A Path-constrained Framework for Discriminating Substitutable and Complementary Products in E-commerce

Zihan Wang*†
Data Science Lab, JD.com
zihanwang.jd@gmail.com

Ziheng Jiang†
Data Science Lab, JD.com
ziheng.j@gmail.com

Zhaochun Ren
Data Science Lab, JD.com
renzhaochun@jd.com

Jiliang Tang
Michigan State University
tangjili@msu.edu

Dawei Yin‡
Data Science Lab, JD.com
yindawei@acm.org

ABSTRACT

In personalized recommendation, *candidate generation* plays an infrastructural role by retrieving candidates out of billions of items. During this process, *substitutes* and *complements* constitute two main classes of retrieved candidates: substitutable products are interchangeable, whereas complementary products might be purchased together by users. Discriminating *substitutable* and *complementary* products is playing an increasingly important role in e-commerce portals by affecting the performance of candidate generation, e.g., when a user has browsed a t-shirt, it is reasonable to retrieve similar t-shirts, i.e., substitutes; whereas if the user has already purchased one, it would be better to retrieve trousers, hats or shoes, as complements of t-shirts.

In this paper, we propose a path-constrained framework (**PMSC**) for discriminating substitutes and complements. Specifically, for each product, we first learn its embedding representations in a general semantic space. Thereafter, we project the embedding vectors into two separate spaces via a novel mapping function. In the end, we incorporate each embedding with path-constraints to further boost the discriminative ability of the model. Extensive experiments conducted on two e-commerce datasets show the effectiveness of our proposed method.

ACM Reference format:
Zihan Wang, Ziheng Jiang, Zhaochun Ren, Jiliang Tang, and Dawei Yin. 2018. A Path-constrained Framework for Discriminating Substitutable and Complementary Products in E-commerce. In *Proceedings of WSDM 2018: The Eleventh ACM International Conference on Web Search and Data Mining, Marina Del Rey, CA, USA, February 5–9, 2018 (WSDM 2018),* 9 pages.
https://doi.org/10.1145/3159652.3159710

*Work performed during an internship at Data Science Lab, JD.com.
†These two authors contributed equally to the paper.
‡Corresponding author

WSDM 2018, February 5–9, 2018, Marina Del Rey, CA, USA

ACM ISBN 978-1-4503-5581-0/18/02…$15.00
https://doi.org/10.1145/3159652.3159710

1 INTRODUCTION

Personalized recommender systems (PRS) have become indispensable and ubiquitous for bridging users and items in various scenarios, e.g., multimedia recommendations [6, 7, 11, 18], news recommendations [37], and product recommendations [36], etc. In modern e-commerce portals, although billions of items exist, only a small portion of them attract users' attention. Modern e-commerce PRS usually consist of two procedures, i.e., *candidate generation* and *candidate ranking* [4, 6]: candidate generation refers to a process that retrieves personalized candidates out of billions of items [6, 17, 39]; whereas candidate ranking procedure ranks the retrieved candidates for a better conversion rate or click-through rate [4, 12, 29, 35, 40]. Thus given a user's contextual information, e-commerce PRS can be optimized by addressing these two aspects, respectively. In this paper, we address the *candidate generation* problem in e-commerce PRS.

In general, there are several strategies for retrieving personalized product candidates, among which substitutable and complementary products are of the most importance. *Substitutable* products are those that are interchangeable, while *complementary* products are those that might be purchased together [23]. Understanding substitutable and complementary products can help candidate generation. For example, when a customer is shopping for a t-shirt online, it is better to retrieve similar t-shirts, or jeans, shoes, and socks etc., rather than totally irrelevant items such as computers or movies. More importantly, apart from being general retrieval sources, substitutable and complementary products can also be used as the unique source in some specific recommendation scenarios, such as "similar products" and "recommend after purchasing", respectively.

Recently, a system named Sceptre [23] is proposed to distinguish substitutes and complements. Given two datasets called *also-view* and *also-bought* from Amazon, Sceptre is able to model and predict substitutes and complements relations by learning latent topics from textual information, such as products reviews and descriptions etc. Despite the success of Sceptre, several issues still have not been fully addressed yet: 1) Severe sparseness problem: Based on our *also-view* and *also-bought* datasets from Amazon and JD, we find there are 57.2 millions of products, with only 2.24 billion pairs. This implies that existing models may not be well trained under e-commerce circumstance, especially for those tail products in e-commerce portals. 2) Neglect of complex paths: Direct links are a kind of paths that can naturally encode products and relations as

graph, where nodes are products and edges are relations. In addition to that, more complex paths exist in such a product graph, which is a sequence of relations and products, *e.g.* $i \xrightarrow{subst.} j \xrightarrow{compl.} k$. These paths represent more complex relation dependencies, which are able to further improve the performance of candidate generation.

In this paper, we propose a path-constrained method to discriminate substitutes and complements (**PMSC**). Specifically, we first predict whether there is a directed link between any pair of products, which is achieved by learning product embedding vectors in a general semantic space. Then, we define a latent vector for each relation and map the original product representations into the corresponding relation space, which can discriminate what type the directed link is, e.g. substitute or complement. Moreover, we incorporate product category constraints and multi-step path constraints to alleviate the data sparsity problem.

We conduct extensive experiments on two real-world datasets from two e-commerce portals. Experimental results show that **PMSC** significantly outperforms the baselines in various tasks in e-commerce. The embeddings of existing products can be easily generalized for new products, which helps mitigate the "cold-start" problem. The learned embeddings are visualized to understand how the proposed algorithm works. The main contributions of this paper are as follows:

- We propose a novel multi-relation learning framework, **PMSC**, which identifies substitutes and complements simultaneously.
- Two types of path constraints are incorporated, which can substantially alleviate the data sparsity problem. The visualization of the learned representations clearly shows that with path constraints, **PMSC** can effectively distinguish substitutes and complements.
- Experiments on two large-scale real world datasets show that **PMSC** significantly outperforms the baselines in various tasks in e-commerce.

2 RELATED WORK

In this section, we detail relevant previous work on collaborative filtering and neural embedding models.

