Intelligent Systems with Applications 14 (2022) 200070

Contents lists available at ScienceDirect

Intelligent Systems with Applications

journal homepage: www.elsevier.com/locate/iswa

Two-stage system using item features for next-item recommendation

Madiraju Srilakshmi*, Gourab Chowdhury, Sudeshna Sarkar

Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur, West Bengal, 721302, India

ARTICLE INFO

Article history:
Received 2 November 2021
Revised 4 February 2022
Accepted 3 March 2022
Available online 4 March 2022

Keywords:
Next-item recommendation
Cold-start problem
Item embedding

ABSTRACT

The next-item recommendation task aims to predict the item that a user is likely to interact with next, given the previous transactions of the user. In this paper, we explore the use of discrete item features to improve the performance of a next-item recommendation system. We design a hybrid embedding method based on user transactions and item features to represent items that can capture item-item co-occurrence as well as item features. We propose a two-stage model for the task of next-item recommendation. In the first stage, features of the next item are predicted, based on which candidate items are generated. In the second stage, the candidate items are ranked, and the top item is recommended to the users. Our model is capable of recommending new items since the candidate items are selected based on item features. We evaluate our model on three different datasets, including two public datasets and show that our model achieves significant improvement compared with several state-of-the-art next-item recommendation models.

1. Introduction

An e-commerce portal typically hosts a large number of items, and recommendation systems try to predict the user preferences over these items and provide a relevant set of items to the users.

The major sources of information for a recommendation system are user-item interaction records, item content and user profile.

1. The user-item interaction may be in the form of purchase, click, view etc. Some systems may have additional information such as user reviews or ratings for a given set of items.
2. The item content can be in the form of structured or unstructured data. Structured data includes discrete features like category, colour, brand, etc., and numeric features like the price of an item. The image of an item and review text written for an item are examples of unstructured content.
3. The user profile includes user demographic features such as age, gender, location and occupation.

The standard approaches for Recommendation Systems are collaborative filtering and content-based filtering. Collaborative filtering systems consider the interactions of the users to generate recommendations. However, these systems perform poorly when user-item interactions are very few. Collaborative filtering methods are also unable to handle new users and new items, a problem which is referred to as a cold-start issue. Content-based filtering methods recommend items to the users based on content such as item features and user profiles. Hybrid recommendation systems combine both collaborative filtering and content-based filtering methods to recommend the items.

We consider the task of next-item recommendation given user-item transactions and item content comprising of discrete features of items. Item features play an important role in the effectiveness of a recommendation system and are essential when user-item transactions are sparse, or items are completely new. Therefore, we wish to make use of item features along with user transaction history to learn an effective representation of items and leverage this representation to improve the quality of the recommendations that are provided to the users.

We generate embeddings from user-item transactions and item features separately and concatenate them to obtain the final embedding. We show that this embedding is effective when compared to the popular embedding methods such as Prod2Vec (Grbovic et al., 2015) and Meta-Prod2Vec (Vasile et al., 2016) when multiple item features are available.

A typical recommendation model for next-item prediction scores the available items in the catalogue for each user and recommends the top items (Hidasi et al., 2015). However, scoring all items in the catalogue for each user is challenging in real-world scenarios where the size of the catalogue is huge. This issue may be addressed using a two-stage system (Covington et al., 2016). In such a system the first stage is the candidate generation stage where a subset of items is selected as a candidate set. This reduces

* Corresponding author.
E-mail addresses: sreelakshmi@iitkgp.ac.in (M. Srilakshmi), gourab@iitkgp.ac.in (G. Chowdhury), sudeshna@cse.iitkgp.ac.in (S. Sarkar).

the total number of items to be scored in the second stage which is called the candidate ranking stage.

In this work, we propose an effective two-stage model for the task of next-item recommendation in the scenario where discrete item features are available. Our system predicts the features of the next item which is used as the basis of candidate generation. This approach also addresses the cold start problem of items. Since the candidate items are selected based on the predicted features, the items that are not seen before may also get selected.

1.1. Methodological contributions

The contributions of this study can be summarized as follows.

- We propose a hybrid embedding method to represent the items that can capture item-item co-occurrence as well as the features of items. We show that this embedding helps in improving the performance of a recommendation model when compared to the standard techniques such as Prod2Vec and Meta-Prod2Vec.
- We define a baseline method (one-stage model) that uses the above item representation for recommending the next item. We show that this approach is better and comparable to state-of-the-art systems.
- We propose an effective two-stage model for the task of next-item recommendation. We design a novel multi-tasking framework for candidate generation, which uses item features to generate the candidate items. Our approach can address the item cold-start problem which is very significant in recommendation systems.
- We estimate the computational complexity of our one-stage and two-stage systems in terms of trainable parameters and show that the two-stage model is less complex than the one-stage model.
- We experiment with three different datasets, i.e., Capillary Apparel dataset, Amazon Beauty review dataset (McAuley et al., 2015) and Behance dataset (He et al., 2016). We compare the results of the two-stage model with the one-stage model, several baseline methods and show that our two-stage model has outperformed them significantly.

The remainder of the paper is organized as follows. Section 2 describes the related work. Section 3 presents the studied problem and the datasets used for experiments. Section 4 presents the method of item embedding generation, and Section 5 presents the one-stage model, and the proposed two-stage model is described in Section 6. Section 7 presents the experiment results. Finally, Section 8 concludes the paper.

2. Related work

In this section, we first give an overview of different item embedding techniques. Then, we review next-item recommendation systems which use only user-item interactions for the recommendation. Finally, we give an overview of the methods which utilize item content along with user-item interactions for next-item prediction.

2.1. Item embedding methods

Item representation plays an important role in the effectiveness of a recommendation system. Many techniques have been proposed to generate good item representation by leveraging user-item transactions and item content information such as features, images, review text.

One of the popular techniques is Prod2Vec (Grbovic et al., 2015). Given a sequence of items interacted by the user, the objective of Prod2Vec is to find a low dimensional representation of items such that the items which co-occur in the user transactions are close in the resulting vector space. This method applies the skip-gram model on the user transaction sequences to obtain item embeddings.

Meta-Prod2Vec (Vasile et al., 2016) is an extension to Prod2Vec. In this method, item content in the form of discrete features is leveraged in addition to user transactions to generate the item embedding. The embeddings are learned by modelling each of the interactions between the items and the features separately.

Content2Vec (Nedelec et al., 2017) uses both structured and unstructured content of items to generate item embeddings. The items are represented by a unified embedding obtained from item-item co-occurrence information, item image and review text. The image embedding is generated by using pre-trained AlexNet and the product text is encoded using Word2Vec. The embedding based on item-item co-occurrence is obtained using Prod2Vec. These modules are combined by using a pairwise residual unit.

