Product Collection Recommendation in Online Retail

Pigi Kouki
pigi.kouki@relational.ai
relational AI

Ilias Fountalis
ilias.fountalis@relational.ai
relational AI

Nikolaos Vasiloglou
nik.vasiloglou@relational.ai
relational AI

Nian Yan
nian_yan@homedepot.com
The Home Depot

Unaiza Ahsan
unaiza_ahsan@homedepot.com
The Home Depot

Khalifeh Al Jadda
khalifeh_al_jadda@homedepot.com
The Home Depot

Huiming Qu
huiming_qu@homedepot.com
The Home Depot

Abstract

Recommender systems are an integral part of eCommerce services, helping to optimize revenue and user satisfaction. Bundle recommendation has recently gained attention by the research community since behavioral data supports that users often buy more than one product in a single transaction. In most cases, bundle recommendations are of the form "users who bought product A also bought products B, C, and D". Although such recommendations can be useful, there is no guarantee that products A, B, C, and D may actually be related to each other. In this paper, we address the problem of *collection recommendation*, i.e., recommending a collection of products that share a common theme and can potentially be purchased together in a single transaction. We extend on traditional approaches that use mostly transactional data by incorporating both domain knowledge from product suppliers in the form of hierarchies, as well as textual attributes from the products. Our approach starts by combining product hierarchies together with transactional data or domain knowledge to identify candidate sets of product collections. Then, it generates the product collection recommendations from these candidate sets by learning a deep similarity model that leverages textual attributes. Experimental evaluation on real data from the Home Depot online retailer shows that the proposed solution can recommend collections of products with increased accuracy when compared to expert-crafted collections.

CCS Concepts

• **Information systems → Collaborative filtering**.

Keywords

online product recommendation; bundle recommendation; text embeddings for recommender systems; product taxonomies

ACM Reference Format:
Pigi Kouki, Ilias Fountalis, Nikolaos Vasiloglou, Nian Yan, Unaiza Ahsan, Khalifeh Al Jadda, and Huiming Qu. 2019. Product Collection Recommendation in Online Retail. In *Thirteenth ACM Conference on Recommender Systems (RecSys '19), September 16–20, 2019, Copenhagen, Denmark*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3298689.3347003

RecSys '19, September 16–20, 2019, Copenhagen, Denmark

ACM ISBN 978-1-4503-6243-6/19/09…$15.00
https://doi.org/10.1145/3298689.3347003

1 Introduction

Recommender systems in eCommerce aim at solving a wide set of problems, from revenue maximization [11], to next product [18], and complementary product recommendations [17]. The problem is usually formalized as identifying one *individual* product to recommend given the user's context. However, our analysis on behavioral data from a large eCommerce retailer shows that users buy more than one product in a single transaction. More specifically, analysis of online transactional data of the Home Depot[1] showed that 28% of the transactions involve more than one product purchase and these transactions account for 56% of the total product purchases.[2] As a result, bundle recommendation, i.e., recommending a *set* of products that can *all* be purchased together, is of benefit to both sellers and buyers [19]. Traditionally, research in bundle recommendation [2, 6, 19] focuses on using transactional data to come up with a set of products that may be purchased together (i.e., co-purchased), without any constraint around how relevant the recommended products are to each other. Although this approach works well in a variety of scenarios, it does not address the cases where users are looking to buy a set of products having a common theme. As an example, consider a user interested in remodelling her bathroom. In this case, the user may need a sink faucet, a showerhead, and a towel bar that have similar style to match each other. In most cases, the users would try to mix and match their desired products manually.

In this work, we focus on generating bundles of products that can be bought in a single transaction, where the products fit well together in terms of style, and share the same high-level functionality, but satisfy different needs (i.e., they are not interchangeable). We express such needs through a set of constraints. We refer to the problem of bundling products that fit together as the *collection recommendation*. Figure 1 shows an example of five products that constitute a collection. All products in this collection share the same style (e.g., they are all chrome color) and can all be bought in the same transaction. Additionally, all products constitute solutions for bathroom, however, each product serves a different need.