2.1 Collaborative Filtering

The input of a recommendation system is often the users' demography, behaviors, and context, while the output is the relevant products. In recent years, collaborative filtering (CF) based techniques have received increased attention [10, 16, 30, 38, 41]. Unlike content-based filtering strategies [19] which predict ratings through the analysis of user profiles, collaborative filtering [31] methods, either memory-based CF or model-based CF, predict ratings using user-item ratings matrices. Early CF-based methods apply memory-based techniques. The most widely used memory-based CF methods include the nearest neighbor approach [3], user-based methods [27] and item-based methods [28]. Among the model-based CF methods, latent factor models [13] have become very popular as they show state-of-the-art performance on multiple datasets. Aimed at factorizing a rating matrix into products of a user-specific matrix and an item-specific matrix, matrix factorization based methods [13, 15] are widely used. Zhang et al. [38] propose a localized matrix factorization approach to address data sparsity and scalability by factorizing block diagonal form matrices. Ranking-oriented collaborative filtering algorithms have achieved good results: Shi et al. [30] use a list-wise learning to rank method, called CliMF. Linden et al. [17] predict user behavior links such as "Users who viewed X also viewed Y" or "Users who bought X also bought Y" . These "co-counting" methods are simple and may also produce a noisy recommendation for the rare products and has limited ability to explain results. More browsing and co-purchasing methods are proposed in [39]. Recently, He et al. [10] successfully apply neural networks to collaborative filtering. Unlike all above collaborative filtering methods, this paper is to learn the semantics of related products.

2.2 Neural Embedding Models

Recently, neural embedding methods for recommendation system and linguistic tasks have significantly improved accuracy of state-of-art models [1, 2, 5, 20, 24, 25] . In the neural embedding method, words, phrases or items are mapped into a low dimension space in order to capture semantics and correlation. For instances, motivated by word2vec [24], Barkan and Koenigstein [2] propose a modified version of the skip-gram model with negative sampling, item2vec, so as to induce a similarity measure for different items. Moreover, many recent works learn representations for relation inference with other features [9, 22, 23, 32–34]. For example, He et al. [9] learn complicated and heterogeneous relationships between items with high-level visual features.

The most similar work as ours is Sceptre [23], which learns the semantics of the substitutes and complements from data associated with products, such as the text of product reviews, ratings, specifications. However, Sceptre neglects the relations between categories and the paths of relations between two products. Our proposed method takes these two factors into consideration and formulates them as path constraints to obtain the further improvement.

3 THE PROPOSED FRAMEWORK

3.1 An Overview

Before we give a detailed description of our method, a high-level overview is presented as follows. There are four main phases: (A) directed link prediction; (B) multi-relation learning; (C) path constraints; (D) joint learning and optimization. Table 1 describes the notation we use throughout the paper.

To model plausibility of every edge between products, in phase (A), we first embed products into a general semantic space, and because of the asymmetry of relations, we utilize two types of vectors, i.e., target vector v and context vector v', to predict directed links between two products. This idea is closely related to the model of Item2vec [2]. To enable the scalability, we adopt a logistic loss function to judge the existence of the directed link between products. On top of link prediction using general semantic vectors, in phase (B) we introduce a latent vector for each relation and map the original representations into relation-specific spaces to discriminate edge types. In this way, **PMSC** enables implicit data sharing among related products and relations. Due to the data

Table 1: Glossary

Symbols	Descriptions
v_i, v'_i	target,context vector of product i
$v_{r,i}, v'_{r,i}$	projected target ,context vector of product i for r
r	product relation
β_r	projecting vector for relation r
$\mathcal{R}$	product relation set
$\mathcal{E}$	product set
$\mathcal{E}_P, \overline{\mathcal{E}}_P$	positive, negative ordered pair set
$\mathcal{E}_E, \overline{\mathcal{E}}_E$	positive, negative edge set
$\mathcal{E}_F, \overline{\mathcal{E}}_F$	positive, negative formula set
V, V'	target, context representation set
$\boldsymbol{\beta}$	projecting vector set
f, f'	positive, negative formula
N	number of negative samples

sparseness problem, embedding vectors may not be trained substantially and many products from very different categories may share indistinguishable representations. To address this problem and boost the discriminative ability of product representations, in phase (C) we incorporate two types of path constraints into the proposed model:1) The meta information such as product category is an important supplement to identify the relations between products, especially for tail products. 2) Most previous models pay much attention on learning relational data using the direct links, while multi-step paths is able to imply much more complicated patterns among the relations.

3.2 (A) Directed Link Prediction

In fact, neither substitute relation nor complement relation is necessarily symmetric. Similar to the method in [2], we use two types of vectors, *i.e.* target and context vectors, which capture the different semantics for products in directed relations. Specifically, if product i is a good substitute or a reasonable complement for products j, *i.e.* there is a substitute or complement edge from product i to product j, the probability whether this edge exists ($y_{i,j} = 1$) or not ($y_{i,j} = 0$) is calculated as:

$$P(y_{i,j}|V, V') = \sigma(v_i^T \cdot v'_j), \tag{1}$$

where v_i is the target vector of product i and v'_j is the context vector of product j. $\sigma(\cdot)$ denotes the sigmoid function. It is easy to verify that the larger $p(y_{i,j}|V, V', \boldsymbol{\beta})$ is, the more likely the edge from i to j exists.

Then, we adopt the logistic loss, which leads to the following objective function:

$$P(Y|V, V') = \prod_{(i,j)\in\mathcal{E}_P} \sigma(v_i^T \cdot v'_j)) \cdot \prod_{(i,j')\in\overline{\mathcal{E}}_P} (1 - \sigma(v_i^T \cdot v'_{j'})), \tag{2}$$

where $\mathcal{E}_P$ is the positive ordered pair set, meaning that if $(i, j) \in \mathcal{E}_P$, there exists a directed edge from product i to product j. $\overline{\mathcal{E}}_P$ is the negative ordered pair set constructed by randomly replacing the products in the corresponding positive pairs.

Since it is impractical to enumerate all the negative pairs in the training process, we adopt the efficient negative sampling strategy proposed in [24] and construct the negative instances by replacing j in the positive pair (i, j) with j' so that $(i, j') \in \overline{\mathcal{E}}_P$. Therefore, the loss function can be written as:

$$\begin{aligned} L(Y|V, V') &= -log(P(Y|V, V')) \\ &= -\sum_{(i,j)\in\mathcal{E}_P} log(\sigma(v_i^T \cdot v'_j)) \\ &\quad - N \cdot E_{j'\sim P_n(w)}[log(\sigma(-v_i^T \cdot v'_{j'}))], \end{aligned} \tag{3}$$

where N is the number of negative samples, and $P_n(w)$ is the noise distribution which is often set to the 3/4rd power of the unigram distribution $U(w) \sim 1/|\{(i, i') \in \overline{\mathcal{E}}_P\}|$. The goal of the objective function is to distinguish the positive product pairs from the noise distribution $P_n(w)$.

