Inferring Substitutable Products with Deep Network Embedding

Shijie Zhang[1]*, **Hongzhi Yin**[1]†*, **Qinyong Wang**[1], **Tong Chen**[1], **Hongxu Chen**[1] and **Quoc Viet Hung Nguyen**[2]

[1]School of Information Technology and Electrical Engineering, The University of Queensland, Australia
[2]School of Information and Communication Technology, Griffith University, Australia
{shijie.zhang, h.yin1, qinyong.wang, tong.chen, hongxu.chen}@uq.edu.au,
quocviethung.nguyen@griffith.edu.au

Abstract

On E-commerce platforms, understanding the relationships (e.g., substitute and complement) among products from user's explicit feedback, such as users' online transactions, is of great importance to boost extra sales. However, the significance of such relationships is usually neglected by existing recommender systems. In this paper, we propose a semi-supervised deep embedding model, namely, **S**ubstitute **P**roducts **E**mbedding **M**odel (SPEM), which models the substitutable relationship between products by preserving the second-order proximity, negative first-order proximity and semantic similarity in a product co-purchasing graph based on user's purchasing behaviours. With SPEM, the learned representations of two substitutable products align closely in the latent embedding space. Extensive experiments on seven real-world datasets are conducted, and the results verify that our model outperforms state-of-the-art baselines.

1 Introduction

In the era of information overload, recommender systems play a pivotal role in helping users find the products they are interested in [Koren and Bell, 2015; Yin *et al.*, 2016; Chen *et al.*, 2019]. Aiming to model the user's personal preferences, traditional recommender systems heavily rely on the availability of long-term user-item interactions and consistent user identifications for preference learning. Consequently, being able to infer users' intention with information mainly in the current browsing session, session-based recommendation [Li *et al.*, 2017; Wu and Yan, 2017; Guo *et al.*, 2019] based on implicit feedback (e.g., click, purchase) has become a rather practical research direction.

However, a critical drawback of existing session-based recommender systems is that they ignore the different relationships among products when providing recommendations in different intention contexts [Zheng *et al.*, 2009; McAuley *et al.*, 2015; Rakesh *et al.*, 2019]. There are two main product relationships, namely **substitute** and **complement**. Substitutable products are interchangeable and competitive, e.g., Sony cameras and Canon cameras, while complementary products are likely to be purchased together, e.g., Zeiss camera lenses are the complements for a Sony camera [Mas-Colell *et al.*, 1995]. More importantly, these two types of products have varied importance in different user intention contexts. Let us consider a scenario where a user is choosing a printer to buy. Apparently, before a purchase is observed in the current session, the recommender systems should recommend substitute products of the items visited by this user. This is because the current intention of the user is mainly to purchase a specific type of product (i.e., printer in this case), and these substitute products can offer substantial options for the user, thus increasing the chance of a successful purchase. Meanwhile, after the purchase of a printer is finalized, it would be more realistic to recommend complements like printing papers and ink cartridges instead of similar printers to stimulate further purchase intention. Hence, investigating the substitutable and complementary relationships can create a game-changer in session-based recommendation as it enables the recommendation of products that closely fit the context of each user's purchase intention [Wu and Yan, 2017]. Since complementary products naturally share frequent co-occurrence (e.g., purchased in the same order), many well-established methods like frequent pattern mining [Aggarwal *et al.*, 2014] and item-based recommendation [Sarwar *et al.*, 2001] can identify such complementary relationship in a straightforward way. As a result, in this paper, we only focus on inferring substitutable products because it is a relatively harder task compared with mining complementary relationship.

Despite the significant benefit, the study of substitutable product relationships remains largely unexplored. The most relevant models in existing literatures are Sceptre [McAuley *et al.*, 2015] and Linked Variational Autoencoder (LVA) [Rakesh *et al.*, 2019]. The problem of mining product relationships is first introduced in [McAuley *et al.*, 2015], and the proposed Sceptre infers substitutable products by extracting the textual features of products from the reviews and descriptions with the Latent Dirichlete Allocation (LDA) [Blei *et al.*, 2003]. LVA is a recent attempt that links two Variational Autoencoders (VAE) [Kingma and Welling, 2013] conditioned on the observed links among items. How-

*The authors equally contribute to this work. †indicates the corresponding author.

ever, both methods suffer from two major limitations. Firstly, they only consider the review texts, and the valuable information within users' historical purchase behaviours is neglected. In addition, while it is impractical to assume the constant availability of high-quality and sufficient reviews, the LDA in Sceptre is known to underperform on short texts [Jo and Oh, 2011; Titov and McDonald, 2008], which are common in user reviews. Secondly, they extract text features from the whole reviews which require large memory spaces for learned features, making these models inefficient on large datasets.

