3D Convolutional Networks for Session-based Recommendation with Content Features

Trinh Xuan Tuan
NextSmarty R&D
Hanoi, Vietnam
tuantx@nextsmarty.com

Tu Minh Phuong*
Department of Computer Science
Posts and Telecommunications Institute of Technology
Hanoi, Vietnam
phuongtm@ptit.edu.vn

ABSTRACT

In many real-life recommendation settings, user profiles and past activities are not available. The recommender system should make predictions based on session data, e.g. session clicks and descriptions of clicked items. Conventional recommendation approaches, which rely on past user-item interaction data, cannot deliver accurate results in these situations. In this paper, we describe a method that combines session clicks and content features such as item descriptions and item categories to generate recommendations. To model these data, which are usually of different types and nature, we use 3-dimensional convolutional neural networks with character-level encoding of all input data. While 3D architectures provide a natural way to capture spatio-temporal patterns, character-level networks allow modeling different data types using their raw textual representation, thus reducing feature engineering effort. We applied the proposed method to predict add-to-cart events in e-commerce websites, which is more difficult then predicting next clicks. On two real datasets, our method outperformed several baselines and a state-of-the-art method based on recurrent neural networks.

CCS CONCEPTS

• **Information systems → Recommender systems**;

KEYWORDS

Convolutional neural networks; session-based recommendation; recommender systems

ACM Reference format:
Trinh Xuan Tuan and Tu Minh Phuong. 2017. 3D Convolutional Networks for Session-based Recommendation with Content Features. In *Proceedings of RecSys'17, August 27–31, 2017, Como, Italy, ,* 9 pages.
DOI: http://dx.doi.org/10.1145/3109859.3109900

1 INTRODUCTION

In order to generate personalized recommendations, traditional recommender systems rely on user profiles created from purchase histories, explicit ratings or other kinds of past interactions such as views, comments etc. For a given user, collaborative filtering (CF) approaches make predictions based on users with similar profiles [7] or by computing hidden factors of users and items with matrix factorization methods [15]. On the other hand, content based approaches recommend items based on their similarity to those present in the profile [2]. In either case, informative user profiles are assumed to be available. Unfortunately, this is often not the case in real-life applications. To create a profile, the user must be identified across sessions, for example by authentication. In practice, however, many e-commerce websites allow users to browse and even make purchases without authentication. Although there are other methods for user identification, e.g. by using cookies or fingerprinting techniques, the applicability of those methods is limited because of their relatively low reliability and privacy concerns. In addition, the creation of an informative profile requires the user to have enough interactions with the system in the past. This is often not satisfied, as many retail websites have small percentage of returning users, i.e. most users have only one visit or purchase.

When user profiles are not available, a typical solution is to base recommendations on session data, e.g. session clicks. These data have two important characteristics. First, session clicks are sequential in nature and the order of clicks may contain information of user intent. Second, clicked items are often associated with metadata such as names, categories, and descriptions, which provide additional information about user taste. In the absence of user profiles, it is important to exploit these two characteristics to draw more information from data. Therefore, a session-based recommendation method should have the following desired properties:

- The method should be able to model sequential patterns in streams of clicks. Previously, a popular method for session-based recommendation is item-to-item kNN. This method makes recommendation based only on item co-occurrences, completely ignoring click orders. There are other methods that use transition probabilities but they consider only the last click, ignoring information from past clicks. Recent findings show that explicitly taking into account the sequential nature of clicks and considering all past events result in improved recommendations [6, 11].
- The method should provide a simple way to represent and combine item IDs with metadata. Usually an item is associated with features of different types. For example, a product may have a numerical product ID, textual name/description, and it may belong to one or more categories from some category hierarchy. Most existing methods perform feature extraction and selection for each type of feature independently, or even construct an independent model for each feature

*Also with FPT Software Research Lab.

type and then combine their outputs [12]. This requires a lot of effort for feature engineering and may fail to capture interactions between features. It would be more convenient to have a general way to represent different feature types and jointly model their interactions.

In this work, we propose a session-based recommendation method with these two desired properties. The method uses a 3D convolutional neural network (3D CNN) [13, 24] to capture sequential patterns in streams of clicks and associated features. It considers item IDs and all content features including hierarchical categories as texts and represent the resulting textual data with a character-level model [28]. A deep 3D CNN provides a natural way to jointly model temporal and content patterns that are indicative of purchasing intentions. At the same time, representing all features at character level frees us from the need of time consuming feature engineering and can be applied for different types of features. We empirically show that character-level encoding, combined with 3D CNN, can deliver competitive recommendation accuracy. In addition, unlike previous works that use 1-hot encoded vector for numerical item-IDs and words [12], character-level encoding results in much more compact representation for input. And, unlike other works that learn item embeddings [23], our model does not require any embedding layer. Given that both 1-hot representation of words and item embeddings results in large numbers of parameters, our model has substantially fewer parameters than previous deep neural network models for session-based recommendation.

Most existing methods for session-based recommendation predict the next click and present to the user a short list of top predictions. While predicting the next click is appropriate for news and media sites, it is not enough for retail websites. In the latter case, it is more useful to predict and recommend the list of products the user intents to buy. This will reduce browsing time, focus the user on potential products, and thus increase the purchase probability. However, directly predicting purchased items is more difficult than predicting the next click and requires appropriate organization of training data from session logs. Here we show that the proposed 3D CNN, if trained appropriately, is able to predict add-to-cart items with relatively high accuracy. Specifically, when user clicks on some item, this click and all the previous clicks of the current session (if any) are given as input, from which the model will predict a short list of items the user is likely to add to cart and present them as a recommendation.