3.3 (B) Multi-Relation Learning

In this section, we generalize our model for the multi-relation task. Usually, when a product has multiple semantics, various relations focus on different semantics of products accordingly. We learn the product representations by projecting them from the general semantic space to corresponding relation space as follow:

$$v_{r,i} = v_i + \beta_r \odot v_i, v'_{r,i} = v'_i + \beta_r \odot v'_i, \tag{4}$$

where $v_{r,i}$ is the projected target vector for relation r and $v'_{r,i}$ is the projected context vector for relation r and β_r denotes the project vector. The difference of the product representations and projected vectors will be determined by both products and relations. In projecting operation, every entry of product representation interacts sufficiently with the relation representation in the corresponding position. In that way, we define the confidence of the directed edge (i, r, j) as $I(i, r, j)$ and calculate it as follow:

$$I(i, r, j) = P(z_{i,j,r}) = \sigma(v_{r,i}^T \cdot v'_{r,j}), \tag{5}$$

where $z_{i,j,r}$ is the label of the edge (i, r, j). The objective function for relation identification can be rewritten as:

$$\begin{aligned} L(Z|V, V', \boldsymbol{\beta}) &= -log(P(Z|V, V', \boldsymbol{\beta})) \\ &= \sum_{r\in R} \sum_{(i,r,j)\in\mathcal{E}_E} -log(I(i, r, j)) \\ &\quad - N \cdot E_{j'\sim P_n(w_0)}[log(1 - I(i, r, j'))], \end{aligned} \tag{6}$$

where $\mathcal{E}_E$ is the positive edge set. We also adopt the negative sampling strategy and $P_n(w_0) \sim 1/|\{(i, r, j') \in \overline{\mathcal{E}}_E\}|^{3/4}$, in which $\overline{\mathcal{E}}_E$ is the negative edge set.

3.4 (C) Path Constraints for Substitutes and Complements

To model path constraints in product dataset, t-norm fuzzy logics are adopted. We take a directed edge (i, r, j) as an atomic formula f_0. In that case, the probability of a complex formula f, *i.e.* $I(f)$, can be obtained by a composition of the probability of atomic formulae through specific t-norm based lattices and conjunctions in logic. We follow [8] and use the product t-norm. The compositions containing logical conjunction ($\wedge$), disjunction ($\vee$), and negation ($\neg$) are defined as follow:

$$\begin{aligned} I(f_a \wedge f_b) &= I(f_a) \cdot I(f_b), \\ I(f_a \vee f_b) &= I(f_a) + I(f_b) - I(f_a) \cdot I(f_b), \\ I(\neg f_a) &= 1 - I(f_a), \end{aligned}$$

Figure 1: An Example for Category Constraints.

where f_a and f_b are two constituent formulae, either atomic or complex. Given these compositions, the confidence value of any complex formula can be calculated recursively, e.g.,

$$I(\neg f_a \wedge f_b) = I(f_b) - I(f_a) \cdot I(f_b),$$
$$I(f_a \Rightarrow f_b) = I(f_a) \cdot I(f_b) - I(f_a) + 1,$$
$$I(f_a \wedge f_b \Rightarrow f_c) = I(f_a) \cdot I(f_b) \cdot I(f_c) - I(f_a) \cdot I(f_b) + 1.$$

Product Category Constraints. Most existing data-driven approaches often suffer from the data sparsity problem in practice. To address this problem, we enhance the product representation distinctiveness by fusing information from two sources (basic structure information, *i.e.* direct links, and category feature) into a unified framework. Specifically, we take the following constraint into consideration:

$$\begin{aligned} & (Prod_A, RelatedTo, Prod_B) \\ \Rightarrow \quad & (Category_A, RelatedTo, Category_B). \end{aligned} \tag{7}$$

Here $Category_A$ and $Category_B$ are respectively the product categories of $Prod_A$ and $Prod_B$. The constraint indicates that when product A and product B are related, the categories of product A and product B probably share semantic information and their representations are much closer in latent semantic space than other irrelevant product categories. For instance, in Figure 1, t-shirt and jeans are closely related to each other and so are their categories. To directly combine Eq. (7) with the original model, We consider the following constraints:

$$(Prod_A, Subst, Prod_B) \Rightarrow (Prod_A, Subst, Prod_C), \tag{8}$$

$$(Prod_A, Compl, Prod_B) \Rightarrow (Prod_A, Compl, Prod_C). \tag{9}$$

Note that $prod_C$ is in the same product category as $prod_B$. In this way, our method brings more interactions among representations of related products, which is beneficial for representation learning. Given an instance of the constraints $f_1 \triangleq (i, r_1, j) \Rightarrow (i, r_1, k)$, the confidence is obtained by:

$$\begin{aligned} I(f_1) = & I(i, r_1, j) \cdot I(i, r_1, k) \\ & - I(i, r_1, j) + 1. \end{aligned} \tag{10}$$

Multi-Step Path Constraints. In addition to the category constraints, multi-step paths between product pairs can reflect much more complex patterns than the direct links. In the graph, a multi-step path is a sequence of nodes and relations. For instance, as in Figure 2(a), a t-shirt is a substitute to a polo shirt and both of them

