Behavior2Vec: Generating Distributed Representations of Users' Behaviors on Products for Recommender Systems

HUNG-HSUAN CHEN, National Central University

Most studies on recommender systems target at increasing the click through rate, and hope that the number of orders will increase as well. We argue that clicking and purchasing an item are different behaviors. Thus, we should probably apply different strategies for different objectives, e.g., increase the click through rate, or increase the order rate. In this article, we propose to generate the distributed representations of users' viewing and purchasing behaviors on an e-commerce website. By leveraging on the cosine distance between the distributed representations of the behaviors on items under different contexts, we can predict a user's next clicking or purchasing item more precisely, compared to several baseline methods. Perhaps more importantly, we found that the distributed representations may help discover interesting analogies among the products. We may utilize such analogies to explain how two products are related, and eventually apply different recommendation strategies under different scenarios. We developed the Behavior2Vec library for demonstration. The library can be accessed at https://github.com/ncu-dart/behavior2vec/.

CCS Concepts: • **Information systems** → **Recommender systems**; *Collaborative filtering*; • **Computing methodologies** → **Learning latent representations**; • **Computer systems organization** → *Neural networks*;

Additional Key Words and Phrases: Distributed representation, behavior embedding, Word2Vec, collaborative filtering, matrix factorization, learning representations

ACM Reference format:
Hung-Hsuan Chen. 2018. Behavior2Vec: Generating Distributed Representations of Users' Behaviors on Products for Recommender Systems. *ACM Trans. Knowl. Discov. Data.* 12, 4, Article 43 (April 2018), 20 pages.
https://doi.org/10.1145/3184454

1 INTRODUCTION

Recommender systems are widely adopted nowadays to actively suggest useful, relevant, and interesting events or objects from large amount of information. For example, a news website may recommend articles based on the prediction of users' interest (Liu et al. 2010). In academic circle, researchers may rely on digital libraries and recommendation tools to discover suitable references for a working paper (Huang et al. 2014), potential collaborators (Chen et al. 2011), or the domain experts of a certain area (Chen et al. 2013, 2015; Tang et al. 2008). Among various types of recommender systems, the most typical kind is probably provided by the online retailers (such as Amazon and Walmart) or online service providers (such as Netflix or Spotify). These online shops

This work is partially supported by the Industrial Technology Research Institute (ITRI) and the Ministry of Science and Technology (MOST).
Author's address: H.-H. Chen, Computer Science and Information Engineering, National Central University, No. 300, Zhongda Rd., Zhongli District, Taoyuan City 32001, Taiwan (R.O.C.); email: hhchen@g.ncu.edu.tw.

https://doi.org/10.1145/3184454

suggest products, movies, or other items that are likely to be of interest to the users, in the hope that users may purchase more products or continue using their service for a longer period, and eventually spend more money.

Among the recommendation algorithms, Collaborative Filtering (CF) is probably the most famous and widely adopted type of method (Breese et al. 1998; Sarwar et al. 2001). CF finds similar users or similar items based on all (or many) users' collective browsing and purchasing behaviors on a website. This idea motivates many following works, such as Matrix Factorization (MF) (Koren et al. 2009), neural network based CF (Salakhutdinov et al. 2007), MF based on Bayesian probability (Salakhutdinov and Mnih 2008), and so on.

However, previous works mostly analyze users' clicking (viewing) behaviors and target at increasing the click through rate. Unfortunately, clicking and purchasing are different behaviors. As recently shown in Chen et al. (2017), increasing click through rate may not necessarily increase the number of orders. This motivates us to differentiate users' behaviors (e.g., clicking and purchasing) on items. In this article, we propose to generate the *distributed representations* of users' clicking and purchasing behaviors on the products, based on the logs of a large e-commerce (EC) website. We then recommend the products that may interest the user based on the cosine distance between the generated distributed representations. Because we leverage on many users' collective browsing and purchasing history, our method can be categorized as one type of CF.

The central concept of our model — distributed representation — represents a many-to-many mapping between two types of objects: an entity is represented by many (neuron-like) concept elements, and a concept element is involved in representing many entities (Hinton et al. 1986). Recently, researchers in the information retrieval (IR) community utilized the neural network techniques to define the distributed representations of words (a.k.a. word embedding) (Mikolov et al. 2013b) and the distributed representations of documents (a.k.a. document embedding) (Le and Mikolov 2014). They discovered that word embedding reveals surprisingly precise semantic and syntactic relationship among words. For example, they found that $Paris - France + Italy \approx Rome$ (Mikolov et al. 2013a) (semantic relationship), $King - Man + Woman \approx Queen$ (Mikolov et al. 2013c) (semantic relationship), $walking - walked + swam \approx swimming$ (Mikolov et al. 2013a) (syntactic relationship), $mouse - mice + dollars \approx dollar$ (Mikolov et al. 2013a) (syntactic relationship), and the like.

Encouraged by the success of the distributed representation in the IR domain, we are interested in studying the distributed representations of users' behaviors on an EC website. Conceptually, we model users' behaviors on products (i.e., viewing a product, buying a product, etc.) as different events and generate the distributed representations of these events based on the Word2Vec model. We found that, because we differentiate the behaviors, Behavior2Vec may recommend different products under different scenarios. To infer the product analogy based on embedding, we post-process the behavior embedding to form the distributed representations of the products. Such a technique discovers better product analogy compared to the Prod2Vec model, which will be introduced later.

We conducted extensive experiments based on the server logs on a large-scale EC website in Taiwan and Southeast Asia. We found that, compared to several baseline recommendation methods, we better predict the next product a user is going to click or purchase, based on the cosine distance between the learned distributed representations. Perhaps more importantly, we found that product analogy can be inferred based on the distributed representations. For example, our method can successfully discover the mapping of camera body and the corresponding kit lens of five different brands (Canon, Nikon, Panasonic, Sony, and Pentax). We may further utilize such analogy to infer a user's intension and provide different recommendation strategies under different scenarios.

This article makes the following contributions:

—We argue that increasing the click through rate may not necessarily increasing the order rate. We propose a simple yet effective Behavior2Vec model to generate the embedding of the behaviors on the items, so that the model can recommend different items under different objectives. We developed and open sourced the Behavior2Vec library.[1]
—We found that, compared to several popular baseline methods, the Behavior2Vec indeed make better recommendations, especially, when the task is to predict the next *purchased* item given previous *clicked* items. If there exists only one type of behavior (e.g., predicting the next clicked item based on clickstreams) in the log, Behavior2Vec performs better than some baselines and comparable to the strong baselines, such as the Prod2Vec model, the session-based Recurrent Neural Network (S-RNN), and the Factorizing Personalized Markov Chains (FPMC) model.
—By post-processing the embedding of the behaviors on items, we can infer the relationship between different products. Based on case studies, our method better infers the relationship between the products, compared to the Prod2Vec model.

The rest of the article is organized as follows. In Section 2, we review previous works on recommendation algorithms and recent progress of the distributed representations of texts in the IR domain, especially about word embedding. Section 3 introduces the method to compute the distributed representations of the behaviors. Section 4 reports the experiments and results, including (1) a comparison of various recommendation algorithms on predicting the next viewing or purchasing products, and (2) case studies about the analogy of products inferred from the generated distributed representations. Finally, we discuss our discoveries and future research directions in Section 5.

2 RELATED WORK

This section reviews the classic recommendation algorithms, recent progress of the neural network-based recommendation models, and the word embedding studies in the IR domain.

2.1 Recommendation Algorithms

Researchers often categorize the recommendation algorithms into two paradigms: content-based and CF (Lops et al. 2011). The content-based algorithms leverage on the product profiles and the user profiles, mostly in text format, to define the distance between two products, between two users, or between a product and a user. The widely used method includes TF-IDF (term frequency–inverse document frequency), topic-modeling, and the variations of these methods (Pennacchiotti and Gurumurthy 2011). However, the performance of the content-based recommendation relies heavily on the quality of the content, which is not always clean or available in practice.