We experimentally evaluated the proposed method on product recommendation in two retail websites, where each product is associated with textual descriptions and categories. We compared our method with several baselines and a leading session-based recommendation method using recurrent neural networks. The results show that our method achieved superior performance, while requiring less feature engineering steps.

In summary, our contributions are as follows:

- We propose a 3D CNN model for session-based recommendation, which allows jointly modeling sequential pattern of session clicks and different content features of items. The model can predict add-to-cart items based on the past clicks of the current session.
- We propose using character-level representation for all types of features, which frees us from feature engineering steps and reduces the number of model parameters. Experiments demonstrate the effectiveness of the proposed method.

2 RELATED WORK

Collaborative and content-based filtering

Collaborative filtering (CF) and content-based filtering (CBF) are two main groups of approaches used in recommender systems. The most popular CF algorithms are matrix factorization and nearest neighbor methods. Given a matrix of user-item ratings, matrix factorization [15] finds vectors of hidden factors for each user and each item so that, for each user-item pair, the inner product of their vectors closely approximates their original rating value, if this rating exists. The missing ratings are then computed as inner products of respective user and item vectors. Nearest neighbor methods [7] utilize the similarity between users based on user-item interactions to form a neighborhood for an active user, based on which recommendations are generated for the user. Symmetrically, it is possible to utilize similarity between items and compute item neighborhood, which is known as item-based recommendation [5, 22]. While matrix factorization and user-based kNN require user profiles in forms of user-item interactions, item-based nearest neighbor methods can be modified to work in session-based recommendation settings, in which items frequently appearing together in sessions are considered similar.

Content-based methods use content features of items to calculate similarities between them and recommend items similar to ones the user preferred in the past [2]. An advantage of CBF methods is their ability in handling new items, for which few or no user interactions exist. CF and CBF can be combined in so called hybrid systems to utilize the strengths of both methods [1]. Our work here also combines content feature with session clicks to boost the prediction accuracy.

Deep leaning

Deep learning has delivered state-of-the-art results in computer vision [16, 18], speech recognition [8], and several other application domains [17]. Two most popular deep learning models are convolutional neural networks (CNN) and recurrent neural networks (RNN). Other deep learning models include autoencoders, restricted Boltzman machines (RBMs), and fully connected networks with multiple hidden layers [17]. Deep learning methods have also been shown to be successful for recommendation tasks. A pioneering work along this direction was presented by Salakhutdinop et al. [21], in which several layers of RBMs are stacked together to deliver a better accuracy than a CF algorithm using singular value decomposition. Wang et al. [27] propose a hybrid method, in which they use a network of stacked denoising autoencoders to extract features from textual descriptions of items. The extracted features are then incorporated into a CF model to alleviate the problem when user-item interaction data are sparse. Van den Oord et al. [25] proposed a somewhat similar hybrid method, which uses deep learning to learn features from content descriptions of songs, which are then incorporated into a CF model to tackle the data sparsity problem. The difference is that they use CNNs for feature learning rather than autoencoders. Our method also uses CNNs and content features, but our CNN model allows capturing both spatial and temporal patterns, which is important for sequential nature of session clicks. Recently, Covington et al. [4] describe the system Google uses for video recommendation in Youtube. The system consists of a layer for learning user and item embeddings. These embeddings with other features such as time, video freshness etc. are then fed to several fully embedded layers to generate a distribution over millions of videos, from which top videos are recommended.

Session-based recommendation

Conventional CF and CBF do not work well in session-based settings, where user profiles are not available. A natural approach for this situation is item-based recommendation [5, 22], in which two items are deemed to be similar if they are frequently clicked together in the same sessions. Note that, this is slightly different from original item-based CF, in which two items are considered similar if they receive similar settings from the same set of users. While simple, item-based methods have been shown to be effective and widely deployed. A drawback of item-based recommendation is that it does not consider click order and generates predictions based only on the last click. Rendle et al. [20] propose so called Factorized Personalized Markov Chains to model sequential behaviors as a transition graph of items, which they use to predict next-basket items. Their work is similar to ours in that they predict next-basket items. However, unlike our work, they do that based only on previous baskets of the same user, while we make predictions based on all previous clicks without user identity. Figueiredo et al. [6] propose a Bayesian generative model to model click sequences. Their method is a generic method for any settings where data are sequences of events, which include session clicks. Learning item embeddings is another approach applicable for session-based recommendation. Inspired by learning word distributional representation in NLP, methods of this approach learn embeddings for items by training a shallow neural network to predict next purchased item in a transaction [9, 26]. The authors of [26] also leverage item metadata to regularize item embeddings, which makes it relevant to content-based approaches. Following successful applications of deep learning models in recommendation, Hidasi et al. [11] propose a pioneering work on using deep learning for session-based recommendation, in which they use RNN to model whole sequences of session click IDs. This work was then elaborated by Tan et al. [23], where they use a similar RNN architecture but with other loss functions and training data generation methods. In a later work, Hidasi et al. [12] extend their previous work by combining rich features of clicked items such as item IDs, textual descriptions, and images. They use different RNNs to represent different types of features and train those networks in a parallel fashion. Our work is relevant to [12] in that we combine features of different type for better session-based recommendation. However, our method uses a totally different model (3D CNN) and encoding method, which provide improved accuracy while simplify feature engineering steps.