In contrast to Prod2Vec and Content2Vec, we propose a hybrid embedding technique that utilizes user transactions as well as discrete item features to obtain item representation. Our method is a simplified version of MetaProd2Vec. Unlike MetaProd2Vec, we learn feature embeddings and item embedding based on transactions separately and combine them to obtain the final representation. We show that this approach is more effective than MetaProd2Vec in our experiments.

2.2. Collaborative recommendation systems for next-item prediction

In this section, we review the methods which have used only user-item interactions for the task of next-item recommendation. Recurrent Neural Networks (RNN) and their variants (LSTM and GRU) have been widely applied for the task of next-item recommendation since they capture the temporal order among the user transactions. GRU4Rec (Hidasi et al., 2015) by Hidasi et al. was the first work that used RNN to model user transactions in a session. RNN is trained on the one-hot representation of item_ids in a user session and predicts the next item for new user sessions. Tan et al. (2016) improved this approach by adapting to the data augmentation technique, which splits the user session into many sub-sessions for training. GRU4Rec ++ (Hidasi and Karatzoglou, 2018) is an improved version of GRU4Rec which incorporates custom loss functions such as TOP1 loss, BPR loss. Gui and Xu (2018) used Prod2Vec embeddings (Grbovic et al., 2015) to represent items and utilize them with RNN based model.

Recent methods such as NARM (Li et al., 2017), STAMP (Liu et al., 2018) and SR-GNN (Wu et al., 2019) employ attention mechanisms to capture the main purpose of the user in the given user session. NARM (Li et al., 2017) uses a global encoder to encode the full user-session information using GRU and a local encoder to learn the user's main purpose in the session using GRU with attention. The output embeddings of both are combined to represent the user session. The item score is computed by using a bi-linear similarity function between learned session embedding and item embedding. STAMP (Liu et al., 2018) computes the item score as an inner product of item embedding and weighted user embedding. The user embedding is a bi-linear composition of overall preferences (average of all item embeddings in the user session) and current interest (last-clicked item embedding). The attention weights are learned using a simple feed-forward network on session sequences. SR-GNN (Wu et al., 2019) models session sequences as graph-structured data. Based on the session graph, Graph Neural Networks (GNN) learns item embedding, which can capture complex transitions of items. Each session is then represented as the combination of the overall preference and the current preference of that user session using an attention mechanism.

The major limitation of the above works is the inability to recommend new items. Also, these methods score every item in the catalogue to recommend the top items to the user, which is computationally expensive in real-world scenarios. In this study, we propose a two-stage system that addresses both these issues with improved performance by utilizing discrete item features.

2.3. Hybrid recommendation systems for next-item prediction

Various methods have been proposed to utilize item content information along with user transaction history to improve the quality of next-item recommendations. Parallel RNN (P-RNN) (Hidasi et al., 2016) utilized item content in the form of images and reviews to aid the user session modelling. P-RNN consists of multiple RNNs, one for each representation of the item (e.g. one for item_ID, one for text and one for image). The hidden states of these networks are combined to generate the score for all items. The bag-of-words model with tf-idf weights is used to encode the text, and Convolutional Neural Network (CNN) is used to encode the image. This work showed that incorporating item content in user behaviour modelling improves performance significantly.

Qian Zhao et al. proposed a categorical features-based item classification model (Zhao et al., 2018) for recommending items. They introduced an additional task of item feature prediction to enhance the actual item prediction through Multi-Task Learning (MTL) and Hierarchical Softmax (HSM). In MTL, the feature prediction and item prediction models share the same user state. They show that the feature prediction model benefits the item model learning through transfer learning. In HSM, the item model output is based on the feature model output. The probability of an item is calculated as the product of feature probability and the item probability within the set of items defined by the feature. The items are ranked based on their probability values obtained, and the top items are recommended.

This work is similar to our study in the context of utilizing discrete item features to improve the performance of recommender systems. However, this study is limited to dealing with one categorical feature only while our system is designed to handle multiple discrete item features.

Two-stage systems Covington et al. proposed a two-stage system comprising of candidate generation and ranking for recommending YouTube videos (Covington et al., 2016). They learn user representation by mapping the user's watch history and search tokens as well as demographic features such as age, gender into an embedding space. The candidate videos are obtained by performing a nearest neighbour lookup to the user embedding. A deep neural network that gives the score of each video using logistic regression is learned to rank the candidate videos.

Similar to this work, our study also designs a two-stage system to generate recommendations but with a special focus on leveraging discrete item features. We propose a hybrid embedding to represent items that capture user transaction-based similarity as well as item features. Also, we design a novel multi-tasking framework to predict the item features, which are then used to generate a candidate item set. Different from logistic regression to rank the items, we rank the candidate items based on the similarity between the user profile and candidate item.

3. Preliminaries

In this section, we first formally define the task. Then, we present the description of the datasets used in this work.

Table 1
Data statistics after pre-processing.

Statistics	Capillary	Amazon	Behance
# Total interactions	1,063,764	198,502	1,000,000
# Train sequences	649,714	137,967	888,944
# Test sequences	97,352	29,478	47,559
# Avg sequence length	4	8	19
# Total users	287,627	22,363	63,497
# Total items	47,896	12,101	178,788
# Total new items	1,499	326	8,277
# Total mentions of new items	4,317	6,149	8,633
# Total categories	5	233	-
# Total colours	98	-	-
# Total designs	27	-	-
# Total sleeves	3	-	-
# Total fit	23	-	-
# Total fabric	60	-	-
# Total brands	-	4,174	-
# Total owners	-	-	51,487

3.1. Problem definition

Next-item Recommendation is the task of predicting a user's next interaction based on his/her past interactions. In this work, we use the transaction records of all users and item content information in the form of discrete features of items.

Let $I = \{i_1, i_2, \ldots, i_m\}$ be the set of items and $U = \{u_1, u_2, \ldots, u_n\}$ be the set of users and $F = \{f_1, f_2, \ldots, f_k\}$ be the set of item features available. Each feature has a set of values and each item is associated with the value for the features. Let the interaction sequence of user u_i be $(x^i_1, x^i_2, \ldots, x^i_t)$, where x^i_t refers to the item interacted by the user at timestep t.

Thus, given all user interaction sequences, item features and a target user u_i with previous interaction sequence $(x^i_1, x^i_2, \ldots, x^i_t)$, the task is to predict the next possible interaction x^i_{t+1}.

3.2. Dataset description

To evaluate the proposed method, three datasets, namely, Capillary, Amazon and Behance were considered. The capillary dataset is anonymized apparel data provided by Capillary Technologies International Pte Limited. Amazon Beauty and Behance datasets consist of user interactions from various countries across the globe. Therefore, our experimental results are not constrained to any country setting. The details of each dataset are described below. We also present the pre-processing steps applied. The statistics of the data after pre-processing are given in Table 1.