To this end, we propose a novel model named **S**ubstitutable **P**roduct **E**mbedding **M**odel (SPEM), which is able to scale to very large product co-purchasing graph. In SPEM, we first propose an innovative product graph named Product Co-purchase (PC) graph, which is constructed based on user behaviours. Specifically, in the PC graph, an edge between two product nodes indicates that customers have purchased both products frequently. The core idea of SPEM is to learn the vector embeddings of all products from the PC graph, then the proximity (i.e., substitutable relationship) between any two products can be inferred via common metrics like Euclidean distance. From the perspective of previous graph-based approaches [Hsieh *et al.*, 2017; Wang *et al.*, 2016; Chen *et al.*, 2018], if there is a link between two products in the PC graph, then the learned embeddings of these two products will tend to have high proximity. However, this is not true for substitutable product inference, because in reality, the co-purchased products are more likely to be complementary rather than substitutable. Intuitively, two substitutable products tend to have similar complementary products, and two complementary products must not have substitutable relationship. Formally, these properties are referred to as **second-order proximity** and **negative first-order proximity** respectively. For example, Dell and HP computers tend to have the same complementary products like the Logitech keyboards, while Dell computers and Logitech keyboards will never be the substitutes for each other. In addition, substitutable products should share the same categorical hierarchy, thus providing similar functions. In short, a representative embedding of a substitute should possess high **semantic similarity** to the embedding of the original product.

Technically, SPEM is a semi-supervised model that is able to preserve the three aforementioned properties of substitutable products in the embedding space. Motivated by the recent advances in network embedding [Wang *et al.*, 2016], SPEM learns a low-dimensional embedding for each product node via the unsupervised deep autoencoder, which preserves the second-order proximity within the PC graph. Meanwhile, a supervised pairwise constraint further exploits the substitutable and complementary properties to penalize the first-order proximity. In addition, we also devise a tree-based similarity function for SPEM to preserve the semantic similarity among substitutable products.

In summary, the main contributions of this paper are:

- We propose SPEM, a deep network embedding model that infers the substitutable relationship among products. The method is capable of mapping different products from the PC graph to a highly non-linear latent feature space without using any review data.
- We develop a novel semi-supervised deep model which can simultaneously optimize the second-order proximity, semantic similarity and prevent first-order proximity. Hence, the learned representations capture the crucial substitutable properties.
- We extensively evaluate the effectiveness of SPEM on seven real-life datasets from Amazon. The results show the superiority of our proposed model in inferring substitutable products against state-of-the-art baselines.

2 Preliminaries

We define the concepts used throughout this paper and then formulate the substitutable product embedding problem.

Definition 1 *(Product Co-purchase Graph) The PC graph is defined as $G = (V, E)$, where $V = \{v_1, ..., v_n\}$ is a set of n products. $E = \{e_{i,j}\}_{i,j=1}^{n}$ represents all possible edges between product vertices $v_i, v_j \in V$. Given two products $v_i, v_j \in V$, if more than t costumers have purchased both products, then the edge $e_{ij} = 1$, otherwise, we set e_{ij} to 0.*

Definition 2 *(First-Order Proximity) In the PC graph, for any pair of products v_i and v_j, if $e_{ij} > 0$, there exists the positive first-order proximity between v_i and v_j. If $e_{ij} = 0$, then the first-order proximity between them is 0.*

Definition 3 *(Second-Order Proximity) For $v_i \in V$, let $N_i = \{w_{i1}, w_{i2}, ..., w_{in}\}$ denote the first-order proximity between v_i and all other vertices. The second-order proximity between any two products v_i and v_j is defined as the similarity between N_i and N_j. Intuitively, the more neighbour nodes v_i and v_j have in common, the higher their second-order proximity will be. If v_i and v_j do not share any neighbours, their second-order proximity is 0.*

Definition 4 *(Substitutable Products) Substitutable products are usually interchangeable and competitors of each other. In the PC graph, two substitutable products v_i and v_j are expected to have a high second-order proximity, negative first-order proximity and high semantic similarity.*