3 METHODS

In this section we describe in detail the character-level representation for input data and the 3D CNN architecture for modeling both temporal and spatial patterns in sequences of clicks with associated content features. This is followed by an explanation of training procedure and training data preparation.

Let $[c_1, c_2, ..., c_n]$ $(n \geq 2)$ denote the sequence of item-viewing clicks for a given session, where c_i $(i = 1..n)$ is the ID of the i-th clicked item. In each session, there are zero or more clicks immediately followed by add-to-cart events (i.e. the viewed items are added to cart). Let $[a_1, a_2, ..., a_n]$ be an indicator vector so that $a_i = 1$ if c_i is added to cart, and $a_i = 0$ otherwise. For any given prefix $[c_1, c_2, ..., c_t]$ $(t = 1..n-1)$ of the session, the recommendation task is to predict subsequent add-to-cart items if such occur in the rest of the session, i.e. to predict $c_i | t < i \leq n, a_i = 1$. Note that this formulation is different from that of previous works, in which the task is to predict subsequent clicks rather than add-to-cart items [6, 11]. Predicting add-to-cart events, while more challenging, is more useful for e-commerce websites, as it guides users directly to items of highly probable purchase. We also assume that each item has a numerical ID, textual name and descriptions, and belongs to a category defined by the website owner. Here, we seek to find a model that takes into account all these information when making predictions.

3.1 Character-level Representation of Input

Inspired by the success of character-level CNNs in NLP [14, 28], we represent IDs and other item features using character-level encoding, which we describe in this section.

Let V denote the vocabulary of characters (alphabet) of size $|V|$. Suppose an item feature f is given as a sequence of characters $[v_1, v_2, ..., v_k]$, where k is the length of f . Then the character-level encoding of feature f is given by matrix $U^f \in \mathbb{R}^{|V| \times k}$, where the element at i-th row and j-th column is equal to 1 if v_j corresponds to i-th entry in V, and is equal to 0 otherwise. In other words, the j-th column of U^f is the 1-hot encoded vector of character v_j. In this work, we use vocabulary V of 55 characters, including all lower case characters from English alphabet, 10 digit characters, and several other characters. The characters are shown below:

```
abcdefghijklmnopqrstuvwxyz0123456789 -,
;.!?:"'/\|_@#$%
```

For each clicked item, we represent the associated features as follows.

- *Item ID*. A numerical ID is simply treated as a sequence of digit characters.
- *Name and descriptions*. It is straightforward to concatenate item name and descriptions which are already sequences of characters. In practice, the name is usually long enough and already provides a concise and informative description of an item, for example " iPhone 7 Smart Battery Case - Black". Therefore, we use only names and ignore additional descriptions to save computation time and reduce model size.
- *Category*. Categories are usually organized in a hierarchy by the website owner. To utilize the information encoded in the hierarchy, we concatenate the current category with all its ancestors up to the root and use the resulting sequence of characters as category feature, for example "apple/iphone/iphone7/accessories".

Given character-level encoded matrices for each type of features, we stack the matrices on top of each other to form a final matrix of

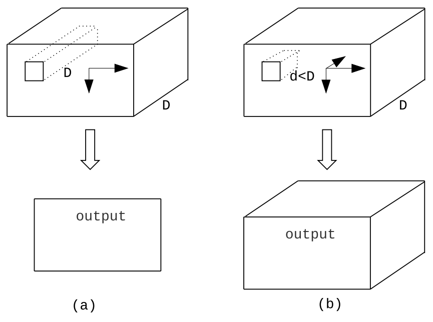

Figure 1: 2D and 3D convolution operations applied to multiple frames. a) 2D convolution results in 2D output. b) 3D convolution results in 3D output.

$X \in \mathbb{R}^{m \times n}$, where $m = |V| * F$ with F being the number of feature types (3 in this case), and n is the length of the longest feature (see figure 2 for an illustration). We set n to 150 in our experiments, ignoring characters beyond this limit. If a feature length is shorter than n then we add columns with all zeros to the right to get a matrix with exactly n columns.

3.2 3D CNN Architecture

Convolutional neural networks (CNNs) [18] are a type of neural network architectures that have achieved state-of-the-art results in computer vision , speech recognition and have been shown to be competitive in several NLP tasks [3, 17]. By applying convolution operations (known as kernels or filters) at different levels of granularity, a CNN can extract features that are useful for learning tasks and reduce the need of manual feature engineering. This characteristic is desired in our problem, where the goal is to extract from streams of clicks useful patterns, which are predictive of add-to-cart events. A pattern can be a sequence of clicks, a sequence of categories, a sequence of name parts, or their combination.

Traditional deployment of CNNs for computer vision mainly involves spatial convolutions, which are known as 2D CNNs. Our recommendation settings, however, involve both spatial and temporal information, which need to be modeled jointly. To achieve this, we adopt the 3D network architecture, originally introduced for video data [13, 24]. A main difference of a 3D network is that convolution and pooling are performed in all three dimensions (i.e. spatio-temporally), while in 2D networks they are performed spatially in two dimensions, even if the input is 3-dimensional. Figure 1 illustrates the difference between 2D and 3D architectures for 3-dimensional input. In the case of 2D networks, the 3D input volume collapses into a 2D structure right after the first convolution layer, thus loosing all temporal information. For 3D networks, the temporal information is preserved for subsequent convolution layers.

