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  
huming_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:


Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

RecSys ’19, September 16–20, 2019, Copenhagen, Denmark
© 2019 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-6245-6/19/09. $15.00
https://doi.org/10.1145/3298689.3347003

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

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 Depot1 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
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 the users. These constraints are usually imposed by the business describe what collections of products are acceptable to present to the users. For example, a user looking to renovate her bathroom ships, i.e., For example, Qarchies, i.e., VH,1, H,1 ∈ H, C,1 = C, hj. (3) The set of hierarchies that the collection products are assigned to, H, cannot contain duplicate hierarchies, i.e., VH,1, H,1 ∈ H, C,1 = H,1. 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 Pm = {p1, ..., pm} along with a set of bidirectional 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. Relationships between 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.

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. We retrieve all products provided by all domain experts Pm and for each product pm, we retrieve its assigned hj. We transform each relationship between products rm(pm, pm) ∈ Rm to a relationship between hierarchies, qm(hj, hj), where pm, and pm, are assigned to hierarchies hj and hj respectively. This process produces a multiset of relationships between hierarchies: Qm = {qm(hj, hj), ..., qm(hj, hj)}
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:

\[
\text{Prob}_m(h_i, h_j) = \frac{N(q_m(h_i, h_j))}{N(h_i)},
\]

where \(N(q_m(h_i, h_j))\) is the frequency of the pair \((h_i, h_j)\) in the multiset \(Q_m\) and \(N(h_i)\) is the frequency of the hierarchy \(h_i\) in \(Q_m\). We compute the \(\text{Prob}_m\) only for pairs of hierarchies that share the same root category.

5.1.2 Leverage 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: \(P_p = \{p_1, ..., p_l\}\). Then, we generate a multiset of relationships between products, \(P_r = \{r_1(p_1, p_2), ..., r_r(p_l, p_l)\}\). A relationship \(r_i(p_j, p_k)\) exists if \(p_j\) and \(p_k\) are purchased in the same transaction (co-purchased). \(P_r\) can contain the same pair of products more than once. As with the previous approach, we retrieve the hierarchy \(h_i\) of each product \(p_i \in P_p\) and transform each relationship \(r_i(p_j, p_k)\) 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: \(Q_r = \{q_p(h_i, h_j), ..., q_p(h_k, h_l)\}\). For pairs that share the same root category only, we compute the probability that \((h_i, h_j)\) is co-purchased as:

\[
\text{Prob}_r(h_i, h_j) = \frac{N(q_p(h_i, h_j))}{N(h_i)},
\]

where \(N(q_p(h_i, h_j))\) is the frequency of the pair \((h_i, h_j)\) in the multiset \(Q_r\) and \(N(h_i)\) is the frequency of the hierarchy \(h_i\) in \(Q_r\).

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. 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 \(r_m(p_n, p_m)\). For each pair of similar products, we generate \(z\) pairs of dissimilar products by randomly sampling products \(p_n, n \neq m\). 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 the 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 \(H_a\) described in the problem definition. For each hierarchy \(h_i \in H_a\) we compute the relationships between \(h_i\) and the anchor product \(p_a\) and each product \(p_i \in P_h\) (as computed in Sec. 5.2). For each \(p_i \in P_h\), where \(\text{color}_{p_a} = \text{color}_{p_i}\), we compute the cosine similarity \(\cos(F(p_a), F(p_i))\). Finally, we rank all the similarities and add to the set \(P_t\) 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. 5.1.2) 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).
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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Bath</th>
<th>Patio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (SD)</td>
<td>0.75 (0.15)</td>
<td>0.83 (0.24)</td>
</tr>
<tr>
<td>CopEmb</td>
<td>0.82 (0.13)</td>
<td>0.83 (0.24)</td>
</tr>
<tr>
<td>DomExpEmb</td>
<td>0.76 (0.23)</td>
<td>0.76 (0.23)</td>
</tr>
</tbody>
</table>

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 (1,07 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 \( r^m(p_m, p_m) \) it may be the case that \( p_m \) and \( p_m \) 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 \( P_e \) 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. 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.](image-url)

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.

---

We anonymized the brand names to maintain the confidentiality of the suppliers.
References