To solve the collection problem we use a combination of transactional data, content information for products, and domain knowledge. To ensure that products share the same high level functionality but are not interchangeable, we use product hierarchies. We propose two methods to identify product hierarchies containing products that may potentially constitute a collection. In the first method we use domain knowledge, while in the second we use

[1] homedepot.com
[2] Statistics computed from approximately 28M online transactions involving 46M product purchases between Jan 1, 2018 and March 31, 2019.

transactional data. Next, we use product relationships that were provided by domain experts, to learn a deep similarity model that identifies whether two products belong to the same collection given their content information. We evaluate our approach using real data coming from the Home Depot online retailer. Our contributions are: (1) a formal definition of the collection problem (Sec. 3), (2) a method that combines transactional data, domain knowledge, and textual embeddings to generate collections of products (Sec. 5), (3) evaluation by expert annotators that showcase that the proposed solution improves expert-crafted collections (Sec. 6).

2 Related Work

Beladev et al. [3] produce bundles by combining collaborative filtering with demand functions and price modeling. Zhu et al. [19] generate bundles from existing recommendations that maximize a reward function based on user data. Bai et al. [2] focus on personalized bundles by modeling diversity with co-purchase information. Qi et al. [12] recommend packages with validity constraints to groups of users. All these works leverage user purchase-history. In our setting there is no such information available. Garfinkel et al. [6] optimize for the cost of the total purchase, while we optimize for the relevance among the collection products. Image-based approaches have been proposed for the fashion domain [5, 8, 9, 13] while we operate in the home improvement domain.

3 Problem Definition

For a given set of products $\mathbf{P} = \{p_1, ..., p_k\}$ offered by an online retailer, we assume that each product p_i is associated with a set of attributes $\mathbf{A_i} = \{a_{i_1}, ..., a_{i_l}\}$. Examples of attributes are *title*, *description*, *brand*, *color*. The online retailer uses a taxonomy T to organize all the products in its catalog. A taxonomy consists of a set of categories and relationships among those. In detail, we are given a set of categories $\mathbf{C} = \{C_1, ..., C_m\}$ that are organized into hierarchies using a set of *subclass* relationships $\mathbf{R} = \{r(C_i, C_j), \forall C_i, C_j \in \mathbf{C}, i < j\}$. For example, $r(C_i, C_j)$ means that C_j is subclass of C_i. A hierarchy is an ordered set of categories that share *subclass* relationships, i.e., $h = (C_1, C_2, ..., C_{k-1}, C_k)$ where $r(C_i, C_{i+1}) \in \mathbf{R}$, $i = 1...k$. C_1 is defined as the *root* category. One example hierarchy may be (*Bath,Bathroom Faucets, Bathroom Sink Faucets, Single Handle Faucets*). Each product p_i is assigned to exactly one hierarchy. The set of all distinct hierarchies is defined as $\mathbf{H}$.

We define the collection problem as follows: Given an anchor product p_a, our goal is to generate a set of *collection* products $\mathbf{P_c} = \{p_{c_1}, ..., p_{c_k}\}$ such that the set of products $\{p_a, p_{c_1}, ..., p_{c_k}\}$ with the respective hierarchies $\mathbf{H_c} = \{h_a, h_{c_1}, ..., h_{c_k}\}$ are desirable to the users. For example, a user looking to renovate her bathroom will be looking at matching faucets for the sink and the bathtub. The set of products $\mathbf{P_c}$ may have to satisfy a set of constraints that describe what collections of products are acceptable to present to the users. These constraints are usually imposed by the business logic. For example, an online retailer may require all products to be of the same brand or may require specific items bundled together. In our case, we are interested in the following constraints: (1) all the products in $\mathbf{P_c}$ need to have similar or matching values for specific attributes with the anchor product p_a such that they match the style of the anchor product. In our setting we define that all products in the same collection need to have the same color and relevant title and description, i.e., $\forall p_i \in \mathbf{P_c} : color_{p_i} = color_{p_a}, title_{p_i} \approx title_{p_a}$, $description_{p_i} \approx description_{p_a}$. (2) The hierarchies of all products in the collection need to share the same root category, which also needs to be the same as the root category C_1 of the anchor product p_a, i.e., $\forall h_i, h_j \in \mathbf{H_c} : C_{i_1} = C_{j_1}$. (3) The set of hierarchies that the collection products are assigned to, $\mathbf{H_c}$, cannot contain duplicate hierarchies, i.e., $\forall h_i, h_j \in \mathbf{H_c} : h_i \neq h_j$. The goal is to form a set of products that does not contain interchangeable products. Although these constraints are specific to our scenario, our overall approach is generalizable to including other constraints as well.