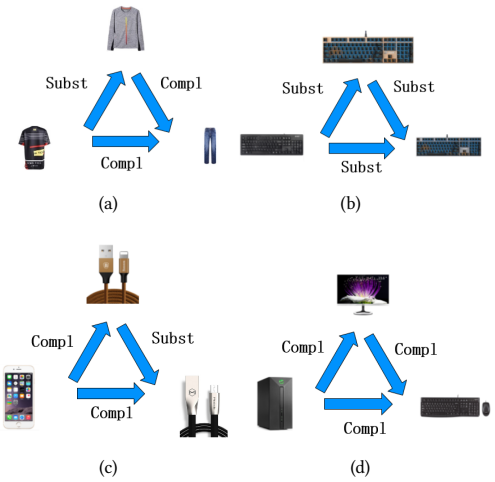


Figure 2: Examples for Multi-Step Path Constraints

are complements to jeans (similar ideas are shown in Figure 2). We count the frequency of instances discovered by all the reasonable two-hop relations in the Amazon dataset, and find out that four patterns, shown in Table 2, have dominated the data distribution. Therefore, to capture such information, multi-step path constraints is designed and incorporated. The form of the multi-step path constraints is $\forall i, j, k : (i, r_1, j) \wedge (j, r_2, k) \Rightarrow (i, r_3, k)$. In the product graph, we can generally represent the constraints as the following:

$$\begin{aligned} & (Prod_A, Subst, Prod_B) \wedge (Prod_B, Subst, Prod_C) \\ & \Rightarrow (Prod_A, Subst, Prod_C), \end{aligned} \tag{11}$$

$$\begin{aligned} & (Prod_A, Compl, Prod_B) \wedge (Prod_B, Subst, Prod_C) \\ & \Rightarrow (Prod_A, Compl, Prod_C), \end{aligned} \tag{12}$$

$$\begin{aligned} & (Prod_A, Subst, Prod_B) \wedge (Prod_B, Compl, Prod_C) \\ & \Rightarrow (Prod_A, Compl, Prod_C), \end{aligned} \tag{13}$$

$$\begin{aligned} & (Prod_A, Compl, Prod_B) \wedge (Prod_B, Compl, Prod_C) \\ & \Rightarrow (Prod_A, Compl, Prod_C). \end{aligned} \tag{14}$$

Given an instance of constraints $f_2 \triangleq (i, r_1, j) \wedge (j, r_2, k) \Rightarrow (i, r_3, k)$, the confidence is obtained by:

$$\begin{aligned} I(f_2) = & I(i, r_1, j) \cdot I(j, r_2, k) \cdot I(i, r_3, k) \\ & - I(i, r_1, j) \cdot I(j, r_2, k) + 1. \end{aligned} \tag{15}$$

Note that the larger the confidence is, the better the ground rules are satisfied. Besides the rules mentioned, our framework is able to handle more general rules and paths with longer length. We might explore more types of rules in the future.

We then incorporate path constraints into our model using Eq.(10) and (15), and the negative log-likelihood loss function for the path constraint i can be rewritten as follow:

$$\begin{aligned} L(F_c | V, V', \boldsymbol{\beta}) = & - \sum\nolimits_{f_c \in \mathcal{E}_F} log(I(f_c)) \\ & - N \cdot E_{f'_c \sim P_n(w_c)}[log(1 - I(f'_c))], \end{aligned} \tag{16}$$

where F_c is the label set for the formula f_c, where f_0 represents direct edges, f_1 is a formula for the product category constraint and f_2 is a formula for the multi-step path constraint. $\mathcal{E}_F$ is the positive formula set for the formula f_c (an atomic formula or a complex

Table 2: Multi-Step Path Ratio

Multi-Step Path	Ratio	
(r_1, r_2, r_3)	Electronics	Women's Clothes
(Sub, Sub, Sub)	22.84%	66.18%
(Sub, Com, Com)	65.75%	13.64%
(Com, Sub, Com)	4.17%	7.64%
(Com, Com, Com)	3.70%	6.83%
$OtherRules$	3.54%	5.71%

formula) and $\overline{\mathcal{E}_F}$ is the negative one. The noise distribution here is $[1/|\{f'_c \in \overline{\mathcal{E}_F}\}|]^{3/4}$.

3.5 (D) Joint Learning and Optimization

After representing directed links and path constraints as atomic and complex formulae, we combine Eq. (3) and (16) and minimize a global loss to learn product and relation representations as follow:

$$\begin{aligned} \min L_{joint} &= L(Y|V, V') + \sum_{c=0}^{2} \alpha_c \cdot L(F_c|V, V', \boldsymbol{\beta}) \\ &= L(Y|V, V') + \alpha_0 \cdot L(Z|V, V', \boldsymbol{\beta}) + \alpha_1 \cdot L(F_1|V, V', \boldsymbol{\beta}) \\ &+ \alpha_2 \cdot L(F_2|V, V', \boldsymbol{\beta}), \end{aligned} \quad (17)$$

$$\text{s.t.} \quad \|\mathbf{v_i}\|_2 \leq 1 \quad and \quad \left\|\mathbf{v'_i}\right\|_2 \leq 1, \forall i \in \mathcal{E}; \quad \|\mathbf{r}\|_2 \leq 1, \forall r \in \mathcal{R}.$$

Before the optimization process, a training set $\mathcal{E}_F$, which contains all positive path constraints found by breadth-first search, is constructed. Next, we construct the negative set as follow: if $f \triangleq (i, r, j)$ is an observed edge in the product graph, we construct f' by replacing j with a random product j'. If $f \triangleq (i, r_1, j) \Rightarrow (i, r_1, k)$ is a positive formula for product category constraint, a possible negative instance is f' by replacing r_1 with other relation, or replacing i or k with other product. If $f \triangleq (i, r_1, j) \wedge (j, r_2, k) \Rightarrow (i, r_3, k)$ is a positive formula for multi-step path constraint, a possible negative instance is f' by replacing r_3 with other relation, or replacing i or k with other product. We filter out randomly generated formula f' appeared in the training set. Then, to optimize the loss function, stochastic gradient descent is used, where α_c is the weight for formula f_c. To satisfy the ℓ_2-constraints, $\mathbf{v}$,$\mathbf{v}'$ and $\mathbf{r}$ are projected to the unit ℓ_2-ball in every iteration. Product and relation feature vectors trained in this way are expected to fit both the directed links and path constraints. The detailed optimization procedure is described in Algorithm 1.