In this section, we describe the architecture of 3D CNNs for session-based recommendations. Figure 2 give a detailed illustration of the first layers, and Figure 3 shows the whole architecture of the proposed model. Recall that, for a click i, we represent the ID, name, and category using one character-level encoded matrix $U_i \in \mathbb{R}^{m \times n}$

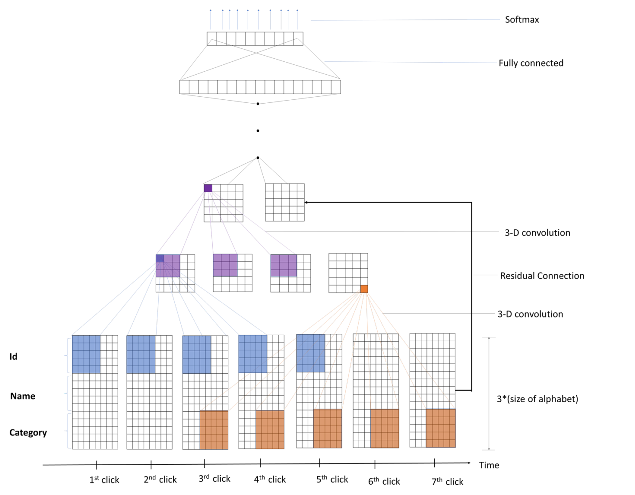

Figure 2: Illustration of the first and last layers.

(see the previous section). Inspired by computer vision terminology we will call such a matrix "frame". Then, a sequence of clicked is represented by putting corresponding frames side by side. As the result, we get a cube $U \in \mathbb{R}^{m \times n \times d}$, where d is the number of clicks. This will serve as the first feature map for the model.

We use a model consisting of several convolution layers stacked together. In each convolution stage, we perform a 3D convolution between a 3D kernel and the feature map at this stage, after which we add a bias and perform a nonlinear transformation to obtain a new feature map (Rectified Linear Unit [ReLU] was used as the linear transformation function for all layers). There are several kernels for each convolution stage, resulting in the corresponding number of feature maps for the next layer.

A recent work of He et al. [10] has shown the advantages of using residual connection in CNNs. We also employ this technique, adding residual connections to several convolution stages. There are two types of residual connections: (i). Identity connection applied to input and output of the same dimensions (solid line shortcuts in figure 3). (ii) The projection connection (dotted line), used to perform spatial downsampling by 1x1 convolution with a stride of 2. The filter shape in the first layer is [5,size of alphabet,5] and the stride is [2,2,2] (corresponding to [width, height, depth]). In the subsequent layers we use filters and average pooling of shape [3,3,3], and alternate between two stride shapes of [1,1,1] and [2,2,2] (see the figure). There are 16 filters in the first convolution stage. For the subsequent layers, we vary the of number of filters following two rules: (1) if the output height, width are the same as the input, the number of filter remain unchanged. (2). If the output height and width are halved, the number of filter is doubled. There is only one pooling layer before the fully connected layer. We use both L2 regularization and dropout with probability of 0.5 for the fully connected layer to prevent over fitting.

Our model uses classification-based output, where the output O' is a vector of probabilities over the items. This is achieved by using a softmax layer, trained to minimize a cross-entropy loss. For a given click, let $C_A = c_i$ be the set of subsequent add-to-cart items. We can

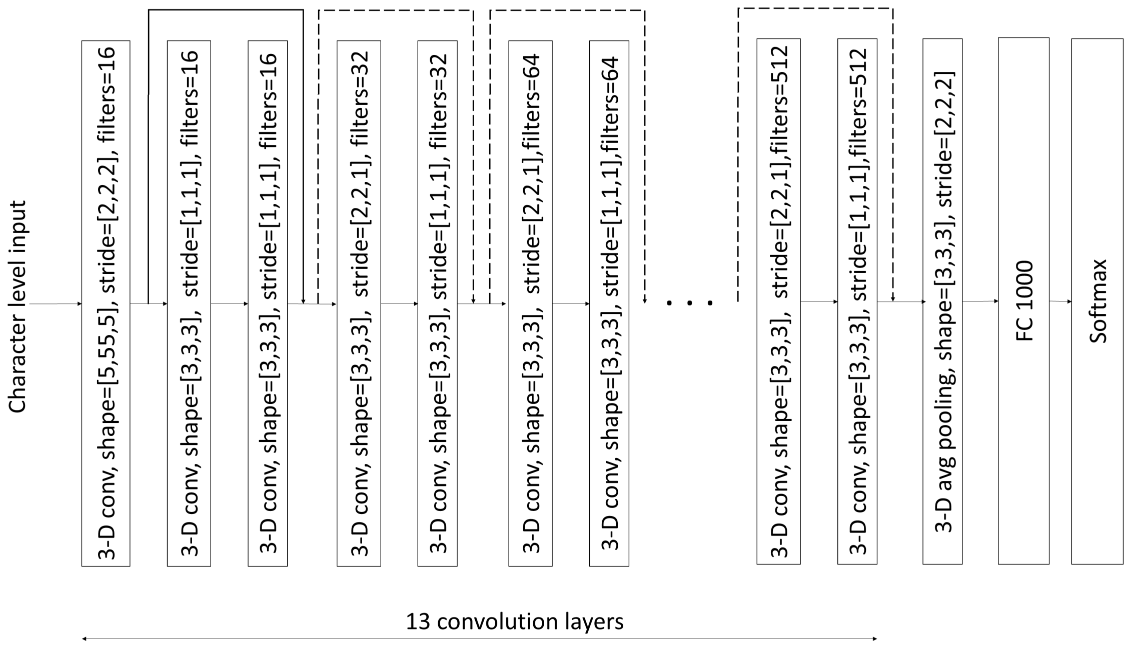
Character level input
3-D conv, shape=[5,55,5], stride=[2,2,2], filters=16
3-D conv, shape=[3,3,3], stride=[1,1,1], filters=16
3-D conv, shape=[3,3,3], stride=[1,1,1], filters=16
3-D conv, shape=[3,3,3], stride=[2,2,1], filters=32
3-D conv, shape=[3,3,3], stride=[1,1,1], filters=32
3-D conv, shape=[3,3,3], stride=[2,2,1],filters=64
3-D conv, shape=[3,3,3], stride=[1,1,1],filters=64
3-D conv, shape=[3,3,3], stride=[2,2,1],filters=512
3-D conv, shape=[3,3,3], stride=[1,1,1],filters=512
3-D avg pooling, shape=[3,3,3], stride=[2,2,2]
FC 1000
Softmax
13 convolution layers