4 Collections from Domain Experts

Online retailers sell a variety of products from a large number of product suppliers who typically have domain experts that manually curate information for each product. Available information can range from attributes such as color or description, to relationships between products. Typically, each supplier m provides a set of products $\mathbf{P_m} = \{p_{m_1}, ..., p_{m_q}\}$ along with a set of bidirectional relationships among products of the form: $\mathbf{R_m} = \{r_m(p_{m_i}, p_{m_j}), ..., r_m(p_{m_k}, p_{m_l})\}$. A relationship $r_m(p_{m_i}, p_{m_j})$ indicates that the products p_{m_i} and p_{m_j} share the same style. Given such relationships, one can create collections of products simply by extracting all relationships r_m that involve an anchor product p_a.

Generating collections using information from experts is straightforward. However, it is expensive and not scalable. When a new product is introduced, experts need to generate relationships between this product and others. In practice, product suppliers provide relationships only for a small fraction of their products. For example, out of approximately 2 million products available by Home Depot online, only 34.2% have at least one collection relationship. Additionally, domain experts are aware only of products from a specific supplier and, as a result, they cannot provide relationships between products from different suppliers. An online retailer that sells products from a variety of suppliers needs to *automatically* generate collections for *all* products available in its catalog.

5 Proposed Approach

We generate collections of products sharing the same theme in two steps. First, we leverage the hierarchy information to generate product candidate sets that potentially belong to the same collection. To this end, we find (a) hierarchies that often appear together in the relationships provided by product suppliers and (b) hierarchies whose products are often co-purchased. Next, we leverage product information to determine which products from the candidate sets belong to the same collection. In detail, we use relationships provided by domain experts along with the product description and title to learn a deep similarity model that generates product embeddings. We present our approach in detail in the next sections.

5.1 Generating Candidate Sets of Collection Products

We can generate candidate sets of collection products by using domain knowledge as well as transactional data.

5.1.1 Leveraging domain knowledge . We retrieve all products provided by all domain experts $\mathbf{P_m}$ and for each product $p_{m_i} \in \mathbf{P_m}$ we retrieve its assigned h_i. We transform each relationship between products $r_m(p_{m_i}, p_{m_j}) \in \mathbf{R_m}$ to a relationship between hierarchies, $q_m(h_i, h_j)$, where p_{m_i} and p_{m_j} are assigned to hierarchies h_i and h_j respectively. This process produces a *multiset* of relationships between hierarchies: $\mathbf{Q_m} = \{q_m(h_i, h_j), ..., q_m(h_k, h_l)\}$

Figure 1: An Example of a Collection of Products

where a given relationship can exist more than once. Next, we compute the probability that each pair of hierarchies (h_i, h_j) co-occurs as: $Prob_m(h_i, h_j) = \frac{N(q_m(h_i,h_j))}{N(h_i)}$, where $N(q_m(h_i, h_j))$ is the frequency of the pair (h_i, j_j) in the multiset $\mathbf{Q_m}$ and $N(h_i)$ is the frequency of the hierarchy h_i in $\mathbf{Q_m}$. We compute the $Prob_m$ only for pairs of hierarchies that share the same root category.

5.1.2 Leveraging transactional data . We retrieve all online transactions with at least two products purchased in a specific period of time (e.g., one year). We construct a set of all products purchased: $\mathbf{P_p} = \{p_{p_1}, ..., p_{p_q}\}$. Then, we generate a multiset of relationships between products, $\mathbf{R_p} = \{r_p(p_{p_i}, p_{p_j}), ..., r_p(p_{p_k}, p_{p_l})\}$. A relationship $r_p(p_{p_i}, p_{p_j})$ exists iff p_i and p_j are purchased in the same transaction (co-purchased). $\mathbf{R_p}$ can contain the same pair of products more than once. As with the previous approach, we retrieve the hierarchy h_i of each product $p_{p_i} \in \mathbf{P_p}$ and transform each relationship $r_p(p_{p_i}, p_{p_j})$ indicating co-purchase of products, to a relationship $q_p(h_i, h_j)$ that indicates co-purchase of hierarchies. Again, we end up with a *multiset* of relationships between hierarchies: $\mathbf{Q_p} = \{q_p(h_i, h_j), ..., q_p(h_k, h_l)\}$. For pairs that share the same root category only, we compute the probability that (h_i, h_j) is co-purchased as: $Prob_p(h_i, h_j) = \frac{N(q_p(h_i,h_j))}{N(h_i)}$, where $N(q_p(h_i, h_j))$ is the frequency of the pair (h_i, j_j) in the multiset $\mathbf{Q_p}$ and $N(h_i)$ is the frequency of the hierarchy h_i in $\mathbf{Q_p}$.