Algorithm 1 Learning **PMSC**

Input: target and context representation set, V and V'; Projecting vector set, $\boldsymbol{\beta}$; formula set (atomic and complex formulas), $\mathcal{E}_F = \{f\}$; product pair set, $\mathcal{E}_P = \{(i, j)\}$; size of batch, b; the number of samples, N.

initialize: $\mathbf{v} \leftarrow$ uniform $(-0.5, 0.5)$ for each target vector $\mathbf{v} \in V$ initialize context vector $\mathbf{v}'$ and projecting vector β_p with $\mathbf{0}$
loop
  Normalize $\mathbf{v}, \mathbf{v}', \beta_p$ with $\mathbf{v}/||\mathbf{v}||, \mathbf{v}'/||\mathbf{v}'||, \beta_p/||\beta_p||$
  Sample $\mathcal{E}_{P_{batch}}, \mathcal{E}_{F_{batch}}$ of size b from $\mathcal{E}_P, \mathcal{E}_F$
  $\overline{\mathcal{E}}_{P_{batch}}, \overline{\mathcal{E}}_{F_{batch}} \leftarrow \emptyset$
  for $f \in \mathcal{E}_{F_{batch}}$ and $(i, j) \in \mathcal{E}_{P_{batch}}$ **do**
    $k = 0$
    for $k < N$ **do**
      $f'_k \leftarrow$ Sample from $\overline{\mathcal{E}}_F$
      $(i_k, j'_k) \leftarrow$ Sample from $\overline{\mathcal{E}}_P$
      $\overline{\mathcal{E}}_{F_{batch}} \leftarrow \overline{\mathcal{E}}_{F_{batch}} \cup f'_k$
      $\overline{\mathcal{E}}_{batch} \leftarrow \overline{\mathcal{E}}_{batch} \cup (i_k, j'_k)$
    end for
    $\overline{\mathcal{E}}_{F_{batch}} \leftarrow \overline{\mathcal{E}}_{F_{batch}} \cup f$
    $\overline{\mathcal{E}}_{P_{batch}} \leftarrow \overline{\mathcal{E}}_{P_{batch}} \cup (i, j)$
  end for
  Updating target vector v, context vector v' and projecting vector β_p w.r.t. $\nabla L_{joint}(\mathcal{E}_{P_{batch}}, \overline{\mathcal{E}}_{P_{batch}}, \mathcal{E}_{F_{batch}}, \overline{\mathcal{E}}_{F_{batch}})$
end loop

4 EXPERIMENTS

In this section, we first describe our datasets in §4.1. The parameters for the proposed framework are listed in §4.2. The baselines and evaluation metrics are detailed in §4.3. We evaluate **PMSC** in terms of link prediction and ranking performances in §4.4. Then we examine the impact of the cold start problem in §4.5.

4.1 Datasets

We use two real-world e-commerce datasets in our experiments:
Amazon dataset: Amazon dataset was collected on the user-product-review graph and was split into top-level categories, e.g., electronics, women's clothes, office product, etc. In this dataset, we select 5 categories from the Amazon dataset released by McAuley et al. [23], whereas four types of product pairs (x, y) are crawled to obtain ground truth of substitutes and complements, i.e., "also viewed" (users who viewed x also viewed y), "buy after viewing" (users who viewed x eventually bought y), "also bought" (users who bought x also bought y), and "bought together" (users who frequently bought x and y together). Edges of "also viewed" and "buy after viewing" are considered as substitutes and edges of "frequently buy together" and "also bought" are treated as complements. The detailed statistics of the dataset are shown in Table 3

Table 3: Statistics on Amazon dataset

Category	Item	Review	Edge
Electronic	498K	11.4M	7.87M
Women's Clothes	838K	14.5M	17.5M
Home and Kitchen	437K	4.25M	12.9M
Cell Phones and Accessories	347K	3.44M	6.22M
Office Products	134K	1.24M	4.20M
All	2.26M	34.9M	48.7M

JD dataset: we randomly sample edges of "also viewed" and "also bought" from logs of a real e-commerce site, **JD.com**. Because these edges form the majority of the dataset, we ignore the other two types of edges. The detailed statistics of the dataset are shown in Table 4.

4.2 Experimental Settings

We split our data into 80% for training, 10% for validation and 10% for test. Since it is impractical to train our model on all the negative product pairs, we randomly sample N non-links from the dataset.

Table 4: Statistics on JD dataset

Category	Item	Review	Edge
Baby	707K	3.90M	35.2M
Women's Clothes	1.06M	4.20M	35.5M
Men's Clothes	734K	3.65M	18.1M
Cell Phones and Accessories	581K	3.32M	39.7M
Office Products	382K	2.88M	46.2M
All	34.7M	18.0M	175M

Given each positive instance (i, r, j), we generate N negative instances using the method proposed in [23]. We tune the dimension of our representations in $\{50, 100, 200\}$, the number of negative samples N in $\{5, 10, 20\}$, weights α_0, α_1 and α_2 in $\{0.001, 0.01, 0.1, 1.0\}$, and learning rate in $\{0.1, 0.01, 0.001\}$.

4.3 Baselines and Evaluation Metrics

We compare PMSC to several baseline methods, including Random, Matrix Factorization (MF)[13], Non-Negative Matrix Factorization (NMF)[14], and Sceptre[23].

Random. The link prediction probability $P(i, r, j)$ is replaced by random number between 0 and 1. When link prediction probability is larger than a threshold α, the edge (i, r, j) holds. The threshold α is selected on validation set.

Matrix Factorization (MF) and Non-Negative Matrix Factorization (NMF). Matrix factorization [13] and non-negative matrix factorization [14] are both developed for finding linear representation of data. In experiments, we set the entry (i, j) of original matrix as 1 for directed edge $prod_i \rightarrow prod_j$ and 0 for non-links. We train individual model for each relation. We tune the dimension of the features in $\{200, 500, 1000, 2000, 5000\}$.

Item-to-Item Collaborative Filtering (CF). Item-to-Item Collaborative Filtering is the solution for *Amazon* recommender system reported in [17]. In our case, we evaluate the method with sets of users who review each item instead of browsing and purchasing data. Then we compute the cosine similarity between products by calculating the cosine similarity between user sets. We only test this method in ranking tasks, since it is not probabilistic.

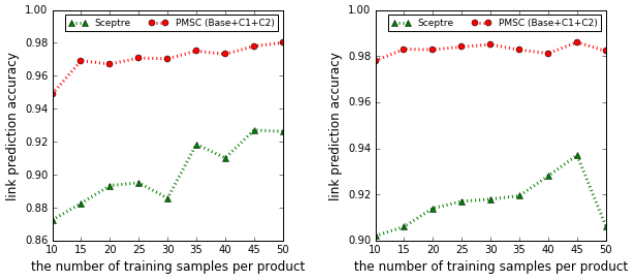
Sceptre is proposed in [23]. The key idea of Sceptre is to combine topic modeling and supervised link prediction. In addition, Sceptre can harness additional features (such as brands, price, and rating) and predict multiple types of relations simultaneously. In the link prediction and ranking task, we follow the same experimental settings in [23]. We tune the number of products needed for a new topic in $\{200, 500, 1000, 2000, 5000\}$.

