[27] 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.
[28] 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.
[29] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa, "Effective Personalization Based on Association Rule Discovery from Web Usage Data," in Proceedings of the Third International Workshop on Web Information and Data Management (WIDM), 2001.
[30] 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.
[31] 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.
[32] R. Dolphin, "LSTM Neworks A Detailed Explanation," Web Page, 2020. (accessed Jan. 06, 2022).
[33] P. P. Barman and A. Boruah, "A RNN based approach for next word prediction in assamese phonetic transcription," in Procedia Computer Science, 2018.
[34] J. Brownlee, "Text Generation With LSTM Recurrent Neural Networks in Python with Keras," Machine learning mastery, 2018.
[35] J. Gordon and J. M. Hernández-Lobato, "Combining deep generative and discriminative models for Bayesian semi-supervised learning," Pattern Recognit., vol. 100, 2020.
[36] 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.
[37] 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.
[38] N. C. Dvornek, X. Li, J. Zhuang, and J. S. Duncan, "Jointly Discriminative and Generative Recurrent Neural Networks for Learning from fMRI," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2019.
[39] A. Carta, "Building a RNN Recommendation Engine with TensorFlow," Medium.com2, 2021. (accessed Jan. 06, 2022).
[40] S. Ambulgekar, S. Malewadikar, R. Garande, and B. Joshi, "Next Words Prediction Using Recurrent NeuralNetworks," ITM Web Conf., vol. 40, 2021.

[41] S. G. Vadlamudi, S. Kumar, S. Sahu, A. Malviya, and P. Choudhury, "Multi-Window Time-Aware Personalized Recommendation System," IN201741023671, 2017.
[42] Q. Zhang, L. Cao, C. Shi, and Z. Niu, "Neural Time-Aware Sequential Recommendation by Jointly Modeling Preference Dynamics and Explicit Feature Couplings," IEEE Trans. Neural Networks Learn. Syst., 2021.
[43] H. Miao, A. Li, L. S. Davis, and A. Deshpande, "Towards unified data and lifecycle management for deep learning," in Proceedings - International Conference on Data Engineering, 2017.
[44] 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.
[45] M. Kaminskas and D. Bridge, "Diversity, serendipity, novelty, and coverage: A survey and empirical analysis of beyond-Accuracy objectives in recommender systems," ACM Transactions on Interactive Intelligent Systems, vol. 7, no. 1. 2016.
[46] P. Gravino, B. Monechi, and V. Loreto, "Towards novelty-driven recommender systems," Comptes Rendus Physique, vol. 20, no. 4. 2019.
[47] K. Kapoor, V. Kumar, L. Terveen, J. A. Konstan, and P. Schrater, "'i like to explore sometimes': Adapting to dynamic user novelty preferences," in RecSys 2015 - Proceedings of the 9th ACM Conference on Recommender Systems, 2015.
[48] Z. Zolaktaf, R. Babanezhad, and R. Pottinger, "A generic top-n recommendation framework for trading-off accuracy, novelty, and coverage," Proc. - IEEE 34th Int. Conf. Data Eng. ICDE 2018, pp. 149–160, 2018.
[49] k. Nanath and M. Ahmed, "User-Centric Evaluation of Recommender Systems: A literature review," Int. J. Bus. Inf. Syst., vol. 1, no. 1, 2020.
[50] D. Y. A. Waykar, "Human-AI Collaboration in Explainable Recommender Systems: An Exploration of User-Centric Explanations and Evaluation Frameworks," INTERANTIONAL J. Sci. Res. Eng. Manag., vol. 07, no. 07, 2023.
[51] T. M. Akhriza and I. D. Mumpuni, "Quantitative class association rule-based approach to lecturer career promotion recommendation," Int. J. Inf. Decis. Sci., vol. 13, no. 2, 2021.
[52] T. M. Akhriza, Y. Ma, and J. Li, "Novel Push-Front Fibonacci Windows Model for Finding Emerging Patterns with Better Completeness and Accuracy:," ETRI J., vol. 40, no. 1, 2018.
[53] R. Patel and S. Patel, "Deep Learning for Natural Language Processing," in Lecture Notes in Networks and Systems, 2021, pp. 523–533.
[54] K. Bradley and B. Smyth, "Improving Recommendation Diversity," in Proceedings of the 12th irish conference on artificial intelligence and cognitive science, 2001.
[55] T. Yu, J. Guo, W. Li, H. J. Wang, and L. Fan, "Recommendation with diversity: An adaptive trust-aware model," Decis. Support Syst., vol. 123, 2019.
[56] R. Xie et al., "Improving Accuracy and Diversity in Matching of Recommendation With Diversified Preference Network," IEEE Trans. Big Data, vol. 8, no. 4, 2022.
[57] Y. Lin, C. Huang, W. Yao, and Y. Shao, "Personalised attraction recommendation for enhancing topic diversity and accuracy," J. Inf. Sci., vol. 49, no. 2, 2023.
[58] C. Matt, A. Benlian, T. Hess, and C. Weiß, "Escaping from the filter bubble? the effects of novelty and serendipity on users' evaluations of online recommendations," in 35th International Conference on Information Systems "Building a Better World Through Information Systems", ICIS 2014, 2014.