5.2 Leveraging Text Embeddings

Given the hierarchies with candidate sets of collection products, we next learn a model that identifies products that indeed belong to the same collection. For training data, we use relationships from product suppliers ensuring that the products share the same style.[3] For our model, we use a deep similarity network based on Long Short Term Memory (LSTM) networks. LSTMs are a particular class of recurrent neural networks and have proved very effective in tasks related to natural language processing [1, 7, 15]. Bidirectional LSTMs (biLSTM) [14] are an extension of LSTM networks where the input sequence is read both from left to right and from right to left from two separate LSTMs. The output of the combined model is the concatenated output of the two LSTMs. We generate text embeddings for products using a siamese biLSTM architecture [4, 10]. Siamese networks are trained to discriminate between the class identity of input pairs. A siamese network consists of input pairs x_i, x_j that are passed through a deep network F. The embedding outputs $F(x_i), F(x_j)$ are joined by a metric function. Next, we introduce a cosine similarity layer. In particular, we use the cosine similarity $cos(F(x_i), F(x_j)) = \frac{F(x_i)\cdot F(x_j)}{||F(x_i)||||F(x_j)||}$ as our metric function combined with a sigmoid activation. The sigmoid activation maps the cosine similarity to [0,1] making the cross-entropy a natural choice for our loss function.

In our work, we generate text embeddings for products using the product description and title, i.e., the siamese biLSTM architecture consists of two biLSTMs, one for product description and one for product title. The hidden states of the two biLSTMs are concatenated and then are used as an input to the cosine similarity layer. Siamese networks are trained to discriminate between pairs of products in the same collection versus pairs of products belonging to different collections. As discussed, we use information from domain experts to generate training data: we regard that two products are similar if there exists a relationship of the form $r_m(p_{m_i}, p_{m_j})$. For each pair of similar products, we generate z pairs of dissimilar products by randomly sampling products $p_{n_j}, n \neq m$. Once we train the network, we use the concatenated output of the two LSTMs to generate product embeddings.

5.3 Generating Collections of Products

For a given anchor product p_a, we generate a collection by first retrieving its hierarchy h_a. Using either the method described in Sec. 5.1.1 or 5.1.2, we compute the probabilities between h_a and all hierarchies that share the same root category. We rank the probabilities from highest to lowest and retrieve the top-N hierarchies. The N hierarchies along with the h_a form the set $\mathbf{H_c}$ described in the problem definition. For each hierarchy $h_i \in \mathbf{H_c} (i \neq a)$ we retrieve all products $\mathbf{P_{h_i}}$ assigned to this hierarchy and we consider this to be the candidate set. Next, we retrieve the text embeddings for the anchor product p_a and each product $p_i \in \mathbf{P_{h_i}}$ (as computed in Sec. 5.2). For each $p_i \in \mathbf{P_{h_i}}$, where $color_{p_a} = color_{p_i}$, we compute the cosine similarity $cos(F(p_a), F(p_i))$. Finally, we rank all the similarities and we add to the set $\mathbf{P_c}$ the product p_i that ranked at the top in terms of the $cos(F(p_a), F(p_i))$ similarity.

6 Experimental Validation

In this section, we compare the performance of the proposed methods to generate collections, to the manually generated collections from domain experts. We evaluate the performance of the following methods: (1) **DomExp**: the manually created collections from the relationships provided by domain experts (Sec. 4). This is the baseline that we compare against. (2) **DomExpEmb**: generate collections by computing pairs of hierarchies that frequently appear in domain experts' relationships (Sec. 5.1.1) and text embeddings (Sec. 5.2). (3) **CopEmb**: generate collections by computing pairs of hierarchies that are frequently co-purchased (Sec. 5.1.2) and text embeddings (Sec. 5.2).

[3] Using transactional data here would not be effective because products that are co-purchased do not necessarily share the same style.