CF-based methods utilize many users' collective interaction with the products to make recommendation. The most famous algorithm of this type is probably Amazon's item-to-item CF, in which two items are considered similar if they are purchased or clicked by a similar set of users (Linden et al. 2003). Likewise, we may claim two users are similar if they purchased or clicked a similar set of items, and recommend items to a user u based on the browsing behavior of the users who are like u. This method is often called user-based CF.

Instead of recommending the top-k products that might be appealing to users, sometimes it is desired to predict users' ratings on items. This is often modeled by the MF techniques, which

[1] https://github.com/ncu-dart/behavior2vec/.

approximate the missing values in a large user-by-product rating matrix based on the product of the low-ranked matrices. This type of methods have shown successful results (Hu et al. 2008; Koren 2008; Koren et al. 2009). Recently, Wu et al. (2017) showed that, instead of using fixed latent factor vectors for the users and the items, varying the latent factors over time may improve the prediction accuracy, especially, when the log contains the records of an individual for a long period, e.g., months to years. For a short period, however, the latent factors of the users and the items are unlikely to vary dramatically.

Although MF techniques dominated the user rating prediction tasks (Koren et al. 2009), studies have shown that MF-based method may not perform well in the top-k recommendation task (Cremonesi et al. 2010), which might be a more realistic recommendation scenario. Association Rules (AR) discovers the relationship between items based on their co-purchasing or co-viewing frequency (Tan et al. 2006). The central idea of AR is very close to the item-to-item CF used by Amazon. However, AR must specify the thresholds (the support and the confidence) to rule-out the uninteresting rules. As a result, AR usually generate a few rules, but these rules might be powerful. Another related method, FPMC (Rendle et al. 2010), generates personalized transition graph of item clicks based on MF. The experimental results showed that FPMC performs better than MF or the global transition graph. The main issue of behavior-based methods is the cold start problem in which the recommender systems do not have enough clues to make a recommendation because a new user just started using the EC website, or a new item just started to be sold. Recently, researchers found that assigning non-uniform weights to the negative (missing) data may generate a better model (He et al. 2016). Specifically, they set the weight of a negative instance to be positively related to the popularity of the item. The argument is that, since users are more likely to be aware of the popular items, these items are likely to be the true negatives if not clicked or purchased by users. On the other hand, the less-popular items are ignored probably because users are unaware of these items.

Our approach utilizes the browsing and purchasing logs from all users to generate the distributed representations. Thus, it can be regarded as a behavior-based approach.

2.2 Neural Network-Based Recommendation Models

The traditional MF assumes the interaction between a user and an item based on an inner product operation. However, this probably over-simplifies the scenario. Recently, researchers have started to utilize the (deep) neural network to model higher degree of interactions between users and items (Hong-Jian Xue 2017; He et al. 2017). Such methods may also incorporate the explicit user features or item features into the models. The time information and sequence information can also be included by the neural network models. For example, the S-RNN model (Hidasi et al. 2016) applied the RNN (specifically, Gated Recurrent Unit) model on sessions to predict the next item in the clickstream. They found that such a mechanism outperforms item-to-item recommendation, probably because S-RNN models the whole session instead of only the latest click. However, S-RNN does not generate product embeddings. Thus, it could be difficult to produce the relationship among the products based on the model. Recently, Wu et al. (2017) showed that the latent factors of users and the items may vary over time. Thus, if the training data contain the logs for a long period, e.g., months to years, it might be less effective to infer fixed latent factors for all the users and items. They proposed to generate the latent factors for users and items continuously based on the LSTM (long short-term memory) model.

The embedding-based approach can also be derived based on neural networks. Since the embedding-based approaches are highly influenced by the Word2Vec model in the IR domain, we introduce such approach in the next section.

2.3 Distributed Representations of Words

In the IR domain, the distributed representations of words, sometimes called word embedding, refers to a many-to-many mapping between the words and the hidden dimension, which typically represents the semantic and syntactic concepts of the words (Bengio et al. 2003; Hinton et al. 1986). Researchers have developed several methods to transform the words from the one-hot-encoding representations into the distributed representations, which are essentially dense vectors of a lower dimension. Studies have shown that word embedding captures precise word analogy in terms of both semantics and syntax. In addition, we may treat the word embedding vectors as the features and feed supervised learning algorithms with these features. Such a setting usually yields better results compared to the traditional word features using one-hot-encoding.

Researchers may leverage on the neural networks to generate word embedding, because the structure of a neural network fits naturally to the concept of distributed representation. Depending on the training objective, we can further categorize the neural network model into following two types: the skip-gram model and the continuous bag-of-words (CBOW) model (Mikolov et al. 2013a). The skip-gram model predicts the surrounding words given the central (current) word, whereas the CBOW model predicts the central word given the surrounding words. Both models, in their original format, are inefficient in training. In practice, researchers usually utilize the negative sampling technique or the hierarchical softmax technique to address the computational issue. The neural network-based methods are also called the prediction-based model, because the neural network method aims at maximizing the accuracy of predicting the word usage in a sentence. The word embedding vectors are the natural by-products of the optimization process – we adjust these vectors during the training process to meet the objective.

In addition to the neural network (prediction-based) models, we may perform dimension reduction on the matrix of the word co-occurrence statistics to generate the word embedding (Bullinaria and Levy 2012; Lebret and Collobert 2013; Pennington et al. 2014). Because this type of methods relies on counting the co-occurrence of words, it is also called the count-based model. Researchers have conducted experiments to measure the performances of both the prediction-based and the count-based models (Baroni et al. 2014; Pennington et al. 2014). Although these two models look different, careful studies have shown that fundamentally they are very similar (Levy and Goldberg 2014; Pennington et al. 2014). Recently, researchers have applied the concept of embedding on longer texts to discover the sentence embedding and the paragraph embedding (Le and Mikolov 2014).

Recent studies have started to apply word embedding technique on other domains. The closest work to our study is Yahoo Inbox's product recommendation (Grbovic et al. 2015). The work discovers the distributed representations of the products (this is the Prod2Vec model we compared in Section 4). Essentially, Prod2Vec can be regarded as treating each product as one word and each clickstream as sentences. Our method, Behavior2Vec, on the other hand, differentiates clicking an item from purchasing an item, so that the distributed representations are defined by not only the items but also users' behaviors. Compared to the Prod2Vec model, Behavior2Vec is capable of recommending different products under different scenarios. Perhaps more importantly, Behavior2Vec generates the embedding that better reveals the product analogy, which could be a more fundamental contribution to the society of the recommender systems.

3 GENERATING DISTRIBUTED REPRESENTATIONS FOR USER BEHAVIORS

This section introduces the method to generate the distributed representations for users' behaviors (in particular, we differentiate the clicking and the purchasing behaviors in this article). We first

present the overview and the motivation of the model, followed by defining the terms that will be used later. Finally, we formally introduce the model.

3.1 Motivation and Model Overview

In the IR community, several researchers believe that the context of a word can be defined by its surrounding words (Chan 2014). Thus, two words are conceptually similar if the same or similar set of words surrounds the two.

Partially motivated by this idea, we suspect two *behaviors* b_1 and b_2 are similar if the surrounding set of the behaviors of b_1 is like the surrounding set of the behaviors of b_2. Thus, if we are given a model to predict the surrounding behaviors of a behavior, we can judge the similarity of two behaviors based on the distance between their predicted surrounding behaviors.

Later in this section, we will formalize this idea by showing how to leverage on the skip-gram model (Mikolov et al. 2013b) to predict the surrounding behaviors given a behavior.