Figure 3: Architecture of the 3D CNN model with detailed information for each layer. Solid lines correspond to residual identity connections. Dotted lines correspond to residual projection connections.

represent C_A as a $|C_A|$-hot vector O of size N (N is the number of items), so that elements corresponding to C_A members are set to 1 and the others are set to zeros. The model is trained to minimize the cross-entropy loss between O and O'. At prediction, the items are presented as a ranked list according to their predicted probabilities. From the list we use top K items for a recommendation.

3.3 Training

In this section we describe the procedure for preparing training data. For each session containing at least one add-to-cart item, we use all prefixes up to the last add-to-cart item as training sequences. For each such training sequence, all subsequent add-to-cart item IDs are used as corresponding training labels (see figure 4 for an illustration). Since a 3D CNN model must have a fixed temporal size (which we set to 7 in our experiments), we process training sequences that are shorter or longer than seven clicks as follows. For shorter sequences, we use right padding, i.e. adding frames with all zeros to the right, to get the required number of frames. For a longer sequence, we remove some clicks to retain the required number of clicks. There are three options to remove clicks:

- Remove the first clicks. This means only the last clicked items are used as input. The assumption behind this method is that the most recent clicks have more impact on add-to-cart decision.
- Remove the last clicks: only the first clicked items are retained. The assumption is that the first impression is more important while later clicks are less important.
- Remove the middle clicks and retain the same number of the first and last clicks. This combines the above two assumptions.

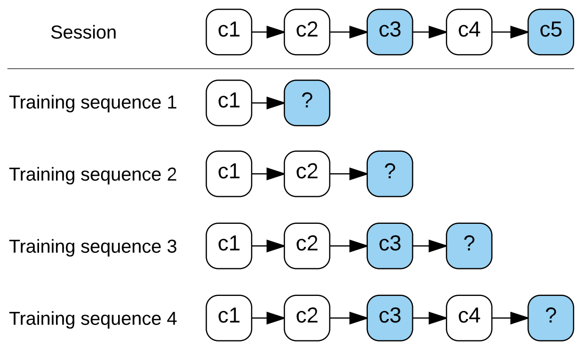

Figure 4: Generation of training sequences. Normal clicks are shown in white, clicks with add-to-cart are shown in blue. For each training sequence, the labels to predict are all subsequent add-to-cart clicks.

The same processing is applied at prediction time.

4 EXPERIMENTS AND RESULTS

In this section, we evaluate the performance of the proposed method compared to state-of-the-art session based recommendation algorithms on two datasets. We also reports results for different settings and parameter values.

Table 1: Statistics of the datasets

Datasets	JEWELRY	ELECTRONICS
Training set		
Sessions	42,698	30,982
Clicks	604,176	319,829
Sequences	591,884	307,135
Test set		
Sessions	4,966	4,421
Clicks	71,304	46,361
Sequences	69,135	45,127
Items	8,086	4,520

4.1 Experimental Setup

Datasets

We evaluated the proposed method on two proprietary datasets. The datasets were collected from two retail websites of small to medium size. The first dataset consists of product view events of a website selling jewelry products. The view events immediately followed by add-to-cart events were specially marked. The data were collected for a period of about three months without holiday or sale seasons. We will refer to this dataset as JEWELRY. The second dataset consists of product viewing clicks of a website selling electronics products over a period of three months. Similar to the first dataset, the view events with following add-to-cart were marked. We refer to this dataset as ELECTRONICS. Both sites only had simple recommendation in forms of most popular and new coming products. Note that the products in two datasets belong to two different domains of different characteristics. While jewelry is usually considered hedonic products (pleasure-oriented consumption), electronics products are more utilitarian (goal-oriented consumption) [19].

We removed all sessions not containing any add-to-cart event. We also removed sessions with only one click followed by an add-to-cart event because they are not suitable for session-based recommendation. We filtered items that were not added to cart in any session of the experimented period. For each product, its name (used as descriptions) and category name were retrieved and transformed to lower-case. Table 1 summarizes the statistics of the two datasets.

Setup

We used the sessions of the last ten days to form the test set and all remaining sessions for training. From session clicks, training and test sequences were generated as described in section 3.3. Each test sequence is fed to the trained model, from which a ranking over items is returned. We evaluated the experimented methods on how well they predicted subsequent add-to-cart clicks from each test sequence. In real-life settings, a recommender system presents to a user a short list of top-N recommended items. An accurate recommendation should contain actual add-to-cart items in this list. Thus, the first evaluation metrics here is recall@N. For a test sequence, recall@N is the number of actual subsequent add-to-cart items among top-N recommended items divided by the number of all subsequent actual add-to-cart items. We report recall@N averaged over all sequences in the test set.