Besides, to specify performance of each component of our model, we also train **PMSC (base)** according to Eq. (3) and Eq. (6), **PMSC (base+C1)** using Eq. (17) with only product category constraints and **PMSC (base+C1+C2)** with both product category and multi-step path constraints in Eq. (17).

4.4 Link Prediction and Ranking

Link Prediction. To predict whether there is a directed edge r between the given product pair (i, j), we optimize our objective function in Eq. (17). A prediction is correct when:

- if there is a directed edge of type r from i to j: $\sigma(v_{r,i}^T \cdot v'_{r,j}) > \alpha$,



(a) Baby on JD Dataset (b) Office Product on Amazon

Figure 3: Accuracy-Training-Samples Curves

- if there is non-edge r from i to j: $\sigma(v_{r,i}^T \cdot v'_{r,j}) \leq \alpha$.

Note that the threshold α is selected between 0.1 and 0.9 on the validation set.

Results are shown in Table 5 and 6. Note that due to the different number of non-edge samples, our results of Sceptre method are different from those reported in [23]. To measure the overall predictive performance, we also demonstrate the ROC curves in Figure 4 and 5, where the AUC value of **PMSC** is coherently greater than those of the baselines. We summarize our observations as follows:

- **PMSC** is capable of accurately predicting both substitute and complement links in all categories, especially for clothes on Amazon dataset and Baby on JD dataset.
- **PMSC** significantly outperforms other baselines. MF and NMF can barely handle the substitutes and complements prediction for extremely sparse data. Comparing to Sceptre which extracts the related topics between product pairs from the review texts, product category, and multi-step path constraints uncover potential links directly. Besides, **PMSC** takes the dependencies among multiple relations into the consideration, which significantly facilitates learning the product and relation representations. Additionally, we show the performances of varying numbers of training samples per product in the link prediction task in Figure 3, where we can observe that the performance of **PMSC** is more stable than Sceptre, demonstrating that **PMSC** can address the sparsity problem well.
- **PMSC (Base+C1+C2)** consistently outperforms both **PMSC (Base+C1)** and **PMSC (Base)**, while **PMSC (Base+C1)** achieves higher accuracy on all categories comparing to **PMSC(Base)**. Thus, both product category constraints and multi-step path constraints can improve the accuracy of identifying substitutes and complements.

In summary, we are able to conclude that each component of **PMSC** contributes to the prediction accuracy. The basic **PMSC(Base)** has relatively low predictive accuracy in some product categories. The product category constraints and multi-step path constraints are helpful for both product and relation representation learning, as well as alleviating the data sparsity problem.

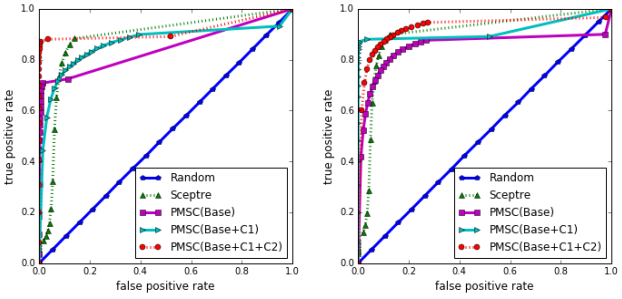
Ranking. In many applications, only a limited set of substitutes and complements are recommended. Thus, in addition to distinguishing links and non-links between the given product pairs, it is

Table 5: Link Prediction Accuracy on Amazon

Method	Electronics		Women's Clothes		Home and Kitchen		Cell Phones and Accessories		Office Products	
	Subst.	Compl.	Subst.	Compl.	Subst.	Compl.	Subst.	Compl.	Subst.	Compl.
Random	0.7800	0.7801	0.7803	0.7800	0.7808	0.7796	0.7809	0.7799	0.7800	0.7804
MF	0.8056	0.7916	0.7963	0.7850	0.8140	0.8094	0.7910	0.8000	0.7950	0.7927
NMF	0.8147	0.8013	0.8179	0.8023	0.8276	0.8159	0.7979	0.8269	0.8073	0.8165
Sceptre	0.9281	0.8789	0.9142	0.9091	0.8528	0.9161	0.8966	0.8895	0.9385	0.8912
PMSC (Base)	0.8698	0.8554	0.9178	0.8595	0.8483	0.8841	0.8820	0.9068	0.8688	0.8718
PMSC (Base+C1)	0.9176	0.8735	0.9284	0.9032	0.8501	0.8973	0.8923	0.9248	0.8770	0.8508
PMSC (Base+C1+C2)	**0.9790**	**0.9242**	**0.9777**	**0.9602**	**0.8631**	**0.9252**	**0.9016**	**0.9306**	**0.9778**	**0.9079**

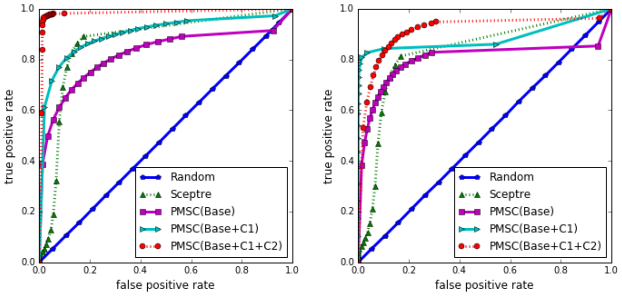
Table 6: Link Prediction Accuracy on JD Dataset