The closest work we could find is Yahoo Inbox's product recommendation (Grbovic et al. 2015), which treats each *item* as one word and the item sequences as sentences. However, as shown in Chen et al. (2017), clicking (viewing) and purchasing are different behaviors. For example, increasing the click through rate may not necessarily increase the order rate. We empirically found that differentiating the behaviors (i.e., purchasing or clicking) on the items can improve the result. As far as we know, we are the first to apply the Word2Vec model to discover the behavior embedding, which is later utilized to build the item embedding. In fact, the Prod2Vec model is a special case of the Behavior2Vec model in which only one type of behavior (e.g., clicking items) exists.

Compared to the word usage in a sentence, it seems that the clicking or the purchasing sequences of items could be noisier, as unrelated items can be viewed or clicked in one sequence. However, it appears that users still managed to click or purchase the relevant items in a session in many cases. For example, we found that users tend to click products of the same brand during a session. Even though some viewing or the purchasing sequences are noisy, as the number of training instances are enough, the learning models can still discover the patterns and filter the noisy information. In fact, many successful recommendation strategies (e.g., association rules, user-based or item-based CF, MF, and S-RNN) are applied on the noisy clickstreams, and the models can still identify useful rules.

3.2 Terminology

For better explanation, here we define the terms that will be used later.

Given a user's browsing and purchasing log as the following sequence of entries $s = \{e_1, e_2, \ldots, e_{i-1}, e_i, e_{i+1}, \ldots\}$ and a pre-defined *window size* parameter w, we define $C(e_i)$ the *surrounding entries* of a behavior entry e_i as the set of the behaviors shown in the following equation:

$$C(e_i) = \{e_{i-w}, e_{i-w+1}, \ldots, e_{i-1}, e_{i+1}, \ldots, e_{i+w}\}. \tag{1}$$

Figure 1 shows an example of the surrounding entries of the behavior entry e_2, e_5, and e_9, with the pre-defined windows size $w = 3$. For e_5, the previous three products and the following three products are $\{e_2, e_3, e_4\}$ and $\{e_6, e_7, e_8\}$, respectively; thus, $C(e_5)$ the surrounding products of e_5 are $\{e_2, e_3, e_4, e_6, e_7, e_8\}$. On the other hand, there is only one product before e_2; thus, the number of surrounding products of e_2 is only four: $\{e_1, e_3, e_4, e_5\}$. Similarly, $C(e_9)$ the surrounding products of e_9 are only $\{e_6, e_7, e_8\}$.

The distributed representations of words are sometimes called "word embedding." Following this convention, we also use the term *behavior embedding* to refer to the distributed representations of the behaviors. In addition, we use the term "surrounding behaviors" and "context behaviors" interchangeably.

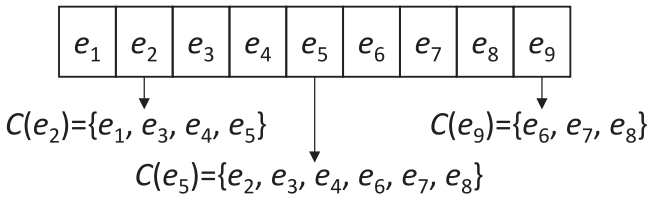

Fig. 1. An example of the surrounding entries of e_2, e_5, and e_9, given the pre-defined window size = 3. When the window size exceeds the beginning or the end, the exceeding parts are ignored.

3.3 Behavior Embedding Model

Given a session that browses through a list of products of the sequence $s = \{e_1, e_2, \ldots, e_{i-1}, e_i, e_{i+1}, \ldots, e_{|s|}\}$, where $|s|$ is the length of the sequence, we define the probability function of observing such a sequence based on the skip-gram model (Mikolov et al. 2013b), as shown by the following equation:

$$p(s) = \prod_{i=1}^{|s|} \left(\prod_{e_j \in C(e_i)} p(e_j|e_i;\theta) \right), \tag{2}$$

where $C(e_i)$ returns the context products of the product e_i, as defined in Section 3.2., and e_i defines the user's behavior on the product i by a dense vector, which we call the behavior embedding. Specifically, in this article, we assume that a user's possible behaviors on a product i is either "viewing" (clicking) it or "purchasing" it. Thus, we may further define $e_i \in \{v_i \text{ (viewing)}, p_i \text{ (purchasing)}\}$ to differentiate the two different behaviors on product i. We call v_i and p_i as the *viewing embedding* and the *purchasing embedding* of product i, respectively.

We define the objective function as the logarithm to the probability function (Equation (2)) to remove the product operations. As a result, the goal is to obtain the parameter θ that maximizes the log of the probability function, as defined by the following equation:

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{|s|} \sum_{\forall e_j \in C(e_i)} \log(p(e_j|e_i;\theta)) \tag{3}$$

In the typical setting of word embedding, the parameter θ is composed of $w_i^{(I)}$ and $w_i^{(O)}$, which represent the input vector representation and the output vector representation, respectively, for each word i (Mikolov et al. 2013b). However, since we differentiate the "viewing embedding" and the "purchasing embedding" of a given product i, the parameter θ in our case is composed of $v_i^{(I)}$ (the input viewing embedding of the product i), $v_i^{(O)}$ (the output viewing embedding of the product i), $p_i^{(I)}$ (the input purchasing embedding of the product i), and $p_i^{(O)}$ (the output purchasing embedding of the product i).

We set $e_i^{(I)} \in \{v_i^{(I)}, p_i^{(I)}\}$ and $e_i^{(O)} \in \{v_i^{(O)}, p_i^{(O)}\}$. Thus, the probability $p(e_j|e_i;\theta)$ can be defined based on the softmax function, as shown in the following equation:

$$p(e_j|e_i;\theta) = \frac{\exp(e_j^{(O)} \cdot e_i^{(I)})}{\sum_{k=1}^{K} \exp(e_k^{(O)} \cdot e_i^{(I)})}, \tag{4}$$

where K is the number of distinct possible behaviors on all the products. Equation (4) can be treated as a proxy of the similarity score between the behavior e_i and the behavior e_j because of the inner-dot product operation in the numerator.

We may leverage on any optimization method to obtain θ the parameters that maximizes the objective function (Equation (3)). Here, we take the derivative of the objective function to every parameter (i.e., all the elements in the vectors $e_i^{(I)}$ and $e_i^{(O)}$) and apply stochastic gradient descent to iteratively update all the parameters. Interested readers can refer to Rong (2014) for details.

In practice, an EC website typically provides hundreds of thousands to millions of different products. Thus, there is a huge number of combinations on the possible behaviors for all these products. As a result, it is inefficient to obtain the denominator of Equation (4) for every possible entry e_k of every session. We apply hierarchical softmax and negative sampling to address the issue. Hierarchical softmax obtains the probability distribution with a time complexity $O(\log K)$ rather than $O(K)$ based on constructing a Huffman tree. Negative sampling samples a small set of behaviors each time as the negative instances. Both methods are successfully applied in the IR domain to efficiently generate the distributed representations of words (Mikolov et al. 2013b).

During the online recommendation phase, the model applies different recommendation strategies based on the requested objective. If the target is to increase the click through rate, Behavior2Vec recommends item whose corresponding output *viewing* embedding is closest to e_i the embedding of the current behavior, in terms of cosine similarity. On the other hand, if the target is to increase the order rate, Behavior2Vec recommends items whose corresponding output *purchasing* embedding is closest. The recommendation formula is formalized as follows:

$$j = R(e_i) = \begin{cases} \arg\min_{\forall k \neq i} \cos(e_i^{(I)}, p_k^{(O)}) & \text{if the objective is maximize order rate} \\ \arg\min_{\forall k \neq i} \cos(e_i^{(I)}, v_k^{(O)}) & \text{if the objective is maximize click through rate,} \end{cases} \tag{5}$$

where the function cos() returns the cosine similarity between the two input vector parameters.

3.4 Time Complexity

Here, we discuss the time complexity of model training and online recommendation.