The second metrics used in evaluation is Mean Reciprocal Rank (MRR@N), which is the average of reciprocal ranks of actual add-to-cart items with a rank set to zero if it exceeds N. MRR@N measures how well a recommender system assigns a higher rank for an actual item, thus is more sensitive than recall@N for ranking within top-N ranked items. We computed recall@N and MRR@N for N = 5, 10, 20, which correspond to different sizes of recommendation panels in webpages.

We conducted hyperparameter search on ELECTRONICS dataset, using a validation set consisting of the last 10% sessions from the training set. The optimal hyperparameters tuned on ELECTRONICS validation set were then used in all experiments. Specifically, we trained our model using mini-batches of size 250 with Adam optimizer and learning rate of 0.001. The model was trained for 100 epochs. The weights were initialized using truncated Gaussian with zero mean and standard deviation of 0.1. The length of each feature is limited to 150 characters. This length was enough for the longest category path in both datasets.

Baselines

We compared our method with the following session-based recommendation methods.

- Item-based kNN. The first, and very competitive baseline is Item-based kNN. This is a simple, yet effective method, which is widely deployed in practice. In this method, two item are considered similar if they co-occur frequently in different sessions. Because the problem is to predict subsequent add-to-cart events, we calculate the similarity between each item and only add-to-cart items. In prediction, the method outputs top-N items, which are most similar to the current clicked one.
- ID-RNN. This method [11] uses RNN to model session clicks and require only item IDs. The method was reported to deliver substantial improvement over Item-based kNN. We used the implementation provided by the authors ID-RNN with default parameters (https://github.com/hidasib/GRU4Rec). Although ID-RNN was originally proposed for next click recommendation, it is straightforward to train ID-RNN to predict subsequent add-to-cart items by using such items as training labels.
- Parallel-RNN. This method [12], proposed by the same authors of ID-RNN, also uses RNN but extend the model to consider textual descriptions of items and associated images, in addition to IDs as in ID-RNN. This is a leading method for session-based recommendation, which has been reported to be very effective in combining data of several types. Parallel-RNN is the most closely related to our method as it can combine different item features by combining different RNNs, trained on those features. We re-implemented Rich-RNN in TensorFlow with parameters corresponding to the best performance in their paper (bag-of-word TF-IDF for text, hidden unit of size 500+500, parallel networks, Interleaving training). Like in the case of ID-RNN, we trained Parallel-RNN to predict add-to-cart items by using such items as training

Table 2: Comparison of methods for shortening long sequences on JEWELRY dataset.

Shortening method	Recall@5	Recall@10	Recall@20
Only first clicks	0.3212	0.4005	0.4754
Only last clicks	0.3302	0.4078	0.4830
First and last clicks	0.3412	0.4152	0.4892

Table 3: Comparison of methods for shortening long sequences on ELECTRONICS dataset

Shortening method	Recall@5	Recall@10	Recall@20
Only first clicks	0.3662	0.4101	0.4734
Only last clicks	0.3654	0.4178	0.4810
First and last clicks	0.3907	0.4227	0.4894

labels. We used the same ID, names, and categories data as used with our method.

We could not compare our method with TribeFlow [6] because there is no obvious way to train TribeFlow to predict add-to-cart events.

4.2 Results

The first experiment was designed to evaluate the methods used in shortening long sessions to fit the width of the first network layer. For CNNs, this processing is necessary but can lead to information lost. Recall that we remove clicks from long sessions in three ways: remove the first, last, and medium clicks (see section 3.3). We applied these methods to process long sequences in both training and prediction and measured corresponding Recall@N for each method. The Recall@N values for three methods and two datasets are shown in Table 2 and Table 3. As can be seen, removing the medium clicks results in the highest Recall, while removing the first and last clicks result in lower Recall values, especially for top-5 recommendation. This patterns are similar across datasets. These results suggest that the first and last clicks are more important for predicting add-to-cart events than those in the middle. While the first clicks may be indicative of user intent when entering the site, the last ones may be indicative of user convergence on the items of interest. Middle clicks are likely be performed for exploration purposes. Based on these results, in the remaining experiments, we retained only the first and last clicks from long sequences.

In the next experiment, we evaluated the effectiveness of the proposed model in predicting add-to-cart items. This task is more difficult than predicting the next click because the distance between the current and add-to-cart click is usually much longer. To measure how the model is effective in recommending distant add-to-cart events we used the trained model to predict the last add-to-cart clicks in sessions of different lengths from the test set. For a test session of a given length, we generated all prefixes as described in section 3.3, used them as input for predicting the last add-to-cart item, and recorded Recall and MRR of the predictions. We used the Recall and MRR values, averaged over all prefixes as the measures of recommendation effectiveness for this session. This experiment

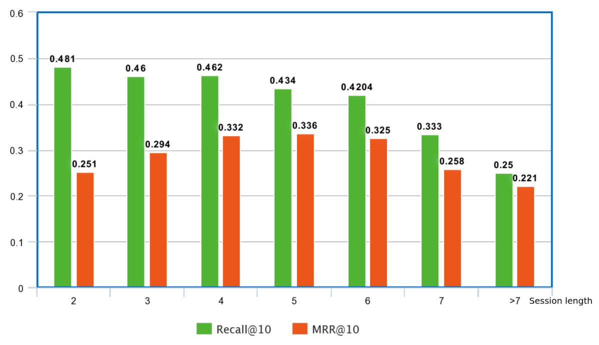

Figure 5: Effect of session length on accuracy. Shown for ELECTRONICS dataset.