3.2.1. Capillary apparel dataset

This dataset is anonymized and belongs to the apparel domain. The dataset comprises of 1-year data of user transaction records as well as item features. The user transaction data includes user_id, item_id and transaction_time. The item features are category, colour, fabric, design, fit, sleeve and size. All features are categorical. The first ten months data is used as a training set and the rest is used for the test. We remove users with only one transaction. We consider an item in the test set as new if it does not appear in the training set. Some items have multiple colour values. In such cases, we consider the primary colour only. For example, if the colour value is black with red, the primary colour is taken as black.

3.2.2. Amazon beauty review dataset

Amazon Beauty review dataset is a subset of Amazon review dataset (McAuley et al., 2015), which was introduced by McAuley et al. This dataset belongs to the beauty domain. We consider the user review as an interaction between the user and the item for which the review was written. The review text or rating value that is given by the user is not taken into account. We consider only discrete features of an item, i.e., brand and category. Each item is associated with a single brand and belongs to one or more categories. The user interactions (reviews) are sorted by time and the first 80% of the data is used as the training set and the remaining as the test set. The users with only one interaction are removed.

3.2.3. *Behance dataset*

Behance dataset (He et al., 2016) was collected and released by He et al. from Behance.net, which is a website where users can design projects and share them with other users. Users can appreciate projects created by others while browsing on the site. The task of project recommendation to the users can be formulated as predicting the probabilities of a user appreciating the projects in the system. Each project has an owner and some projects may have multiple owners. For each user, we consider the whole sequence of appreciated projects except the last one as a training set and the last project is used for the test. The users with only one appreciation are removed.

4. Item embedding generation

The first step in designing an effective recommendation model is to come up with good item representation. It has been shown that item embeddings play a key role in improving the quality of recommendations that are provided to the users (Grbovic et al., 2015; Nedelec et al., 2017; Vasile et al., 2016). In order to create an effective item embedding, we want to make use of item features along with user transaction history. While Prod2vec (Grbovic et al., 2015) generates item embeddings based on user transaction sequences only, Meta-Prod2Vec (Vasile et al., 2016) utilizes discrete item features along with user transaction sequences to obtain the item embeddings. We first present the details of Prod2Vec since we utilize this technique in our proposed embedding approach. We then describe Meta-Prod2Vec in Section 4.2. With respect to this, we present our proposed method in Section 4.3 which is a simple and effective method for incorporating multiple item features.

4.1. *Prod2Vec*

Given a sequence of user transactions, Prod2Vec (Grbovic et al., 2015) learns a D-dimensional representation of items such that the items that co-occur in user transactions are close in the resulting vector space. This technique applies the skip-gram model (Mikolov et al., 2013) on item sequences using a notion of an item sequence as a ̨sentenceg and items within the sequence as ̨wordsg. The embeddings are learned by minimizing the weighted cross-entropy between the empirical and the modelled conditional distributions of context items (C) given the target item (T). The size of the context window (c) determines how many items before and after a given target item are included as context items. In our experiments, c is set to 2.

The graphical representation is shown in Fig. 1. As shown in Fig. 1, the skip-gram model uses a fully connected neural network with a single hidden layer. The input layer is set to have as many neurons as the total number of items (m) since the one-hot encoding of a target item is given as input. The hidden layer size is set to the dimensionality of the resulting item embeddings (D). The weight matrix (W) between input and hidden layer is of dimension $m \times D$ and each row in this matrix represents an item from the item set (I). Similarly, the connections from the hidden layer to the output layer can be described by matrix (U) of size $D \times m$. In this case, each column of U represents an item from the given item set. The objective of Prod2Vec is to maximize the following

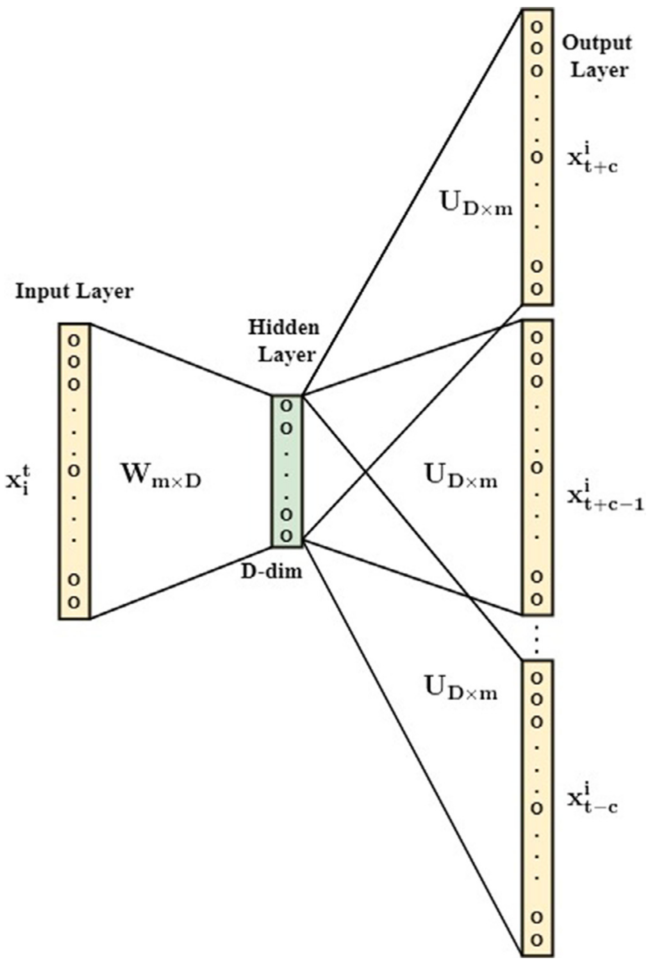

Fig. 1. Prod2Vec Architecture.

term (J) and the negative of this is considered as the loss function.

$$J = \sum_{d\in D_{tr}} \sum_{x_t^i \in d} \sum_{-c\le j\le c, j\ne 0} \log P(x_{t+j}^i | x_t^i) \tag{1}$$

$$L_{C|T} = -J \tag{2}$$

where D_{tr} is the set of item sequences and c is the context window size. The probability $P(x_{t+j}^i | x_t^i)$ of observing a context item x_{t+j}^i given the current item x_t^i is defined using the softmax function.

$$P(x_{t+j}^i | x_t^i) = \frac{exp(W_{x_t^i}^T U_{x_{t+j}^i})}{\sum_{p\in I} exp(W_{x_t^i}^T U_p)} \tag{3}$$

where $W_{x_t^i}$ and $U_{x_t^i}$ are latent vectors that correspond to the weight matrices W and U for the item x_t^i respectively and $W_{x_t^i}$ is used as final representation for the item x_t^i after training. I refers to the entire item set and x_t^i refers to the item interacted by user u_i at timestep t.

4.2. *Meta-Prod2Vec*

Meta-Prod2Vec learns the embeddings by modelling each of the interactions between the items and the features separately. The loss function (L_{MP2V}) of Meta-Prod2Vec is given below.

$$L_{MP2V} = L_{C|T} + \lambda\ (L_{f(C)|T} + L_{C|f(T)} + L_{T|f(T)} + L_{f(C)|f(T)}) \tag{4}$$

In the above expression, T and C refer to target items and context items. $f(C), f(T)$ refer to features of context items and features of target item respectively. λ is the regularization parameter.