3.4.1 Model Training. The training process of Behavior2Vec model is based on the skip-gram model, which uses the embedding of the current behavior on items to predict the surrounding behaviors on the items. Like (Mikolov et al. 2013a), we use the hierarchical softmax and negative sampling for faster computation. Thus, the time complexity is $W(S + S\log(BI)) \approx S\log(BI)$, where B is the number of behaviors (in this article, $B = 2$, since we only consider item clicking and purchasing), I is the number of items, W is the size of the surrounding entries (usually much smaller than BI and S), and S is the number of sessions. As a result, it takes only hours to train a model with 1-month log as the input.

3.4.2 Online Recommendation. Given e_i the embedding of the current behavior on an item, we need to find the k closest embedding and make recommendation based on these embeddings. However, it is challenging to locate the k closest embedding in real time. To bypass this issue, we compute the k-nearest embedding for every embedding once a day, and store the information in a relational database. Therefore, the online recommendation only involves of database lookup, which is very efficient in practice. In fact, most recommender systems rely on similar techniques for online recommendation, regardless of the training model.

4 EXPERIMENT

We measure the performance of Behavior2Vec in this section. First, we make recommendations based on the Behavior2Vec generated embedding according to Equation (5). We compare the recommendation hit rate with several baselines based on several prediction tasks, as explained in

Table 1. The Information of the Training and the Test Datasets

ID	Period	# Sessions	# Clicked Products	# Unique Clicked Products
train	Day1–Day 7	1,282,280	11,629,989	558,031
test-1	Day8 00:00:00–11:59:59	26,482	36,197	26,482
test-2	Day8 12:00:00–23:59:59	34,022	47,584	34,022
test-3	Day9	54,537	85,235	54,537

The numbers of the purchased and distinct purchased items are omitted to protect the business sensitive information.

Section 4.1.1. Next, we present case studies on the product analogy discovered based on the distributed representations of the behaviors.

4.1 Next Behavior Prediction

4.1.1 Evaluation Method. In this section, we report the performance of the following prediction tasks, which are used to demonstrate different scenarios: Task 1: given the current *viewing* item, what product will the user *click* (view) next? Task 2: given the current *viewing* product, what product will the user *purchase* next? Task 3: given the item a user just *bought*, what will the user *click* (view) next? Task 4: given the item a user just *bought*, what will the user *buy* next? We measured the correctness of several recommendation modules and utilized the correctness scores as the proxy of the performance of each compared method. We measure the four tasks, because we believe that some methods may have better performance only in certain scenarios. For example, some recommendation modules may tend to recommend *alternative* products to the current browsing item. This type of recommendation may receive a high click rate but not necessarily the order rate.

Because a user can only have one action (i.e., viewing or purchasing an item in our study) at a given time point, the common correctness measures, such as average precision, precision-at-k, or mean average precision, may be inappropriate metrics. In addition, the EC website from which we collected the logs shows the recommendation items by a row of items, which makes the ranking-based measures (e.g., Discounted Cumulative Gain or Mean Reciprocal Rank) ill-suited. As a result, we defined the *average-hit-at-k* as the evaluation metric. The metric hit-at-k returns 1 if the next clicked (or purchased) item appears in the top-k of the prediction list and 0 otherwise. The average-hit-at-k is simply the average of the hit-at-k's for each test case.

4.1.2 Experiment Data. We sampled sessions from the server log of seven continuous days (Day1–Day7) to generate the training data. We compiled three datasets (test-1, test-2, and test-3) for testing. These test datasets were sampled from the server log on Day8 (00:00:00–11:59:59), Day8 (12:00:00–23:59:59), and Day9, respectively. Table 1 shows the information of the training and the three test datasets. The number of the purchased items and the distinct number of the purchased items are not shown to protect the business sensitive information.

4.1.3 Compared Baseline Methods. Here, we briefly introduce the compared baseline methods. Note that we treat each session as a distinct user. In other words, if a user arrives at the website on two different time points and the system regards the two arrivals as two different sessions, we treat the two sessions as two different users. Thus, we use the term session and user interchangeably.

—Prod2Vec (Grbovic et al. 2015): The model learns the product embedding based on the Word2Vec model by treating users' clickstreams as the "sentences" and the products in the

streams as the "words." The model recommends the product j whose corresponding vector is closest to the user's current browsing product i. The window is set to 5, and the size of the embedding is 300.

—Class Top Popular (ClassTP): We obtained the mapping of the products to the product categories that were manually organized by the employees of the EC website. Based on the information, we computed the most popular items in each category. When a user is browsing an item of a category, ClassTP recommends the 10 most popular items in this category. Previous studies have shown that simple popularity-based recommendation, even without class information, represents a strong baseline (Cremonesi et al. 2010; Jannach et al. 2015). Including the class information, which indicates the content that may interest users, may further boost the performance.

—Brand Class Top Popular (BrandClassTP): We observed that many users browse items of the same or similar brands in a session. Based on this observation, we computed the most popular items of each brand within the same category. BrandClassTP recognizes the brand and the class of the current browsing item and recommends the most popular items of the same brand within the category. Note that the products of the same brand may belong to different categories. For example, the shoes and the shirts made by Nike may belong to different categories.

—Association Rules (AR): We created the association rules between pairs of items based on their co-purchasing frequency and their co-occurrence frequency. Due to the nature of the algorithm, the generated number of rules is usually small. As a result, AR usually recommends far less than 10 items, sometimes even recommends nothing.

—Association Rules Plus ClassTP (AR-ClassTP): If AR alone recommends 10 or more items, AR-ClassTP returns the same recommendation list as AR. However, as explained above, AR usually recommends far less than 10 items. In this case, we added ClassTP's top recommended items to the end of AR's list to make the length of the recommendation list equals 10. This makes a fairer comparison.

—Item-based Collaborative Filtering on the Session-to-Item matrix (ICF-S2I): We created a session-to-item (S2I) matrix M in which the entry m_{ij} represents the number of times session i visits the item j. We calculated the cosine distance between every pair of columns to obtain the similarity scores between all pairs of products. When a user is browsing an item x, ICF-S2I recommends the 10 most similar products to the target item x.

—Item-based Collaborative Filtering on Item-to-Item matrix (ICF-I2I): We created an item-to-item (I2I) matrix N in which the entry n_{ij} represents the number of times item i and item j co-appear in a session. We calculated the cosine distance between every pair of rows to obtain the similarity scores between all pairs of products. When a user is browsing an item x, ICF-I2I recommends the 10 most similar products to the target item x.

—Non-negative Matrix Factorization on Session-to-Item matrix (NMF-S2I): We created a S2I matrix M as in the method ICF-S2I. We performed non-negative MF on M with rank $K = 300$ based on a variant of the alternative-least-square optimization. As a result, each item is eventually represented by a 300-dimensional dense vector. When a user is browsing an item x, NMF-S2I recommends the 10 most similar items based on the cosine similarity scores between the vector of the target item x and all the other items.

—Non-negative Matrix Factorization on Item-to-Item matrix (NMF-I2I): We created an I2I matrix N as in the method ICF-I2I. We performed non-negative MF on N with rank $K = 300$ based on a variant of the alternative-least-square optimization. When a user is browsing an item x, NMF-I2I recommends the 10 most similar items based on the cosine similarity scores between the vectors of the target item x to all the other items.