was performed on ELECTRONICS dataset. Figure 5 shows averaged Recall@10 and MRR@10 as measures of recommendation effectiveness for each session length. As expected, Recall value gradually decreases as session length grows. For sessions of more than seven clicks, the Recall@10 drops to 0.25, compared to Recall@10 of 0.48 for sessions with two clicks. There are two possible explanations for this phenomenon. First, in a short session, the user is probably more focused and has stronger intent toward some items. Second, it is more difficult to predict something more distant in the future, which is the case in long sessions. At the same time, Recall@10 score of 0.25 for sessions with more than seven clicks is still impressive, showing that one fourth of actual add-to-cart items correctly appear among top-10 recommended. A somewhat less expected result is MRR@10, which achieved the highest value for sessions of length 5 and lower values for shorter and longer sessions. Despite this surprising result, the model achieved averaged MRR@10 of 0.22 for long sessions, which is encouraging for a challenging task like predicting distant add-to-cart events.

In the next experiment, we compared our model with baselines described in the previous section. In addition to the model using all item features (referred to as *3D-CNN Full*), we also measured the accuracy of the 3D CNN model using only IDs as input (referred to as *3D-CNN ID*). The results of experimented models on JEWELRY and ELECTRONICS datasets are summarized in Table 4 and 5 respectively. As can be seen from the tables, the deep learning models consistently outperformed the Item-based kNN baseline in terms of both Recall and MRR at all threshold values (5, 10, 20), despite the fact that Item-based kNN is a very competitive baseline. An interesting observation is the competitive performance of 3D-CNN ID, as compared to ID-RNN. Although the 3D CNN model with ID data achieved lower Recall and MRR scores than ID-RNN on the first dataset, it outperformed ID-RNN on the second dataset, in terms of most metrics, with only one exception - the Recall@20 score. This is interesting because RNNs are usually considered to be more suitable than CNNs in modeling sequential data. We believe the competitiveness of CNN in this case comes from the use of 3D convolution.

Models that combine names and categories with IDs, namely Parallel-RNN and 3D-CNN Full, achieved higher Recall and MRR

Table 4: Results on JEWELRY dataset. The best results for each metrics are shown in bold.

Method	Recall			MRR		
	Recall@5	Recall@10	Recall@20	MRR@5	MRR@10	MRR@20
Item-based kNN	0.2211	0.2990	0.3317	0.1476	0.1608	0.16341
ID-RNN	0.2512	0.3542	0.3944	0.1618	0.1774	0.1798
Parallel-RNN	0.3241	0.4095	0.4874	0.1754	0.1865	0.1923
3D-CNN ID	0.2365	0.2947	0.3650	0.1534	0.1616	0.1668
3D-CNN Full	**0.3412**	**0.4152**	**0.4892**	**0.1883**	**0.1934**	**0.1986**

Table 5: Results on ELECTRONICS dataset. The best results for each metrics are shown in bold.

Method	Recall			MRR		
	Recall@5	Recall@10	Recall@20	MRR@5	MRR@10	MRR@20
Item-based kNN	0.2859	0.3679	0.4108	0.1950	0.2059	0.2106
ID-RNN	0.2907	0.3708	0.4461	0.2025	0.2151	0.2208
Parallel-RNN	0.3469	0.4061	0.4861	0.2059	0.2162	0.2233
3D-CNN ID	0.3333	0.3795	0.4234	0.2478	0.2550	0.2580
3D-CNN Full	**0.3907**	**0.4227**	**0.4894**	**0.2781**	**0.2840**	**0.2883**

values than those using only IDs, across all datasets and metrics. This confirms the effectiveness of combining different features to improve recommendation accuracy.

The 3D CNN model using different features (3D-CNN Full) consistently outperformed other experimented models including state-of-the-art Parallel-RNN, in terms of all evaluation metrics, on both datasets. The superiority of 3D-CNN Full over Parallel-RNN is specially significant for top-5 and top-10 recommendation, as illustrated by Recall@5 and Recall@10 values. This has been observed in both datasets. The difference between two models, however, is not significant for Recall@20. 3D-CNN Full still achieved higher Recall@20 scores, but the difference is negligible. It is worth noticing that high Recall scores for top-5 and top-10 are specially important when the area for recommendation in a webpage is limited, and only a small number of items can be displayed. On ELECTRONICS dataset, 3D-CNN Full achieved much higher MRR scores than Parallel-RNN, for all three threshold values. The difference between MRR scores of two methods is smaller on JEWELRY dataset, but the margin is still substantial for MRR@5 and MRR@10.

5 CONCLUSION

We have introduced a 3D CNN model for session-based recommendation. The proposed model allows combining different item features such as ID, name, and metadata. This is achieved by two recipes: 1) using character-level encoding for all features; and 2) using CNN with 3D convolutions. We have shown that character-level encoding is applicable for important types of item features. More importantly, it frees us from most feature engineering steps. As a side effect, character-level encoding results in networks with fewer parameters as compared to 1-hot encoding of words, for example. 3D convolution allows jointly extracting temporal and spatial features, which is desired in modeling session data. We have presented experimental results with datasets from real e-commerce websites. The results demonstrate the effectiveness of our method in predicting add-to-cart events. This setting, although important in e-commerce, was not well explored previously. The proposed method outperformed all baselines including a state-of-the-art RNN based method. Traditionally, RNNs have been widely considered the model of choice for sequential data. The effectiveness of our method, in addition to existing successful applications of 3D CNN for video and time series data, suggest that 3D CNNs are also a suitable architecture for modeling sequential data, especially when sequence elements are associated with complex features. Finally, although the current model uses only ID and metadata, it is straightforward to incorporate other features such as timestamp or indication of past add-to-cart events from the same session. By doing this, the method can be easily extended to model co-purchase patterns, which have been shown to be useful in practice.