The loss function (L) used is weighted cross-entropy. $L_{C|T}$ and $L_{f(C)|f(T)}$ encode the loss by modelling the conditional likelihood of item sequences and feature sequences. $L_{T|f(T)}$ is the weighted cross-entropy between the observed conditional probability of item_ids given their features and the predicted conditional probability. $L_{f(C)|T}$ and $L_{C|f(T)}$ represent the loss coming from modelling the conditional likelihood of context items' features given target item and context items given target item's features respectively.

In the original paper, the authors considered only a single item feature f. They mention that in case multiple features are available, each feature will have its own terms in the global loss function. Thus if $F = \{f_1, f_2, \dots, f_k\}$ is a set of item features, then the loss function (L_{MP2V}) of Meta-Prod2Vec given multiple features (F) is as follows:

$$L_{MP2V} = L_{C|T} + \lambda \sum_{i=1}^{k} (L_{f_i(C)|T} + L_{C|f_i(T)} + L_{T|f_i(T)} + L_{f_i(C)|f_i(T)}) \quad (5)$$

Thus for k features, the loss function has $4 * k + 1$ terms and $4 * k + 1$ types of training sequences are to be considered.

4.3. Proposed embedding

We propose a simplified mechanism that learns embeddings for item_ids and features separately and concatenates them to represent the items. Let D_{tr} be the set of item sequences. We apply the Prod2Vec technique on item sequences (D_{tr}) to generate embeddings for item_ids. This embedding is referred to as *item-sequences-based embedding*. The loss function (L_{tr}) to learn this embedding is given below.

$$L_{tr} = L_{C|T} \quad (6)$$

where T and C refer to target items and context items respectively.

For each feature $f_i \in F$ where F is the set of features, we compute feature sequences by replacing item_ids in item sequences with corresponding values of feature f_i. Prod2Vec is applied on each of these feature sequences separately to learn feature embeddings. The loss function (L_{f_i}) to learn embedding of feature f_i is given below.

$$L_{f_i} = L_{f_i(C)|f_i(T)} \quad (7)$$

where $f_i(C)$ and $f_i(T)$ refer to feature values of context items and feature values of target item corresponding to feature f_i respectively.

The embedding obtained by combining all feature embeddings is referred to as *feature-sequences-based embedding*. We concatenate *item-sequences-based* and *feature-sequences-based embeddings* to obtain the final representation (*Combined embedding)*. The overview of this process is shown in Fig. 2 and the steps are detailed in Algorithm.

In our experiments, we find that this formulation is more effective than Meta-Prod2Vec when more item features are present and comparable when fewer features are available. When there are k features, we learn $k + 1$ different embeddings. Each embedding is learned independently with a loss function containing a single term as in Eqs. (6) and (7). In contrast, Meta-Prod2Vec comes up with a single embedding to represent items that consider both item sequences and features and the loss function contains $4 * k + 1$ terms.

4.3.1. Features with multiple values

Some features may have multiple values for an item. To generate feature sequences for such features, we replace item_id with all available values of the feature. For example, consider an item sequence (i_1, i_2, i_3) in Amazon Beauty dataset. We want to generate a feature sequence corresponding to the item sequence for feature *category*. Let us assume that *category*(i_1) = (Makeup, Face), *category*(i_2) = (Makeup, Hair), *category*(i_3) = (Perfumes). The category sequence generated would be as follows : (Makeup, Face, Makeup, Hair, Perfumes). To obtain feature embedding for such features, we take the average of the embeddings of all feature values, and the mean embedding is used to represent the feature.

Algorithm 1: Steps to generate *Combined embedding*.

```
Input: $D_{tr}$: Set of user transaction sequences
       $F$: Item feature set
1 Apply Prod2Vec on $D_{tr}$.
  Item-sequences-based embedding = Prod2Vec($D_{tr}$)
2 for $f_i$ in $F$ do
3     Compute feature sequences ($D_{f_i}$) of feature $f_i$ by
      replacing each item in $D_{tr}$ with corresponding value
      of feature $f_i$.
4     Apply Prod2Vec on obtained feature sequences.
      $f_i$ embedding= Prod2Vec($D_{f_i}$)
5 end
6 Feature-sequences-based embedding=$||_{i=1}^{k} f_i$ embedding
7 Combined embedding = Item-sequences-based embedding
  || Feature-sequences-based embedding
```

5. One-stage model

We define a baseline method (One-stage model) for the task of next-item prediction, which uses the proposed embedding and scores all items in the vocabulary for each user and recommends the top item to the user. Most of the recent methods for next-item recommendation systems have used Recurrent Neural Networks (RNN) and its variants (GRU and LSTM) with attention mechanism to model the user transaction sequences. RNN is used to model the temporal ordering among the items and attention is applied to give weights to the items at different time steps. The items which are more relevant in the prediction are weighted more than other items in the input sequence. We adopt this framework to predict the next item directly. We use a Long Short Term Memory (LSTM) network to model user transaction sequences. A schematic diagram illustrating the one-stage model is shown in Fig. 3.

Let h_t be the LSTM hidden unit and y_t be the output at tth time step. For every sequence $(x_1^i, x_2^i, \dots, x_t^i) \in D_{tr}$ where D_{tr} is the set of user transaction sequences, we perform the following steps to train the one-stage model.

1. Each item in the sequence is represented with learned *Combined embedding.*

$$v_t^i = Combined\_embedding(x_t^i) \quad (8)$$

2. The hidden state h_t is updated by the previous hidden state h_{t-1} and the current item embedding v_t^i.

$$f_t = \sigma(W_f \times v_t^i + U_f \times h_{t-1} + b_f) \quad (9)$$

$$i_t = \sigma(W_i \times v_t^i + U_i \times h_{t-1} + b_i) \quad (10)$$

$$o_t = \sigma(W_o \times v_t^i + U_o \times h_{t-1} + b_o) \quad (11)$$

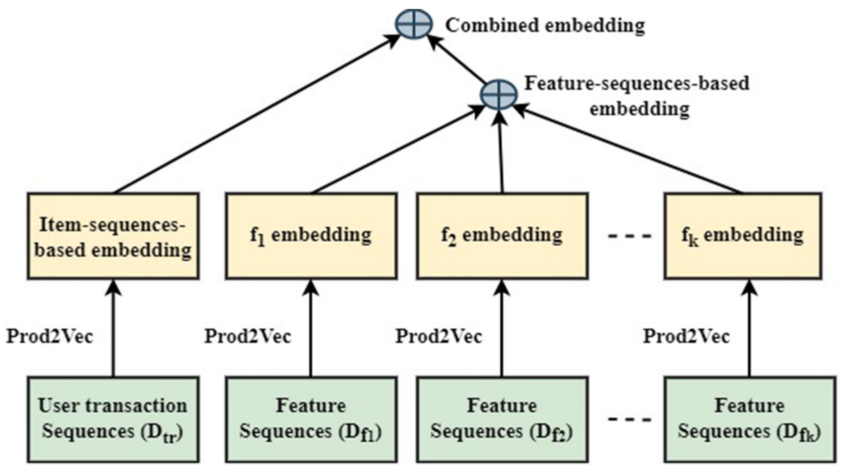

Fig. 2. Overview of item embedding generation.