— Session-based RNN (S-RNN) (Hidasi et al. 2016): Like the Prod2Vec model, the S-RNN treats each session as one sentence and each clicked item as one word, and apply the RNN model to predict the next clicked item. As a result, this model does not need to compute the embedding of each product. However, this property makes S-RNN difficult to discover the relationship among products (which will be discussed in Section 4.2). We set the learning rate to 0.001 and dropout rate 0.1 to train the model. The hit rate drops as we increase the learning rate or the dropout rate.
— eALS (He et al. 2016): This method includes the unobserved clicks into the optimization model and weights the items based on their popularity. The intuition is that users are more likely to be aware of the popular items, and therefore if a popular item is never clicked, users are more probable to be uninterested to the item. Specifically, c_i, the variable that determines product i is a true negative instance to a user, is determined by $c_i = c_0 f_i^{\alpha} / (\sum_j f_j^{\alpha})$. We set α the popularity exponent variable to 0.5, as suggested in He et al. (2016), and c_0 the overall weight of the missing data to 512, based on a small varification set. The embedding size is set to 300, the same as the settings in the other baseline methods. We modified the implementation provided by the authors to perform the experiment.[2]
— FPMC (Rendle et al. 2010): This method generates the personalized Markov Chain transition graph between items based on the MF approach, because the observations to estimate the transition graph are usually sparse. As in the other baselines, we set the embedding size to 300 for all users, items in the previous step, and items in the current step. We modified the implementation from Li.[3]

We did not include the Recurrent Recommender Networks (RRN) (Wu et al. 2017) into the baselines. The RRN model is especially useful when an individual's browsing log is recorded for a long period of time, e.g., months to years. In our scenario, since we treat each "session" as one individual, the typical length is only minutes to hours long. For such a short period, the latent factors of users and items are unlikely to change dramatically. As a result, the RRN model has no obvious advantage.

4.1.4 Performance of Task 1. Here, we show the performance of the first prediction task introduced in Section 4.1.1 — predicting the next click item given the current viewing item. To protect business sensitivity information, we report the relative-hit-at-k, which divides each method's average-hit-at-k by ICF-S2I's average-hit-at-k.

Table 2 reports the result of the task 1 on the first test dataset. Prod2Vec, S-RNN, and FPMC performs better in terms of relative-hit-at-k for $k = 1, 2, \ldots, 10$. These models have similar performances, probably because they train the models based on continuous item clicks. The popularity-based methods (ClassTP and BrandClassTP), AR, and their combination (AR-ClassTP) yield similar performance when $k = 1$, but the performance of AR is clearly worse than the others when $k > 1$. This is because AR mostly recommends a few (much less than 10) items. The CF, MF, and their variations (ICF-I2I, ICF-S2I, NMF-I2I, and NMF-S2I) are popular techniques in predicting the rating of a user to an object (e.g., a movie or a song). However, previous study has shown that such techniques may not be a natural fit to the top-k recommendation task, because they focus on minimizing the error of the predicted ratings, which probably cannot be directly translated into the improvement of the precision of the top-k prediction task (Cremonesi et al. 2010). This is consistent

[2] https://github.com/hexiangnan/sigir16-eals.
[3] https://github.com/khesui/FPMC.

Table 2. The Relative-hit-at-k of Various Methods for the Task 1 (Predicting the Next Click Based on the Just Clicked Item) on the Test Data 1

k	1	2	3	4	5	6	7	8	9	10
Behavior2Vec	2.582	2.323	2.161	2.204	1.984	2.028	1.905	2.101	2.007	2.008
Prod2Vec	**2.607**	2.392	**2.286**	2.237	2.217	2.179	2.129	**2.115**	**2.101**	2.097
ClassTP	1.668	1.597	1.624	1.659	1.690	1.700	1.700	1.717	1.727	1.749
BrandClassTP	2.088	1.985	1.989	1.972	1.975	1.968	1.950	1.974	1.977	1.994
AR	1.553	1.396	0.912	0.893	0.877	0.837	0.808	0.785	0.755	0.741
AR-ClassTP	1.836	1.767	1.756	1.748	1.767	1.760	1.763	1.788	1.794	1.827
ICF-I2I	1.001	1.013	1.066	1.012	1.021	1.033	0.921	1.031	1.092	0.987
ICF-S2I	1	1	1	1	1	1	1	1	1	1
NMF-I2I	1.001	0.984	0.959	0.963	1.030	0.967	1.015	1.034	1.043	0.949
NMF-S2I	0.923	0.973	0.984	0.993	0.939	1.004	0.905	0.887	0.878	0.954
S-RNN	2.586	**2.393**	2.277	2.258	**2.225**	**2.188**	**2.150**	2.109	2.083	2.105
eALS	1.027	1.029	1.100	1.026	1.051	1.055	1.044	1.113	1.206	1.011
FPMC	2.585	2.353	2.243	**2.276**	2.118	2.083	2.035	2.114	2.080	**2.115**

The highest score of each k is highlighted in bold.

Table 3. The Relative-hit-at-k of Various Methods for the Task 1 (Predicting the Next Click Based on the Just Clicked Item) on the Test Data 2

k	1	2	3	4	5	6	7	8	9	10
Behavior2Vec	2.519	2.256	2.371	2.057	2.148	2.107	2.074	1.956	2.009	1.939
Prod2Vec	2.639	**2.417**	2.384	**2.304**	2.267	2.233	2.205	2.158	2.138	**2.125**
ClassTP	1.726	1.615	1.668	1.680	1.693	1.730	1.757	1.748	1.759	1.781
BrandClassTP	2.147	2.010	2.049	2.056	2.032	2.039	2.052	2.037	2.046	2.056
AR	1.650	1.457	1.371	1.293	1.210	1.144	1.094	1.043	0.998	0.967
AR-ClassTP	1.955	1.808	1.814	1.812	1.809	1.818	1.837	1.835	1.843	1.861
ICF-I2I	0.999	1.145	0.874	0.897	1.122	0.990	1.046	0.937	0.907	1.054
ICF-S2I	1	1	1	1	1	1	1	1	1	1
NMF-I2I	1.025	0.930	1.043	0.946	0.982	0.929	0.994	0.939	0.998	0.965
NMF-S2I	0.945	0.883	0.964	1.047	0.892	1.044	1.007	0.884	1.042	0.850
S-RNN	**2.649**	2.394	**2.388**	2.301	**2.287**	**2.254**	**2.213**	**2.167**	2.161	2.117
eALS	1.068	1.262	1.083	1.112	1.129	1.082	1.049	1.058	1.078	1.096
FPMC	2.570	2.390	2.337	2.267	2.172	2.227	2.185	2.140	**2.182**	2.100

The highest score of each k is highlighted in bold.

with our experiment result: these methods perform the worst compared to the other methods. The eALS model performs slightly better than the MF-based approaches, probably because it assigns weights to the unseen items based on popularity. Our proposed Behavior2Vec performs slightly worse than the strong baselines Prod2Vec, S-RNN, and FPMC. This is probably because Behavior2Vec tends to recommend the items the user may want to buy, but these items may not always be the items that may attract users to click immediately. However, if the task is to predict the next *purchase* item, as demonstrated later, Behavior2Vec tends to perform better.

We conducted the same experiment on the other two test datasets, as shown in Tables 3 and 4. The results are consistent with the first test dataset.

Table 4. The Relative-hit-at-k of Various Methods for the Task 1 (Predicting the Next Click Based on the Just Clicked Item) on the Test Data 3

k	1	2	3	4	5	6	7	8	9	10
Behavior2Vec	2.477	2.320	2.273	2.264	2.229	2.113	2.135	2.066	2.100	2.093
Prod2Vec	**2.569**	**2.431**	2.386	**2.330**	**2.290**	2.265	2.239	2.223	2.190	2.165
ClassTP	1.766	1.749	1.785	1.787	1.794	1.807	1.821	1.839	1.842	1.853
BrandClassTP	2.322	2.185	2.194	2.187	2.169	2.169	2.185	2.189	2.186	**2.180**
AR	1.711	1.529	1.462	1.368	1.285	1.228	1.176	1.125	1.073	1.033
AR-ClassTP	2.025	1.939	1.965	1.957	1.951	1.962	1.968	1.975	1.971	1.981
ICF-I2I	1.086	1.118	1.063	1.106	1.138	1.005	1.035	0.974	1.089	0.933
ICF-S2I	1	1	1	1	1	1	1	1	1	1
NMF-I2I	0.925	0.937	0.904	1.023	0.904	1.043	0.923	0.935	1.050	0.963
NMF-S2I	0.873	0.862	0.869	0.927	1.026	1.047	0.987	0.980	0.851	0.893
S-RNN	2.557	2.425	2.383	2.321	2.287	**2.282**	**2.247**	**2.246**	**2.205**	2.175
eALS	1.124	1.174	1.141	1.106	1.219	1.068	1.045	1.003	1.126	1.019
FPMC	2.532	2.386	**2.403**	2.245	2.223	2.213	2.231	2.187	2.169	2.115

The highest score of each k is highlighted in bold.