ACKNOWLEDGMENTS

This work was supported in part by FPT.

REFERENCES

[1] Ngo Xuan Bach, Nguyen Do Hai, and Tu Minh Phuong. 2016. Personalized recommendation of stories for commenting in forum-based social media. *Information Sciences* 352 (2016), 48 – 60.

[2] Marko Balabanović and Yoav Shoham. 1997. Fab: Content-based, Collaborative Recommendation. *Commun. ACM* 40, 3 (mar 1997), 66–72. DOI: https://doi.org/10.1145/245108.245124

[3] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *J. Mach. Learn. Res.* 12 (nov 2011), 2493–2537. http://dl.acm.org/citation.cfm?id=1953048.2078186

[4] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16).* ACM, New York, NY, USA, 191–198. DOI: https://doi.org/10.1145/2959100.2959190

[5] Mukund Deshpande and George Karypis. 2004. Item-based top- N recommendation algorithms. *ACM Transactions on Information Systems* 22, 1 (jan 2004), 143–177. DOI: https://doi.org/10.1145/963770.963776

[6] Flavio Figueiredo, Bruno Ribeiro, Jussara M Almeida, and Christos Faloutsos. 2016. TribeFlow: Mining & Predicting User Trajectories. In *Proceedings of the 25th International Conference on World Wide Web (WWW '16).* International World

Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 695–706. DOI:https://doi.org/10.1145/2872427.2883059

[7] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. 1992. Using Collaborative Filtering to Weave an Information Tapestry. *Commun. ACM* 35, 12 (dec 1992), 61–70. DOI:https://doi.org/10.1145/138859.138867

[8] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 6645–6649.

[9] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. 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 (KDD '15)*. ACM, New York, NY, USA, 1809–1818. DOI:https://doi.org/10.1145/2783258.2788627

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[11] Balázs Hidasi, Alexandros Karatzoglou, L. Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In *International Conference on Learning Representations*.

[12] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. 2016. Parallel Recurrent Neural Network Architectures for Feature-rich Session-based Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, New York, NY, USA, 241–248. DOI:https://doi.org/10.1145/2959100.2959167

[13] S Ji, W Xu, M Yang, and K Yu. 2013. 3D Convolutional Neural Networks for Human Action Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 1 (jan 2013), 221–231. DOI:https://doi.org/10.1109/TPAMI.2012.59

[14] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware Neural Language Models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI'16)*. AAAI Press, 2741–2749. http://dl.acm.org/citation.cfm?id=3016100.3016285

[15] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (aug 2009), 30–37. DOI:https://doi.org/10.1109/MC.2009.263

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS'12)*. Curran Associates Inc., USA, 1097–1105. http://dl.acm.org/citation.cfm?id=2999134.2999257

[17] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.

[18] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. 1990. Handwritten Digit Recognition with a Back-Propagation Network. In *Advances in Neural Information Processing Systems*. Morgan Kaufmann, 396–404.

[19] Dokyun Lee and Kartik Hosanagar. 2016. When Do Recommender Systems Work the Best?: The Moderating Effects of Product Attributes and Consumer Reviews on Recommender Performance. In *Proceedings of the 25th International Conference on World Wide Web (WWW '16)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 85–97. DOI:https://doi.org/10.1145/2872427.2882976

[20] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing Personalized Markov Chains for Next-basket Recommendation. In *Proceedings of the 19th International Conference on World Wide Web (WWW '10)*. ACM, New York, NY, USA, 811–820. DOI:https://doi.org/10.1145/1772690.1772773

[21] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted Boltzmann Machines for Collaborative Filtering. In *Proceedings of the 24th International Conference on Machine Learning (ICML '07)*. ACM, New York, NY, USA, 791–798. DOI:https://doi.org/10.1145/1273496.1273596

[22] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International Conference on World Wide Web (WWW '01)*. ACM, New York, NY, USA, 285–295. DOI:https://doi.org/10.1145/371920.372071

[23] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved Recurrent Neural Networks for Session-based Recommendations. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS 2016)*. ACM, New York, NY, USA, 17–22. DOI:https://doi.org/10.1145/2988450.2988452

[24] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. 2015. Learning Spatiotemporal Features with 3D Convolutional Networks. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV) (ICCV '15)*. IEEE Computer Society, Washington, DC, USA, 4489–4497. DOI:https://doi.org/10.1109/ICCV.2015.510

[25] Aäron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep Content-based Music Recommendation. In *Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS'13)*. Curran Associates Inc., USA, 2643–2651. http://dl.acm.org/citation.cfm?id=2999792.2999907

[26] Flavian Vasile, Elena Smirnova, and Alexis Conneau. 2016. Meta-Prod2Vec: Product Embeddings Using Side-Information for Recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, New York, NY, USA, 225–232. DOI:https://doi.org/10.1145/2959100.2959160

[27] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative Deep Learning for Recommender Systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15)*. ACM, New York, NY, USA, 1235–1244. DOI:https://doi.org/10.1145/2783258.2783273

[28] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS'15)*. MIT Press, Cambridge, MA, USA, 649–657. http://dl.acm.org/citation.cfm?id=2969239.2969312