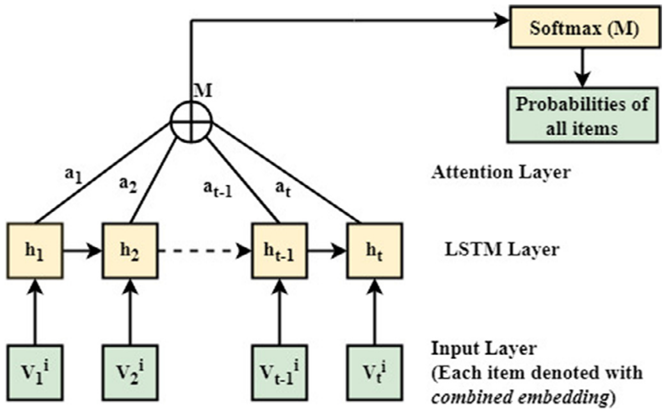
Softmax (M)
Probabilities of all items
M
Attention Layer
a1
a2
at-1
at
LSTM Layer
h1
h2
ht-1
ht
V1i
V2i
Vt-1i
Vti
Input Layer (Each item denoted with combined embedding)

Fig. 3. One-stage model for next-item recommendation .

$$\tilde{c_t} = \psi(W_c \times v_t^i + U_c \times h_{t-1} + b_c) \tag{12}$$

$$c_t = f_t \cdot c_{t-1} + o_t \cdot \tilde{c_t} \tag{13}$$

$$h_t = o_t \cdot \psi(c_t) \tag{14}$$

where, f_t, i_t and o_t are forget, input and output gates respectively. c_t denotes the cell state, σ and ψ refer to sigmoid and hyperbolic tangent functions respectively. W_f, W_i, W_o, W_c are input weights, U_f, U_i, U_o, U_c are recurrent weights and b_f, b_i, b_o, b_c are bias weights.

3. The output of hidden states $(H = h_1, h_2, \ldots, h_t)$ is given as input to attention layer to find attention weights and the weights for each time-step are learnt i.e. $A = (a_1, a_2, \ldots, a_t)$.

$$A = softmax(W_a \cdot \psi(H)) \tag{15}$$

4. The weighted sum of the hidden states (M) is given as input into a fully connected output layer (O). The number of neurons in this layer is equal to the number of unique items in the catalogue and softmax is used as the activation function.

$$M = A \cdot H \tag{16}$$

$$p_t \mid v^i_{1 \le i \le t-1} = O(M) \tag{17}$$

5. The loss function used for optimization is the categorical cross-entropy loss. The loss (L) is calculated as follows.

$$L = \sum_{c=1}^{m} y_c \log(p_c) \tag{18}$$

where m is the number of unique items, $Y \in \mathbb{R}^m$ is the one-hot represented ground truth and $P \in \mathbb{R}^m$ is the estimated probability for each class by softmax.

The trained LSTM with attention model is used for the next-item prediction. Each item in the target user's previous interaction sequence is represented by the *Combined embedding*. If any item in the sequence is not seen in the training set, we represent it with the *feature-sequences-based embedding* since the *item-sequences-based embedding* will be NULL. We input the sequence into the trained LSTM model. The outputs of the final layer in the model are the probabilities of all items in the catalogue. The item with the highest probability is recommended to the user.

5.1. Limitations

This method has certain limitations. The first one is the overhead to score all the items in the catalogue for each target user. This becomes quite expensive when the item set size is large, which is very often in real-world scenarios.

The second limitation is the inability to handle new items. This method cannot handle new items because the items are recommended purely based on the users' past transactions.

6. Two-stage model

We propose a two-stage model to overcome the limitations of the one-stage model. Scoring a large number of items in the catalogue can be avoided by first selecting a subset of the items as the candidate set for the target user and then recommending the top item by ranking this candidate set.

In our work, we propose to predict the next-item features and use them to select the candidate set of items. Since the total number of features and unique values in each feature is much less than the total number of items, predicting features takes less amount of

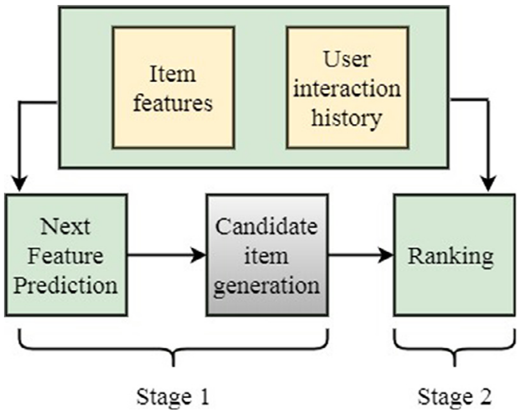

Fig. 4. Block diagram of two-stage model.

time than predicting items. For instance, the total number of items in the Capillary dataset is 47896, and the total number of features is 7. The maximum number of unique values for a feature in the entire feature set is 110, which is significantly smaller than the total number of items.

The size of the candidate items obtained may be much smaller than the size of the total itemset. For example, let us assume that the top features predicted for next-item are : *(category:shirt, design:solid, fabric:cotton)*. In the Capillary dataset, the total number of items is 47896, and the items with the above-mentioned features are 308. If we consider more item features, the size of the itemset can be even reduced further.

Importantly, the *candidate set* may also include *new items* which have not been encountered in the training set, since the items are selected based on predicted features. In this way, the two-stage model addresses the *cold-start problem* of items which is a major limitation of the one-stage model. For example, the predicted features for a target user are : *(category:t-shirt, colour:black)*. Now, all black t-shirts are our candidate items even if some of them are not seen by any user before. In this way, our method can address the cold start problem of items.

6.1. Overview

The overview of our proposed two-stage model is shown in Fig. 4. In the first stage, there are two modules, namely next-feature prediction and candidate item generation. In the next-feature prediction, we predict the features of the likely next item. Candidate item generation selects the candidate items using the predicted features. In the second stage, the candidate items are ranked and the top item is recommended to the user. The details of both the stages are given in the next sections.

6.2. Stage 1: Next-feature prediction

We use a Long Short Term Memory (LSTM) model with an attention mechanism for next-item features prediction. The LSTM is trained on user transaction sequences, and attention is applied on top of the LSTM. The same LSTM is used to predict all features of the next item. The model is trained jointly with k objectives when there are k features. The parameters are shared as it is done in multi-task learning. A schematic diagram illustrating the feature prediction model is shown in Fig. 5 and the detailed steps are given below.