Table 5. The Relative-hit-at-k of Various Methods for the Task 2 (Predicting the Next Purchasing Item Based on the Just Clicked Item) on the Test Data 3

k	1	2	3	4	5	6	7	8	9	10
Behavior2Vec	**3.532**	**3.583**	**3.541**	**3.581**	**3.731**	**3.818**	**3.699**	**3.686**	**3.601**	**3.527**
Prod2Vec	3.331	3.362	3.398	3.305	3.374	3.369	3.347	3.344	3.247	3.221
ClassTP	2.359	2.362	2.345	2.280	2.278	2.250	2.187	2.178	2.080	2.060
BrandClassTP	2.369	2.365	2.350	2.285	2.282	2.254	2.196	2.187	2.085	2.066
AR	2.324	2.322	2.287	2.213	2.234	2.206	2.123	2.100	2.019	1.974
AR-ClassTP	3.214	3.210	3.177	3.079	3.101	3.105	3.023	3.000	2.853	2.792
ICF-I2I	0.745	0.754	0.742	0.720	0.715	0.712	0.688	0.683	0.654	0.650
ICF-S2I	1	1	1	1	1	1	1	1	1	1
NMF-I2I	0.861	0.869	0.859	0.848	0.847	0.844	0.835	0.832	0.826	0.822
NMF-S2I	0.863	0.873	0.870	0.849	0.844	0.855	0.836	0.842	0.820	0.822
S-RNN	3.318	3.337	3.399	3.310	3.373	3.362	3.314	3.324	3.262	3.229
eALS	1.105	1.065	1.074	1.137	1.001	1.004	0.993	1.138	1.140	1.160
FPMC	3.361	3.365	3.325	3.294	3.226	3.337	3.213	3.382	3.214	3.141

The highest score of each k is highlighted in bold.

4.1.5 Performance of Task 2. This section shows the performance of the second prediction task— predicting the next purchasing item given the user's current viewing item. We show only the relative-hit-at-k of the test data 3, since the results on all the three datasets are very similar.

As shown in Table 5, Behavior2Vec outperforms all the compared baselines, including the strong baselines — Prod2Vec, the other embedding-based method, S-RNN, a neural network-based recommendation strategy, and FPMC, a model to generate personalized transition graph. We believe this is because Behavior2Vec separates the viewing embedding and the purchasing embedding of every product and make recommendations based on the distance between the current behavior embedding and every item's purchasing embedding. As a result, Behavior2Vec tends to recommend the items the users may purchase next. On the other hand, since Prod2Vec, S-RNN, and FPMC do not

Table 6. The Relative-hit-at-k of Various Methods for the Task 3 (Predicting the Next Clicking Item Based on the Just Purchased Item) on the Test Data 3

k	1	2	3	4	5	6	7	8	9	10
Behavior2Vec	**4.532**	**4.693**	**5.124**	**5.022**	**5.020**	**5.309**	**5.571**	**5.366**	**5.201**	**5.210**
Prod2Vec	4.343	4.258	4.359	4.702	4.757	4.792	4.938	5.030	5.010	4.680
ClassTP	3.322	3.391	3.588	3.798	3.571	3.686	3.889	4.184	3.833	3.797
BrandClassTP	3.602	3.767	3.977	4.085	4.191	3.948	4.158	4.266	4.240	4.096
AR	3.795	4.080	4.249	4.012	4.000	3.985	3.966	3.847	3.715	3.244
AR-ClassTP	3.995	4.230	4.379	4.207	4.403	4.346	4.541	4.771	4.773	4.352
ICF-I2I	1.110	1.102	1.223	1.220	1.221	1.246	1.330	1.295	1.343	1.225
ICF-S2I	1	1	1	1	1	1	1	1	1	1
NMF-I2I	0.966	1.120	1.121	1.093	1.154	1.207	1.255	1.291	1.251	1.146
NMF-S2I	1.025	1.028	1.068	1.130	1.159	1.168	1.168	1.203	1.182	1.145
S-RNN	4.253	4.151	4.288	4.705	4.775	4.764	4.940	4.970	5.054	4.715
eALS	1.173	1.158	1.288	1.328	1.222	1.249	1.321	1.442	1.487	1.330
FPMC	4.436	4.267	4.135	4.668	4.312	4.695	4.514	5.150	4.906	4.455

The highest score of each k is highlighted in bold.

differentiate the purchasing and the clicking behaviors, different behaviors, which may be valuable clues in certain scenarios, do not play a role in the Prod2Vec model, S-RNN model, and the FPMC model.

The ranking of the performance of all the other baseline methods is similar to the results of Task 1. Overall, the popularity-based methods and the AR-based methods perform better than the CF- and MF-based methods.

4.1.6 Performance of Task 3. This section shows the result of task 3 — predicting the next clicking item given the items a user just bought.

Table 6 shows the relative-hit-at-k of various methods on the test dataset 3. Behavior2Vec outperforms all the compared baselines. As discussed in the last section, since Behavior2Vec considers both the viewing embedding and the purchasing embedding, it is reasonable that Behavior2Vec performs better than the other embedding method Prod2Vec, which recommends similar items without differentiating the viewing and purchasing behaviors. Probably by the same reason, Behavior2Vec performs better than the other strong baselines — the S-RNN model and the FPMC model.

Essentially, "similarity" is an ambiguous concept: two items could be similar because they are alternative to each other, one is the affiliated product of the other, or even some other reasons. As a result, we should probably recommend different types of "similar items" before and after purchasing an item. For example, after purchasing a camera body, a user is probably more interested in browsing the camera lenses that fit the body, but less likely to buy another camera body. However, before the purchase, it is reasonable to recommend another body when a user is browsing a camera body. Behavior2Vec can probably make better recommendation in these cases, but Prod2Vec, S-RNN, FPMC, and other baseline approaches may have difficulties in discriminating them.

We show only the results on test-3 because the experiment results on all the three test datasets are similar.

4.1.7 Performance of Task 4. Here, we show the experimental results of the task 4: given the item a user just bought, what will the user buy next?

Table 7. The Relative-hit-at-k of Various Methods for the Task 4 (Predicting the Next Purchasing Item Based on the Just Purchased Item) on the Test Data 3

k	1	2	3	4	5	6	7	8	9	10
Behavior2Vec	**4.455**	**4.682**	**4.556**	**4.694**	**5.062**	**5.253**	**4.732**	**4.685**	**4.441**	**4.707**
Prod2Vec	3.858	4.132	4.336	4.200	4.542	4.497	4.159	4.581	4.311	4.243
ClassTP	3.055	3.401	3.300	3.158	3.337	3.559	3.270	3.349	3.250	3.637
BrandClassTP	3.873	3.677	3.984	3.954	3.580	3.185	3.771	3.734	3.491	3.286
AR	3.416	3.613	3.504	3.369	3.519	3.851	3.559	3.222	3.111	3.008
AR-ClassTP	3.748	3.861	4.137	3.704	4.090	3.978	4.276	4.407	4.296	4.053
ICF-I2I	1.028	1.108	1.074	1.124	1.056	1.099	1.026	1.054	1.260	1
ICF-S2I	1	1	1	1	1	1	1	1	1	1
NMF-I2I	0.859	0.944	1.002	1.037	1.027	1.032	1.034	1.075	1.027	1.101
NMF-S2I	0.939	0.977	0.948	1.003	1.009	1.019	1.026	1.025	1.072	1.022
S-RNN	3.737	3.861	3.913	3.981	4.010	4.118	4.228	4.163	3.998	3.826
eALS	1.157	1.103	1.164	1.158	1.175	1.129	1.290	1.214	1.271	1.173
FPMC	4.034	3.882	3.716	4.180	4.151	4.252	3.788	3.858	4.242	3.783

The highest score of each k is highlighted in bold.