Method	Baby		Women's Clothes		Men's clothes		Cell Phones and Accessories		Office Products	
	Subst.	Compl.	Subst.	Compl.	Subst.	Compl.	Subst.	Compl.	Subst.	Compl.
Random	0.7799	0.7802	0.7797	0.7822	0.7797	0.7790	0.7800	0.7812	0.7799	0.7798
MF	0.7993	0.8092	0.7957	0.7961	0.7960	0.7903	0.7952	0.8072	0.8013	0.8106
NMF	0.8353	0.8070	0.8164	0.8170	0.8049	0.8281	0.8078	0.8239	0.8211	0.8284
Sceptre	0.9158	0.9156	0.8974	0.9138	0.8822	0.9105	0.9135	0.9148	0.9133	0.9154
PMSC (Base)	0.8035	0.9020	0.8761	0.7778	0.8722	0.8523	0.7691	0.8987	0.7086	0.8439
PMSC (Base+C1)	0.8174	0.9075	0.8876	0.8651	0.9104	0.8588	0.7727	0.9067	0.7283	0.9180
PMSC (Base+C1+C2)	**0.9741**	**0.9457**	**0.9411**	**0.9487**	**0.9481**	**0.9488**	**0.9570**	**0.9483**	**0.9501**	**0.9405**



(a) Substitute (b) Complement

Figure 4: ROC Curves for Cell Phones and Accessories on Amazon



(a) Substitute (b) Complement

Figure 5: ROC Curves for Office Product on Amazon

also important that the relevant products are scored higher than the irrelevant ones.

A standard measurement for ranking methods is the precision@k. To be specific, for a given relation r, precision is the fraction of recommended products that are relevant to the query product. The precision is defined as follow:

$$precision = \frac{|\{relevant\ products\} \cap \{recommened\ products\}|}{|\{recommened\ products\}|} \tag{18}$$

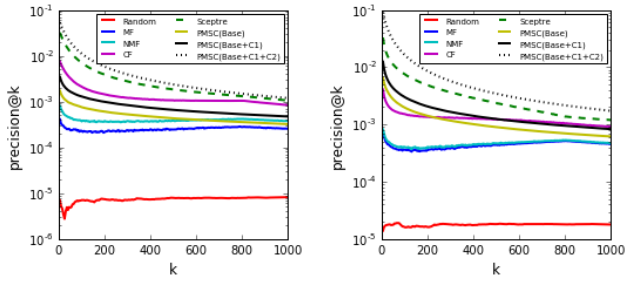
Regarding to the fact that only limited results are returned by the system, precision can be evaluated at a given cut-off rank k, *i.e.* precision@k. In our experiment, we follow the setting in [23] and discard the candidate links existing in the train and validation sets.

Our ranking results are reported in Figure 6 and 7 on Cell Phones and Accessories in both Amazon dataset and JD dataset. Since we only keep the candidate links appearing in the test set, only a small number of relevant products are remain for the query products. The Random method only has the precision around 10^{-5}, meaning that Random can only recommend about 1 relevant product in 100, 000 candidates, which indicates the extreme difficulty of the task.

It can be observed from the Figure 6 and 7 that MF, NMF and CF methods obtain much better performance than the Random method. The curve of **PMSC (Base+C1+C2)** is consistently above all baselines, indicating that **PMSC** is able to retrieve the most relevant products for each relation.

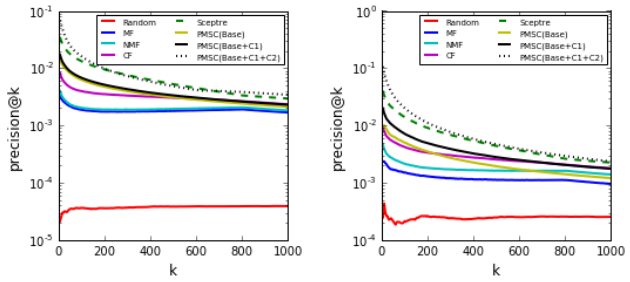
4.5 Cold-Start

From the previous two subsections, we can see that **PMSC** achieves the promising performance on link prediction and ranking. However, another important problem is how to predict links and recommend relevant products for new products, which are unseen in the training set. This kind of problem is notorious as "cold start" problem [26, 29, 40]. To address the "cold start" problem, we argue



(a) Substitute (b) Complement

Figure 6: Precision@k for Cell Phones and Accessories on Amazon



(a) Substitute (b) Complement

Figure 7: Precision@k for Cell Phones and Accessories on JD Dataset

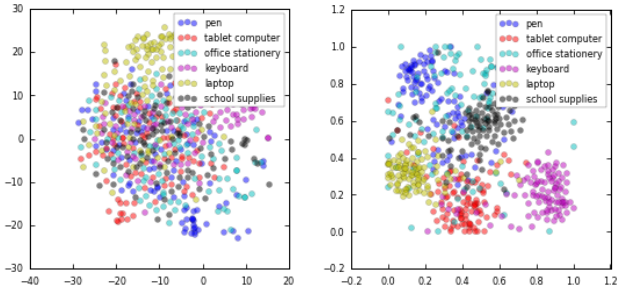
that, as we mentioned in §3.4, when the products are related, their categories are related as well. This indicates that product category information is critical for predicting the edges between products. Thus, we represent the new product using their "category feature representations", which is defined as follow:

$$v_{c,r} = \frac{\sum_{i \in c} v_{i,r}}{N_c}, \; v'_{c,r} = \frac{\sum_{i \in c} v'_{i,r}}{N_c} \tag{19}$$

where v_c and v'_c are the target and context representations of the category c, and $v_{c,r}$ is the projected target vector of category c for r, $v'_{c,r}$ is the projected context vector of category c for r and N_c is the number of the product in category c.

To evaluate the feasibility of using category representation, we predict links between the new products and the results are shown in Table 7 and 8. **PMSC** can still make accurate predictions, especially for Baby in JD dataset with 0.9667 and 0.9335 for substitutes and complements. The results indicate that **PMSC** is able to handle the "cold start" problem with category feature representations very well.

We here visualize the product representations with t-SNE [21] in Figure 8, where points stand for products and different colors signify different product categories. We can see that in Figure 8(b), by trained with constraints, vectors of products in the same category can be gathered together, while points are more randomly distributed in Figure 8(a). It actually implies the reason why **PMSC** is able to accurately predicting "new products" with category feature representations. By further incorporating path constraints,



(a) Training without Constraints (b) Training with Constraints

Figure 8: Visualization for Product Representations

Table 7: Cold Start Prediction Accuracy on Amazon

Category	Method	Accuracy	
		Substitute	Complement
Electronics	cold start	0.9745	0.8733
Women's Clothes	cold start	0.9731	0.8890

Table 8: Cold Start Prediction Accuracy on JD Dataset

Category	Method	Accuracy	
		Substitute	Complement
Baby	cold start	0.9667	0.9355
Women's Clothes	cold start	0.9404	0.8331

PMSC is able to automatically capture the semantics of items and identify product categories.