1. Given user transaction sequences (D_{tr}), we compute the weighted sum of LSTM hidden states (M) as explained in one-stage model.
2. We input M into k fully connected output layers $(O^1, O^2, \dots, O^k)$ to find the probabilities of k features. The number of neurons in a dense layer O^k is equal to the number of unique values in feature f_k. The activation function is softmax for all output layers.
$$p_t^k \mid v^i_{1\le i\le t-1} = O^k[M] \tag{19}$$
3. The loss function used is categorical cross-entropy for all objectives. The total loss (L) is calculated as the sum of the individual losses $(L_1, L_2, \dots, L_k)$. The loss function (L_k) for a given feature f_k and total loss (L) is computed as follows.
$$L_k = \sum_{c=1}^{m_k} y_{o,c} \log(P_{o,c}) \tag{20}$$
$$L = \sum_{j=1}^{k} L_j \tag{21}$$
where m_k is the total values in feature f_k, $y_{o,c}$ is the binary indicator (0 or 1) if feature value c is the correct classification for observation o and $P_{o,c}$ is the probability of an observation belongs to feature value c.

The trained LSTM with attention model is used for next-feature prediction. The outputs of each of the final layers in the model are the probabilities of all feature values in the corresponding feature. The feature value with the highest probability is considered as the next possible feature value in that feature.

6.3. Stage 1: Item subset selection

We obtain the probabilities of all features from the feature prediction model. The items with the best feature value in each feature can be selected. However, the accuracy of the feature prediction model is low for the top-1 predicted value for some features. Therefore, we may consider the top l feature values for each feature, where $l > 1$.

There is a trade-off between the value of l and the size of the item subset. If we increase the value of l, then the size of the item subset increases, which is not desirable. Our main intuition is to reduce the number of items to score for each user. Therefore, we have to choose the value of l such that the size of the item set is comparatively small with satisfactory performance. In our experiments, the best results are obtained with l value 2, 5 and 20 for Capillary, Amazon and Behance datasets respectively. We take the top l ranked values of each feature and select the items with these feature values. The size of this item subset is much smaller than the full item vocabulary.

6.4. Stage 2: Item ranking

The candidate items selected in the first stage are ranked in the second stage. We find the similarity between each item in the candidate item set and the user profile. The items are ranked based on the similarity score. In our work, cosine similarity is used as the similarity measure. We experiment with different ways of computing the user profile. The user profile of a user u_i is referred to as s_i.

- UP-1: The first method is to consider the last item interacted by the user as the user profile.
$$s_i = v_t^i \tag{22}$$

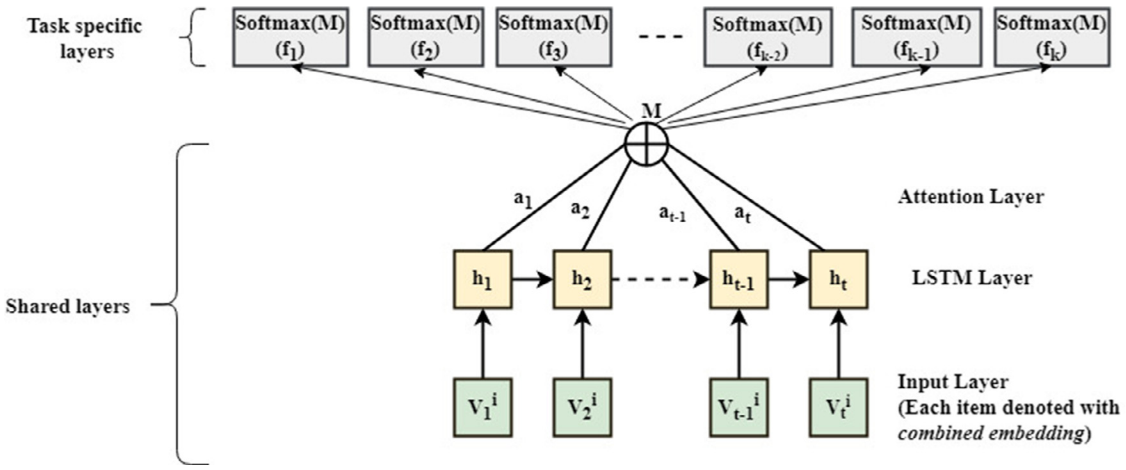

Fig. 5. Multi-task learning for feature prediction using LSTM with attention.

- UP-2: The second method is to take the average of all item embeddings that are interacted by the user (till current timestep t).

$$s_i = Avg(v_1^i, v_2^i, \dots, v_t^i) \quad (23)$$

The cosine similarity between user profile (s_i) and item embedding (v_t^i) is computed as follows.

$$similarity(s_i, v_t^i) = s_i.v_t^i/(|s_i||v_t^i|) \quad (24)$$

6.5. Computational complexity

The computational complexity of a neural network can be estimated based on the total number of trainable parameters in the network. The one-stage system is a LSTM network whereas the two-stage system uses a LSTM network for the feature prediction model. The items with the predicted features are selected as candidate items and a simple cosine similarity score is used for candidate ranking. Therefore, we consider the number of trainable parameters in both models to compare the computational complexity.

The input size, number of layers and number of hidden units in each layer are the same for both the one-stage system and feature prediction model. The only difference is the number of neurons in the final layer. Since the one-stage model scores all items to predict the next item, the number of neurons in the final layer is m, where m is the total number of items in the catalogue. The feature prediction model adopts a multi-task learning framework to predict the features. Therefore, the number of neurons in the final layer is $\sum_k m_k$, where k is the total number of features and m_k is the total number of values in each feature.

Since $m > \sum_k m_k$ in the real world scenario, the feature prediction model is very less expensive than the one-stage system. Thus, the overall two-stage system is less complex than the one-stage system.

7. Experimental results

In this section, we first describe the evaluation metrics employed in our experiments and the details of the experimental setup. Then we present the results of the proposed framework.

7.1. Evaluation metrics

We use the following metrics to evaluate our model.

- Hit Rate (HR) @k: It is the proportion of cases where the actual item is among the top-k recommended items among all test cases. We verify our results for $k \in (1, 10, 20)$.
- Mean Average Precision (MAP) @20: It is the mean of average precision of all test sequences. Since there is only one target item, the average precision is the same as reciprocal rank.

7.2. Experimental setup

In this section, we report the parameters which have given the best results in our experiments.

We have taken the size of *item-sequences-based embedding* to be 30, 50 and 50 for Capillary, Amazon and Behance datasets respectively. Each *feature embedding* is of size 20, 50 and 50, thus *Combined embedding* for these datasets results in the size of 170, 150 and 100 respectively. We have taken the same sizes i.e., 170, 150 and 100 for Meta-Prod2Vec embedding for a fair comparison.