	Bath	Patio
Method	**Accuracy (SD)**	**Accuracy (SD)**
DomExp	0.51 (0.32)	0.76 (0.23)
DomExpEmb	0.75 (0.15)	**0.83** (0.24)
CopEmb	**0.82** (0.13)	**0.83** (0.24)

Table 1: Average accuracy of different methods in two datasets. Numbers in parenthesis indicate standard deviations. Bold shows the best accuracy score for each dataset.

6.1 Setup

We evaluate our approach using two datasets from the Home Depot. The first dataset contains products from the *Bath* root category while the second products from the *Patio* root category. We use 487 anchor products from the *Bath* and 369 from the *Patio*. Each anchor product has 5 collection relationships from domain experts. For the proposed approach, we also set $N = 5$ (i.e., we generate 5 recommendations for each anchor). We used 129, 060 relationships from domain experts for *Bath* and 23, 433 from *Patio* as our training data. We minimally processed the raw text by removing stop words, converting to lower case, and applying sentence padding. We truncate product descriptions to 100 words, and product titles to 20 words. For the training of the two LSTMs (one for description and one for title) we set the input word-embedding dimensions to 128. Each of the two LSTMs has a hidden size of 256 units. Dropout and recurrent dropout are used for regularization (dropout rate is set to 0.5). We train the proposed network for 20 epochs using RMSProp [16] as an optimizer with a batch size of 128.

For each of the 487 *Bath* anchor products (369 for *Patio*) we generate three collections using the three methods described in the previous section. This results in 1, 461 for *Bath* (1, 107 for *Patio*) different collections that need to be evaluated (each collection consists of 5 product recommendations). We evaluate the quality of the collections using human experts, i.e., the in-house validation team of Home Depot. We gave validators specific instructions about what constitutes a collection and what does not, i.e., we described in layman's terms the constraints described in the definition (Sec. 3). For each anchor product, we asked validators to label each of the 5 product recommendations as relevant or not relevant. To avoid order-related biases, we randomized the presentation order of the different collections.

6.2 Results and Discussion

For each anchor product we compute the *accuracy* of the recommendation as the fraction of relevant recommendations to the total number of recommendations (fixed at 5). For each method, we average the accuracy score for all the 487 anchor products for *Bath* (369 for *Patio*). For each dataset, we report the average results for each method along with the standard deviation in Table 1. For both datasets, we observe that both proposed methods **DomExpEmb** and **CopEmb** outperform the manually-created collections (**DomExp**). Performing an unpaired t-test shows that the difference between the **CopEmb** and **DomExp** is considered extremely statistically significant ($p < 0.0001$). This is also true when comparing method **DomExpEmb** to method **DomExp**. For *Bath* only, between the two proposed approaches (**DomExpEmb** and **CopEmb**), we observe that **CopEmb** outperforms **DomExpEmb** and this difference is considered extremely statistically significant ($p < 0.0001$). For *Patio*, methods **CopEmb** and **DomExpEmb** perform the same.

It is surprising that the manually created collections performed significantly worse compared to both automated solutions. After analyzing those collections we found that the set of relationships R_m violate the third constraint, i.e., given a relationship $r_m(p_{m_i}, p_{m_j})$ it may be the case that p_{m_i} and p_{m_j} are assigned to the same hierarchy. This significantly decreases the accuracy of the provided collections. Additionally, as discussed in Sec. 4, domain experts generate collections $\mathbf{P_c}$ with all products coming from the same supplier (i.e., same brand). However, analysis on real transactional data from Home Depot, shows that customers do not shop this way. Figure 2 shows the loyalty to five popular brands computed from 99, 754 transactions that involved exactly 4 products from the *Bath* root category for the year of 2018.[4] The X axis shows the number of products that belong to the same brand, while the Y axis shows the percentage of transactions. For example, for *BrandA* (blue bar) 72% of the transactions involved 1 product from the same brand *BrandA* (all other products were of different brand). We observe similar trends when increasing the total number of products purchased in a transaction. This observation is another reason that explains why the automated collections outperform the manually created ones.