In addition, the "cold start" prediction of substitutes is consistently more accurate than complements. The substitutes are easier to predict since substitute products are in the same category while the semantics of complements are more complex. This is because products in the head position may connect with products from various categories in the complement relation, which actually makes the general semantics more vague.

5 CONCLUSION

In this paper, we present a novel framework for substitutes and complements prediction, called Path-constrained Method for discriminating Substitutes and Complements (**PMSC**). The framework learns and predicts multi relations by projecting product representation to relation-specific space. We identify the useful path constraints to uncover complex data pattern, and then incorporate such path constraints into the framework through t-norm fuzzy logics. We evaluate the proposed **PMSC** on two real-world datasets. Experimental results show that **PMSC** can effectively identify links and distinguish substitutable and complementary products.

ACKNOWLEDGEMENTS

The authors wish to thank the anonymous reviewers for their helpful comments. Jiliang Tang is supported by the National Science Foundation (NSF) under grant number IIS-1714741 and IIS-1715940.

REFERENCES

[1] O. Barkan. Bayesian neural word embedding. In *AAAI*, pages 3135–3143, 2017.
[2] O. Barkan and N. Koenigstein. Item2vec: neural item embedding for collaborative filtering. In *International Workshop on Machine Learning for Signal Processing*, pages 1–6, 2016.
[3] R. M. Bell and Y. Koren. Improved neighborhood-based collaborative filtering. In *KDD Cup and Workshop at the KDD*, 2007.
[4] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 7–10, 2016.
[5] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, pages 160–167, 2008.
[6] P. Covington, J. Adams, and E. Sargin. Deep neural networks for youtube recommendations. In *ACM RecSys*, pages 191–198, 2016.
[7] F. Gelli, X. He, T. Chen, and T.-S. Chua. How personality affects our likes: Towards a better understanding of actionable images. In *MM*, 2017.
[8] P. Hájek. *Metamathematics of fuzzy logic*, volume 4. Springer, 1998.
[9] R. He, C. Packer, and J. Mcauley. Learning compatibility across categories for heterogeneous item recommendation. In *IEEE International Conference on Data Mining*, pages 937–942, 2017.
[10] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. Neural collaborative filtering. In *WWW*, pages 173–182, 2017.
[11] N. Koenigstein, G. Dror, and Y. Koren. Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy. In *ACM RecSys*, pages 165–172, 2011.
[12] Y. Koren and R. Bell. Advances in collaborative filtering. In *Recommender systems handbook*, pages 145–186. Springer, 2011.
[13] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.
[14] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788, 1999.
[15] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562, 2001.
[16] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma. Neural attentive session-based recommendation. In *CIKM*, 2017.
[17] G. Linden, B. Smith, and J. York. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1):76–80, 2003.
[18] D. C. Liu, S. Rogers, R. Shiau, D. Kislyuk, K. C. Ma, Z. Zhong, J. Liu, and Y. Jing. Related pins at pinterest: The evolution of a real-world recommender system. In *WWW Companion*, pages 583–592, 2017.
[19] P. Lops, M. D. Gemmis, and G. Semeraro. *Content-based Recommender Systems: State of the Art and Trends*. Springer, 2011.
[20] Y. Ma, Z. Ren, Z. Jiang, J. Tang, and D. Yin. Multi-dimensional network embedding with hierarchical structures. In *WSDM*, 2018.
[21] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.
[22] J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *ACM RecSys*, pages 165–172, 2013.
[23] J. McAuley, R. Pandey, and J. Leskovec. Inferring networks of substitutable and complementary products. In *KDD*, pages 785–794. ACM, 2015.
[24] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.
[25] A. Mnih and G. E. Hinton. A scalable hierarchical distributed language model. In *NIPS*, pages 1081–1088, 2009.
[26] S.-T. Park and W. Chu. Pairwise preference regression for cold-start recommendation. In *ACM RecSys*, pages 21–28, 2009.
[27] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *ACM Conference on Computer Supported Cooperative Work*, pages 175–186, 1994.
[28] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, pages 285–295, 2001.
[29] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *SIGIR*, pages 253–260, 2002.
[30] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, N. Oliver, and A. Hanjalic. Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering. In *ACM RecSys*, pages 139–146, 2012.
[31] X. Su and T. M. Khoshgoftaar. *A survey of collaborative filtering techniques*. Hindawi Publishing Corp., 2009.
[32] Z. Sun, J. Yang, J. Zhang, and A. Bozzon. Exploiting both vertical and horizontal dimensions of feature hierarchy for effective recommendation. In *AAAI*, pages 189–195, 2017.
[33] I. Titov and R. McDonald. Modeling online reviews with multi-grain topic models. In *WWW*, pages 111–120, 2008.
[34] I. Titov and R. T. McDonald. A joint model of text and aspect ratings for sentiment summarization. In *ACL*, pages 308–316, 2008.
[35] C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. In *KDD*, pages 448–456, 2011.
[36] P. Wang, J. Guo, Y. Lan, J. Xu, and X. Cheng. Your cart tells you: Inferring demographic attributes from purchase data. In *WSDM*, pages 173–182, 2016.
[37] X. Yi, L. Hong, E. Zhong, N. N. Liu, and S. Rajan. Beyond clicks: dwell time for personalization. In *ACM RecSys*, pages 113–120, 2014.
[38] Y. Zhang, M. Zhang, Y. Liu, S. Ma, and S. Feng. Localized matrix factorization for recommendation based on matrix block diagonal forms. In *WWW*, pages 1511–1520, 2013.
[39] J. Zheng, X. Wu, J. Niu, and A. Bolivar. Substitutes or complements: another step forward in recommendations. In *Proceedings of the 10th ACM conference on Electronic commerce*, pages 139–146, 2009.
[40] K. Zhou, S.-H. Yang, and H. Zha. Functional matrix factorizations for cold-start recommendation. In *SIGIR*, pages 315–324, 2011.
[41] M. Zhou, Z. Ding, J. Tang, and D. Yin. Micro behaviors: A new perspective in e-commerce recommender systems. In *WSDM*, 2018.