We use Adam optimizer, and the learning rate is set to 0.001. The batch size is set at 512. There is a dropout layer in between the LSTM layer and the attention layer with 25% dropout. We use one LSTM layer with 100 hidden units. We fix the length of the input sequence in LSTM as 10, 10 and 20 for Capillary, Amazon and Behance datasets respectively. These values are chosen based on the average sequence length. The above specifications are used for both the one-stage model and the feature prediction model.

Our one-stage and two-stage models are implemented in Python (v3.6), Keras (v2.2.4) (Chollet et al., 2015) and Tensorflow (v1.14.0) (Abadi et al., 2015) for the back-end for Keras. We use Gensim (v3.8.0) (Řehůřek and Sojka, 2010) to implement Prod2Vec embedding on transaction sequences and feature sequences. We also use Gensim to find similar items for the user profile.

7.3. Results and analysis

In this section, we present our results on the three datasets. We first compare the results of the one-stage model with the other baseline methods. Then, we present the results of the feature prediction model which are intermediate results of the two-stage model. Finally, we present the results of the two-stage model.

7.3.1. Performance comparison of one-stage model with baselines

We compare the result of our one-stage model with recent next-item prediction methods i.e., GRU4Rec++ (Hidasi and Karatzoglou, 2018), NARM (Li et al., 2017) and SR-GNN (Wu et al., 2019). These methods recommend items purely based on user past interactions. The details of these methods are presented in Section 2.2.

We show the results of one-stage model with *item-sequences-based embedding (IS embedding)*, *feature-sequences-based embedding (FS embedding)* and *Combined embedding*. The results are listed in Table 2.

Table 2
Performance comparison of one-stage model with baselines (The results are in %).

Method	Capillary HR@20	Amazon HR@20	Behance HR@20
GRU4Rec++	2.59	2.38	2.81
NARM	2.89	2.69	2.87
SR-GNN	4.84	5.26	3.19
1-stage (IS embedding)	5.25	5.52	3.22
1-stage (FS embedding)	2.60	6.39	3.60
1-stage (Combined embedding)	**6.60**	**7.16**	**7.59**

Table 3
Results of feature prediction model on Capillary data (results in %).

Feature	HR@1	HR@4	HR@5
Category	83.25	96.30	100.00
Colour	19.80	25.50	36.50
Fabric	35.60	59.20	87.80
Fit	28.90	47.50	82.55
Size	41.80	63.07	90.15
Sleeve	81.06	99.11	100.00
Design	35.51	55.03	83.35

Table 4
Results of feature prediction model on Amazon data (results in %).

Feature	HR@1	HR@4	HR@5
Category	53.45	84.23	93.09
Brand	30.92	52.20	59.32

Table 5
Results of feature prediction model on Behance data (results in %).

Feature	HR@1	HR@4	HR@5
Owner	1.32	5.36	8.28

From the results, it is evident that the performance of the one-stage model with *IS embedding* is comparable with the baseline methods. The one-stage model with *Combined embedding* achieved even better performance. Incorporating item features in the embedding helps to improve the performance. The results reveal that the one-stage model is eligible as a strong baseline for the task of next-item prediction.

7.3.2. Results of next-feature prediction model

The next-feature prediction model is an intermediate module in the two-stage system. The results of this model on Capillary, Amazon and Behance datasets are listed in Tables 3, 4 and 5. We report the results in hitrate@{1,4,5}.

In case any feature has multiple values and the model predicts any one of the feature values correctly, we consider it as a correct prediction. From the results, we observe that the features with a fewer number of values are predicted more effectively than the features with a comparatively large number of values.

7.3.3. Performance comparison with MTL and HSM

HSM and MTL are recent methods that utilized item discrete features along with user previous interactions to predict the next item. The details of these methods are presented in Section 2.3. We compare the results of the one-stage, two-stage models with HSM and MTL on Behance dataset only. Since these methods are designed only for a single categorical feature, we cannot directly apply them to Capillary and Amazon datasets. The results of these methods are taken from the original paper (Zhao et al., 2018). Since the evaluation metric used in the original work is MAP@20, we report MAP@20 values of the two-stage and one-stage model in MAP@20. We show the results with *FS embedding* since we obtain the best value with this embedding. The results are listed in Table 6.

Table 6
Performance comparison of 2-stage, one-stage model with HSM and MTL (results in %).

Method	MAP@20
HSM	1.29
MTL	1.59
1-stage (FS embedding)	1.05
2-stage (FS embedding)	**9.91**

The results reveal that our two-stage model outperformed HSM and MTL significantly. The improvement could be due to the ability of our system to recommend new items.

7.3.4. Performance comparison of the two-stage model with one-stage model

We compare our two-stage model result with the one-stage model. We show the result of the two-stage model with different user profiles. The results are listed in Table 7. For each dataset, we report the results obtained with the best embedding corresponding to that dataset. A detailed comparison of item embeddings is given in Section 7.3.6.

From the results, it is evident that the performance of the two-stage model is significantly better than the one-stage model. The model with UP-2 has given the best result on Capillary and Amazon datasets. This method considers all items that the target user has already seen and takes the average to represent the user. The model with UP-1 has given the best result on the Behance dataset and a comparable result on Capillary and Amazon datasets. In this case, only the last item is considered to represent the user. From this result, it is apparent that the user's next transaction mostly depends on the previous one. The best user profile corresponding to each dataset was considered for further experiments on that dataset.

7.3.5. Cold-start of items

To verify the efficacy of our two-stage model in handling new items, we selected the test sequences with only new items as a target. We compare the performance of one-stage and two-stage models on this test data. We also report the performance of one-stage and two-stage models on old items which are already seen in training data. The results are listed in Table 8.

From the results, it is evident that the one-stage model is unable to recommend new items. In the two-stage model, the candidate items are selected based on the item features which helps to consider new items also as candidate items.

7.3.6. Results of one-stage and two-stage models with different item embeddings

We also compare the result of our one-stage and two-stage models with different item representations i.e., *item-sequences-based embedding (IS embedding)*, *feature-sequences-based embedding (FS embedding)*, Meta-Prod2Vec and *Combined embedding*. The *item-sequences-based embedding* is same as the standard Prod2Vec embedding. The results are listed in Table 9.

We observe that the *Combined embedding* performs better than only *item-sequences-based* or *feature-sequences-based embedding* on both Capillary and Amazon data for one-stage and two-stage models. For Behance data, one-stage and two-stage models performed better with *feature-sequence-based embedding* than other embedding methods. Since Behance data is about project design and appreciation, it seems convincing that a user appreciating a project depends more on the owner of the project rather than the projects that were previously appreciated by the user.

Table 7
Performance comparison of 1-stage and 2-stage models (The results are in %).