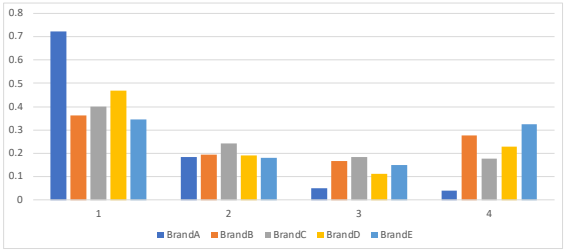

Figure 2: Brand loyalty statistics. The Y axis shows the percentage of products from the same brand for the 5 most popular brands in the *Bath* root category.

6.3 Evaluation of Product Collections Without Domain Expert Information

As discussed, manually created collections do not scale. In contrast, the methods proposed here are able to generate collections for all products offered in the catalog. To illustrate this, we generated and evaluated collection recommendations for anchor products that do not have relationships from the domain experts. We generated collection recommendations for 500 anchor products from the *Bath* root category and for 290 products from the *Patio*. As before, we generated 5 collection products for each anchor and we followed the exact same process to evaluate the collections. The average accuracy is **0.87** (SD=**0.17**) for *Bath* and **0.77** (SD=**0.28**) for *Patio*. Thus, we conclude that the proposed method can be generalized to products without information from domain experts.

7 Conclusions and Future Work

In this paper, we proposed a recommendation algorithm to generate collections of products that share the same theme. Our method used co-purchase information or domain knowledge to generate the candidate sets of collection products and then product attributes with domain expert information to learn product similarities. Our future work includes adding image similarities to the model, evaluating the approach using A/B testing, and studying the effect of a "bad" recommendation on the total collection recommendation.

[4]We anonymized the brand names to maintain the confidentiality of the suppliers.

References

[1] D. Bahdanau, K. Cho, and Y. Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[2] J. Bai, Chang Z., Junshuai S., Xiaoru Q., Weiting A., Zhao L., and Jun G. 2019. Personalized Bundle List Recommendation. In *WWW'19*.

[3] M. Beladev, L. Rokach, and B. Shapira. 2016. Recommender systems for product bundling. *Knowledge-Based Systems* 111 (2016).

[4] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. 2014. Signature verification using a "siamese" time delay neural network. In *NIPS'94*.

[5] Shankar D., Narumanchi S., Ananya H., Kompalli P., and Chaudhury K. 2017. Deep Learning based Large Scale Visual Recommendation and Search for E-Commerce. *CoRR* abs/1703.02344 (2017).

[6] R. Garfinkel, R. Gopal, A. Tripathi, and F. Yin. 2006. Design of a Shopbot and Recommender System for Bundle Purchases. *Decision Support Systems* 42, 3 (2006).

[7] A. Graves. 2012. Supervised sequence labelling. Springer.

[8] X. Han, Z. Wu, Y. Jiang, and L. Davis. 2017. Learning Fashion Compatibility with Bidirectional LSTMs. In *MM'17*.

[9] S. Jaradat. 2017. Deep Cross-Domain Fashion Recommendation. In *RecSys '17*.

[10] G. Koch, R. Zemel, and R. Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*.

[11] W. Lu, S. Chen, Keqian Li, and Laks V. S. L. 2014. Show Me the Money: Dynamic Recommendations for Revenue Maximization. *Proceedings of the VLDB Endowment* 7, 14 (2014).

[12] S. Qi, N. Mamoulis, E. Pitoura, and P. Tsaparas. 2018. Recommending Packages with Validity Constraints to Groups of Users. *Knowledge Information Systems* 54, 2 (2018).

[13] Yin R., Li K., Lu J., and Zhang G. 2019. Enhancing Fashion Recommendation with Visual Compatibility Relationship. In *WWW'19*.

[14] M. Schuster and K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45, 11 (1997).

[15] I. Sutskever, O. Vinyals, and Q. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS'14*.

[16] T. Tieleman and G. Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *Coursera: Neural networks for machine learning* 4, 2 (2012).

[17] Z. Wang, Z. Jiang, Z. Ren, J. Tang, and D. Yin. [n.d.]. A Path-constrained Framework for Discriminating Substitutable and Complementary Products in E-commerce. In *WSDM '18*.

[18] M. Zhou, Z. Ding, J. Tang, and D. Yin. [n.d.]. Micro Behaviors: A New Perspective in E-commerce Recommender Systems. In *WSDM '18*.

[19] T. Zhu, P. Harrington, J. Li, and L. Tang. 2014. Bundle Recommendation in Ecommerce. In *SIGIR'14*.