Table 7 shows the relative-hit-at-k of various methods on the test data 3. Like the previous results, Behavior2Vec outperforms all the baseline methods. We believe this is because Behavior2Vec returns the closest *purchasing embedding*, which represents the hidden vector of buying a certain product. As a result, Behavior2Vec is more likely to correctly predict the next purchasing item.

The rankings of the other methods are very close to the previous tasks. In general, Prod2Vec, S-RNN, and FPMC perform better among the baseline methods, followed by the popularity-based methods (ClassTP and BrandClassTP) and the AR-based methods (AR and ARClassTP), and the CF- and MF-based methods are even weaker.

Again, we show only the results on the dataset test-3, since all three test datasets yield similar results.

4.2 Product Analogy

This section presents case studies of the product analogy discovered by Behavior2Vec.

4.2.1 Motivation of Product Analogy Inference. A user who is currently browsing a product p_1 may click the next product p_2 due to different reasons. We list three possible situations below. First, the product p_1 and the product p_2 could be a substitution of each other (e.g., p_1 and p_2 are two cell phones made by different manufacturers). Second, the product p_2 could be an auxiliary product of the product p_1 (e.g., p_1 is a camera and p_2 is a tripod). Third, although p_1 or p_2 alone is useful, together they might be even more convenient (e.g., p_1 is a desk and p_2 is a chair). Although most recommendation algorithms can measure the distance between two products, they typically cannot identify how two products are related.

Since Word2Vec was shown to discover the semantic relationship among words (e.g., $King : Queen \approx Man : Woman$), we are interested to see whether we can discover the hidden relationship among the products based on the distributed representations of the products and the behaviors on these products. Among the baseline methods introduced in Section 4.1.3, only the Prod2Vec model generates the embedding. Thus, it is the only baseline approach we compared here.

4.2.2 Product Embedding Generating Based on Behavior2Vec. In Section 3, we introduced the methodology to generate the (output) viewing embedding $v_i^{(O)}$ and the (output) purchasing embedding $p_i^{(O)}$ of every product i. However, since we want to compare the relationship between products, we need to generate the embedding of every product instead of every behavior. To do so, we concatenate $v_i^{(O)}$ and $p_i^{(O)}$ to form a new vector $c_i^{(O)}$ (called the *concatenated behavior embedding*) and treat such a vector as the embedding of the product i

4.2.3 Experiment Data. We generated the distributed representations of the products based on a month-long log. We only report a few case studies because, unlike word analogy that has a manually compiled ground truth dataset for evaluation (Mikolov et al. 2013b), we cannot find a large dataset as the ground truth to evaluate the product analogy so far.

4.2.4 Result 1. Table 8 shows several interesting product analogies inferred based on the embedding. Overall, Behavior2Vec is very good at discovering the branding relationship, as shown in the cases 1, 2, 3, 5, 6, and 8. This is probably because users often browse products of the same or similar brands in a session, so we have enough training data to capture the brand relationship. Behavior2Vec also captures the substitution relationship (cases 2 and 5), auxiliary relationship (cases 1 and 3) and several other relationships among the products based on the concatenated behavior embedding. Sometimes, Behavior2Vec makes mistakes, as shown in the cases 4, 6, and 8. However, these cases still correctly capture certain relationship among the products.

4.2.5 Result 2. To compare the product embedding generated by Prod2Vec and the concatenated behavior embedding generated by Behavior2Vec, we visualize the embedding from the selected camera body to the corresponding kit lens from five camera brands (Canon, Nikon, Panasonic, Sony, and Pentax). We set the dimensionality of $v_i^{(O)}$, $p_i^{(O)}$, and Prod2Vec's embedding to 300-dimensional. Thus, the dimensionality of the concatenated behavior embedding $c_i^{(O)}$ becomes 600. We mapped the original high-dimensional product embedding vectors into 2-dimensional vectors by PCA. Figures 2 and 3 show the projected vector from camera body to the kit lens of the five brands for the concatenated behavior embedding and the Prod2Vec's embedding, respectively. It appears that, both embedding-based methods can, at least partially, capture the relationship between a camera body and a camera lens, since the vectors from the camera bodies to the kit lenses are similar on the five brands. However, our proposed Behavior2Vec appears to better capture such a relationship, compared to Prod2Vec.

5 DISCUSSION AND FUTURE WORK

In this article, we proposed Behavior2Vec — a model to generate the distributed representation (we called behavior embedding) for users' online behaviors on the products. Based on Behavior2Vec, we calculated the distance between the behaviors based on the cosine similarity between the embedding vectors. We found that, in nearly all our test datasets and parameter settings, Behavior2Vec better predicts users' next *purchasing* product, compared to several popular recommendation approaches. However, when the task is to predict the next *clicking* (viewing) product based on the current viewing product, strong baselines, such as Prod2Vec, S-RNN, and FPMC, perform slightly better. We believe this is because the Behavior2Vec model differentiates the embedding of *viewing* a product i and *purchasing* a product i, which are not considered in the Prod2Vec model, the S-RNN model, and the FPMC model.

Perhaps more importantly, we found that the product embedding (generated by either Prod2Vec or our Behavior2Vec) captures the hidden relationship between the products. The product relationship may help recommend products using different strategies under different scenarios. For

Table 8. A List of Discovered Interesting Product Analogy

No.	Analogy	Explanation
1	EOS 700D (a Canon camera): EF 24-105mm f/3.5-5.6 IS STM (a Canon lens) ≈ D7200 (a Nikon camera) : AF-S NIKKOR 20mm f/1.8G ED (a Nikon lens)	This reveals the auxiliary relationship and the brand relationship
2	Brand's Essence of chicken drink (6 jars): Brand's Essence of chicken drink (20 jars) ≈ May Flower's toilet paper (12 packs) : May Flower's toilet paper (24 packs)	This reveals the package size, the brand, and the substitution relationship
3	Brita kettle: Brita filter cartridge ≈ Toray faucet filter : Toray filter cartridge	This reveals the auxiliary relationship and the brand relationship
4	Zenfone 2: ZenPad 8.0 ≈ Galaxy phone J7 : ZenPad 8.0 Wifi	This is a mixture of success and failure. It reveals the relationship between a mobile phone and a tablet PC. However, it fails to capture the brand relationship (ASUS and Samsung)
5	ASUS AC1200: ASUS AC66U ≈ DLink AC750 : DLink DIR-619R	This reveals the substitution relationship and the brand relationship. All the four products are wireless routers
6	AB's men shirt: AB's men pants ≈ Fiore's women top 1 : Fiore's women top 2	This reveals the gender relationship and the brand relationship, but it fails to capture the relationship between a shirt and a pair of pants
7	Nikon P610 (a camera): TBC-405 (a camera case) ≈ LJ's grip wrench : LJ's spring compressor with a case	This partially captures that the latter should be a container of the former product
8	Lanew's men casual shoes: Lanew's men leather shoes ≈ Mom's women casual shoes: Mom's women sport shoes	It captures the gender and the brand relationship, but fails to capture the relationship between casual and formal

We translated the product names from Chinese to English and added a few explanations. We skipped the detailed description of the products.

example, we can choose to recommend substitute products before any purchase happens and recommend the affiliated products after any purchase. This is very different from the traditional recommendation strategies that typically recommend the most similar products to the current browsing product. Initial studies show that Behavior2Vec better captures the relationship between products, compared to the Prod2Vec model. This is probably because Behavior2Vec separates different behaviors during the training process and concatenates all the behavior embeddings on a given product as this product's embedding. As a result, the concatenated embedding may better separate different types of similar products, e.g., the alternative products or the affiliated products. The S-RNN model although can be modified to differentiate different behaviors during training, the output contains no product embedding. As a result, it is not straightforward to generate the product analogy by the S-RNN model.