Method	Capillary			Amazon			Behance		
	HR@1	HR@10	HR @20	HR @1	HR @10	HR @20	HR @1	HR @10	HR @20
1-stage	0.47	3.81	6.60	0.68	4.49	7.16	0.84	3.09	7.59
2-stage+UP-1	0.40	4.28	7.16	0.62	3.81	6.19	5.82	**19.26**	**22.20**
2-stage+UP-2	**1.09**	**5.81**	**8.58**	**1.09**	**5.60**	**8.02**	**6.55**	13.92	15.93

Table 8
Performance comparison of 1-stage and 2-stage models with new and old targets (The results are in %).

Method	Targets	Capillary			Amazon			Behance		
		HR @1	HR @10	HR @20	HR @1	HR @10	HR @20	HR @1	HR @10	HR @20
1-stage	new	0	0	0	0	0	0	0	0	0
2-stage	**new**	**0.50**	**2.70**	**3.72**	**0.54**	**4.47**	**6.91**	**5.84**	**16.11**	**18.99**
1-stage	old	0.45	3.35	5.71	0.86	5.72	**9.06**	0.10	0.57	2.30
2-stage	**old**	**1.11**	**5.95**	**8.80**	**1.15**	**5.84**	8.50	**2.26**	**7.11**	**8.82**

Table 9
Performance comparison of 1-stage and 2-stage models with different embeddings (The results are in %).

Method	Embedding	Capillary			Amazon			Behance		
		HR @1	HR @10	HR @20	HR @1	HR @10	HR @20	HR @1	HR @10	HR @20
1-stage	IS embedding	0.39	3.06	5.25	0.48	3.33	5.52	0.14	0.96	3.22
1-stage	Meta-prod2vec	0.37	2.66	4.58	0.74	4.49	7.06	0.89	4.21	10.21
1-stage	FS embedding	0.23	1.48	2.60	0.55	4.14	6.39	0.27	1.17	3.60
1-stage	Combined embedding	0.47	3.81	6.60	0.68	4.49	7.16	0.84	3.09	7.59
2-stage	IS embedding	0.44	2.74	4.29	0.26	1.83	2.91	0.26	1.10	1.46
2-stage	Meta-prod2vec	0.61	3.47	5.26	0.76	5.08	**8.11**	1.16	10.52	13.60
2-stage	FS embedding	0.16	1.18	2.03	0.62	3.81	6.19	**5.82**	**19.26**	**22.20**
2-stage	Combined embedding	**1.09**	**5.81**	**8.58**	**1.09**	**5.60**	8.02	3.52	10.33	12.27

The Meta-Prod2Vec embedding has shown comparable performance with *Combined embedding* on Behance and Amazon datasets. However, the performance of Meta-Prod2Vec is poor when compared with *Combined embedding* on the Capillary dataset. Since the Capillary dataset has more features, Meta-Prod2Vec embedding may not be able to capture the dependencies between the items and all these features.

8. Conclusion

In this paper, we have presented an item embedding method that can capture both item-item co-occurrence as well as item features. This method is found to be especially effective when multiple item features are available. We propose a two-stage system for the task of next-item recommendation given discrete item features. Since the candidate items are generated by predicting the features of the next item, our method is able to address the cold-start problem of items. We demonstrate through experiments on three different datasets that the two-stage model outperforms the one-stage model and other baseline systems significantly. We also show that the two-stage model is better than the one-stage model in recommending both old and new items.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This research was partially supported by a grant from Capillary Technologies International Pte Limited. We thank Dr. Subrat Panda and Mr. Biswa Singh who provided insight and expertise that greatly assisted the research.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., & Zheng, X. (2015). TensorFlow: large-scale machine learning on heterogeneous systems. Software available from https://www.tensorflow.org/.

Chollet, F. et al. (2015). Keras. https://keras.io.

Covington, P., Adams, J., & Sargin, E. (2016). Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems* (pp. 191–198).

Grbovic, M., Radosavljevic, V., Djuric, N., Bhamidipati, N., Savla, J., Bhagwan, V., & Sharp, D. (2015). E-commerce in your inbox: Product recommendations at scale. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1809–1818). ACM.

Gui, Y., & Xu, Z. (2018). Training recurrent neural network on distributed representation space for session-based recommendation. In *2018 International joint conference on neural networks (IJCNN)* (pp. 1–6). IEEE.

He, R., Fang, C., Wang, Z., & McAuley, J. (2016). Behance dataset. http://cseweb.ucsd.edu/~jmcauley/datasets.html#behance.

He, R., Fang, C., Wang, Z., & McAuley, J. (2016). Vista: A visually, socially, and temporally-aware model for artistic recommendation. In *Proceedings of the 10th ACM conference on recommender systems* (pp. 309–316). ACM.

Hidasi, B., & Karatzoglou, A. (2018). Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM international conference on information and knowledge management* (pp. 843–852). ACM.

Hidasi, B., Karatzoglou, A., Baltrunas, L., & Tikk, D. (2015). Session-based recommendations with recurrent neural networks. arXiv preprint arXiv:1511.06939.

Hidasi, B., Quadrana, M., Karatzoglou, A., & Tikk, D. (2016). Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *Proceedings of the 10th ACM conference on recommender systems* (pp. 241–248). ACM.

Li, J., Ren, P., Chen, Z., Ren, Z., Lian, T., & Ma, J. (2017). Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on conference on information and knowledge management* (pp. 1419–1428). ACM.

Liu, Q., Zeng, Y., Mokhosi, R., & Zhang, H. (2018). Stamp: short-term attention/memory priority model for session-based recommendation. In *Proceedings* *of the 24th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 1831–1839). ACM.

McAuley, J., Targett, C., Shi, Q., & Van Den Hengel, A. (2015). Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval* (pp. 43–52).

McAuley, J., Targett, C., Shi, Q., & Van Den Hengel, A. (2015). *Amazon dataset*. http://jmcauley.ucsd.edu/data/amazon/links.html.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111–3119).

Nedelec, T., Smirnova, E., & Vasile, F. (2017). Specializing joint representations for the task of product recommendation. In *Proceedings of the 2nd workshop on deep learning for recommender systems* (pp. 10–18).

Řehůřek, R., & Sojka, P. (2010). Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 workshop on new challenges for NLP frameworks* (pp. 45–50). Valletta, Malta: ELRA. http://is.muni.cz/publication/884893/en

Tan, Y. K., Xu, X., & Liu, Y. (2016). Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st workshop on deep learning for recommender systems* (pp. 17–22). ACM.

Vasile, F., Smirnova, E., & Conneau, A. (2016). Meta-Prod2Vec: Product embeddings using side-information for recommendation. In *Proceedings of the 10th ACM conference on recommender systems* (pp. 225–232). ACM.

Wu, S., Tang, Y., Zhu, Y., Wang, L., Xie, X., & Tan, T. (2019). Session-based recommendation with graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence: vol. 33* (pp. 346–353).

Zhao, Q., Chen, J., Chen, M., Jain, S., Beutel, A., Belletti, F., & Chi, E. H. (2018). Categorical-attributes-based item classification for recommender systems. In *Proceedings of the 12th ACM conference on recommender systems* (pp. 320–328). ACM.