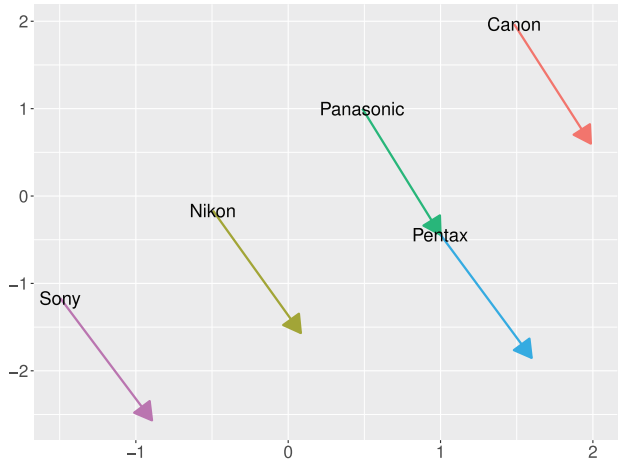

Fig. 2. The distributed representations generated by Behavior2Vec. Compared with the one generated by Prod2Vec (as shown in Figure 3), this seems to better capture the relationship between a camera body and the corresponding kit lens. The figure shows two-dimensional vectors (projected by PCA) from the camera body to the camera lens of five different brands. The original concatenated behavior embedding vectors are 600-dimensional (by concatenating the 300-dimensional viewing embedding $v_i^{(O)}$ and the 300-dimensional purchasing embedding $p_i^{(O)}$).

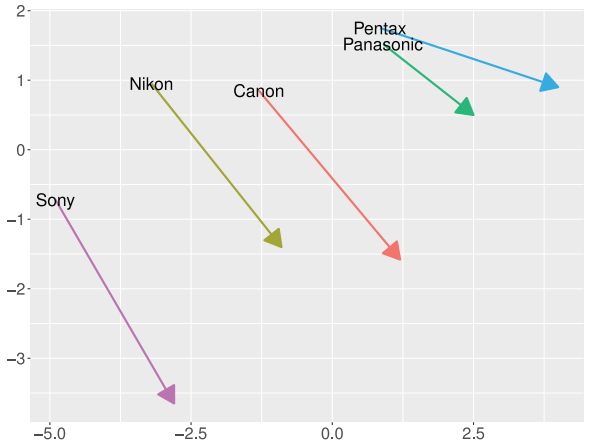

Fig. 3. The distributed representations generated by Prod2Vec. The figure shows two-dimensional vectors (projected by PCA) from the camera body to the camera lens of the five different brands. The original product embedding vectors are 300-dimensional.

The current model recommends the product whose corresponding behavior embedding is most close to the current behavior. The assumption behind such a mechanism is that the distributed representation of the current behavior is a good representation of a user's intension. However, if we can discover the distributed representation of several recent behaviors or the entire behaviors in the session, we can probably better predict the user's intension. This is also one of the research directions we are interested to continue.

ACKNOWLEDGMENTS

We are grateful to the National Center for High-performance Computing for computer time and facilities.

REFERENCES

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. 238–247.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.* 3 (2003), 1137–1155.

John S. Breese, David Heckerman, and Carl Kadie. 1998. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, 43–52.

John A. Bullinaria and Joseph P. Levy. 2012. Extracting semantic representations from word co-occurrence statistics: Stop-lists, stemming, and SVD. *Behav. Res. Methods* 44, 3 (2012), 890–907.

Sin-Wai Chan. 2014. *Routledge Encyclopedia of Translation Technology*. Routledge.

Hung-Hsuan Chen, Chu-An Chung, Hsin-Chien Huang, and Wen Tsui. 2017. Common pitfalls in training and evaluating recommender systems. *ACM SIGKDD Explor. Newsl.* 19, 1 (2017), 37–45.

Hung-Hsuan Chen, Liang Gou, Xiaolong Zhang, and Clyde Lee Giles. 2011. CollabSeer: A search engine for collaboration discovery. In *Proceedings of the 11th Annual International ACM/IEEE Joint Conference on Digital Libraries*. ACM, 231–240.

Hung-Hsuan Chen, Alexander G. Ororbia II, and C. Lee Giles. 2015. ExpertSeer: A keyphrase based expert recommender for digital libraries. arXiv:1511.02058.

Hung-Hsuan Chen, Pucktada Treeratpituk, Prasenjit Mitra, and C. Lee Giles. 2013. CSSeer: An expert recommendation system based on CiteseerX. In *Proceedings of the 13th ACM/IEEE-CS Joint Conference on Digital Libraries*. ACM, 381–382.

Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the 4th ACM Conference on Recommender Systems*. ACM, 39–46.

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*. ACM, 1809–1818.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*. 173–182.

Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 549–558.

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

Geoffrey Hinton, J. L. McClelland, and D. E. Rumelhart. 1986. Distributed representations. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*.

Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining*. IEEE, 263–272.

Wenyi Huang, Zhaohui Wu, Prasenjit Mitra, and C. Lee Giles. 2014. RefSeer: A citation recommendation system. In *Proceedings of the 14th ACM/IEEE-CS Joint Conference on Digital Libraries*. IEEE Press, 371–374.

Dietmar Jannach, Lukas Lerche, and Michael Jugovac. 2015. Adaptation and evaluation of recommendations for short-term shopping goals. In *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 211–218.

Yehuda Koren. 2008. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 426–434.

Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. arXiv:1405.4053.

Rémi Lebret and Ronan Collobert. 2013. Word emdeddings through Hellinger PCA. arXiv:1312.5542.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*. Montréal, Canada, 2177–2185.

Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing* 7, 1 (2003), 76–80.

Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. 2010. Personalized news recommendation based on click behavior. In *Proceedings of the 15th International Conference on Intelligent User Interfaces*. ACM, Hong Kong, China, 31–40.

Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. 2011. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*. Springer, 73–105.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. arXiv:1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, vol. 2. 3111–3119.

Tomas Mikolov, Wen-Tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (HLT-NAACL'12)*. 746–751.

Marco Pennacchiotti and Siva Gurumurthy. 2011. Investigating topic models for social media user recommendation. In *Proceedings of the 20th International Conference Companion on World Wide Web*. ACM, 101–102.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, vol. 14. 1532–1543.

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*. ACM, 811–820.

Xin Rong. 2014. Word2vec parameter learning explained. arXiv:1411.2738.

Ruslan Salakhutdinov and Andriy Mnih. 2008. Bayesian probabilistic matrix factorization using Markov Chain Monte Carlo. In *Proceedings of the 25th International Conference on Machine Learning*. ACM, 880–887.

Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th International Conference on Machine Learning*. ACM, 791–798.

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*. ACM, 285–295.

Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. 2006. *Introduction to Data Mining*, vol. 1. Pearson Addison Wesley, Boston, MA.

Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnetminer: Extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 990–998.

Hong-Jian Xue, Xin-Yu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep matrix factorization models for recommender systems. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17)*. 3203–3209.

Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J. Smola, and How Jing. 2017. Recurrent recommender networks. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining*. ACM, 495–503.

Received June 2017; revised January 2018; accepted January 2018