Problem 1 *(Substitutable Product Embedding) Given a PC graph $G = (V, E)$, the target of embedding a substitutable product is to map each product vertex $v_i \in V$ into a low-dimensional space $\mathbb{R}^s$, i.e., $g : v_i \mapsto \mathbf{y_i} \in \mathbb{R}^s$, where the embedding dimension $s \ll n$. The mapping function aims to preserve the second-order proximity and the semantic similarity, as well as negative first-order proximity in space $\mathbb{R}^s$.*

3 Substitutable Product Embedding Model

In this section, we present the technical details of SPEM, a semi-supervised deep model for substitutable product embedding. The architecture of SPEM is illustrated in Figure 1. SPEM consists of three main components: (1) an **unsupervised deep autoencoder**; (2) a **supervised pairwise constraint**; and (3) a **tree-based semantic similarity function**. Based on the intuition that two substitutes tend to have many same or similar complementary products, the first component aims to preserve such second-order proximity while learning product embeddings. In addition, two substitutable products must not have complementary relationship, hence we develop the second module to maintain the negative first-order

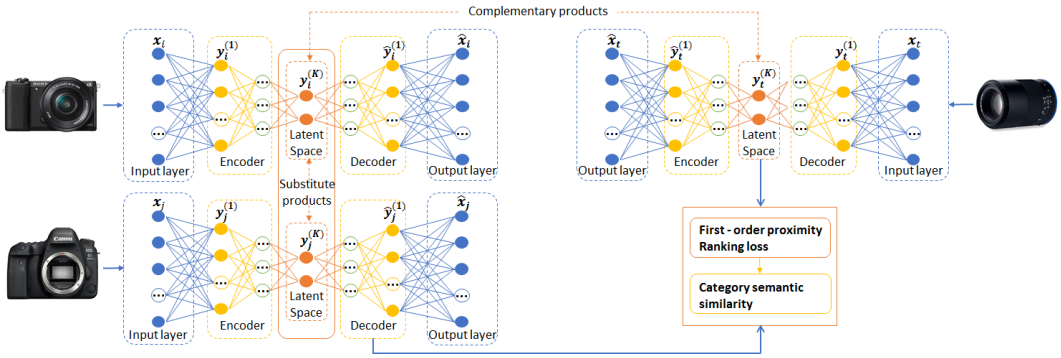


Figure 1: The overview of SPEM

proximity among substitutes. Furthermore, as two substitutes should provide the same or similar functions, thus having high semantic similarity, the third component emphasizes the affinity between the categorical hierarchy of two substitutes.

3.1 Preserving Second-Order Proximity with Unsupervised Deep Autoencoder

To preserve the second-order proximity, we propose an unsupervised deep neural network based on the autoencoder (AE) as the main building block of SPEM. Given a PC graph $G = (V, E)$, we can obtain its adjacency matrix $\mathbf{A} = \{0, 1\}^{n \times n}$, where each entry $a_{ij} \in \mathbf{A}$ is set to 1 if there exists a link between v_i and v_j, and 0 otherwise. We extend the standard AE to preserve the second-order proximity in the product embeddings via the adjacency matrix $\mathbf{A}$. Being widely adopted to learn vector representations of the inputs, AE is an unsupervised deep model with two major parts, i.e., the encoder and decoder. The encoder contains multiple non-linear functions which map raw inputs into a low-dimensional latent space. The decoder contains multiple non-linear functions, which are trained to reconstruct the inputs from the learned latent space to the reconstruction space. Then, for an arbitrary product v_i, given its input feature $\mathbf{x}_i$, the latent representation in the k-th layer is updated by the encoder as follows:

$$\begin{aligned} \mathbf{y}_i^{(1)} &= \sigma(\mathbf{W}^{(1)}\mathbf{x}_i + \mathbf{b}^{(1)}), \\ \mathbf{y}_i^{(k)} &= \sigma(\mathbf{W}^{(k)}\mathbf{y}_i^{(k-1)} + \mathbf{b}^{(k)}), k = 2, \ldots, K \end{aligned} \tag{1}$$

where $\mathbf{y}_i^{(k)}$ is the latent vector learned in the k-th layer, σ is the Sigmoid function, $\mathbf{W}^{(k)}$ and $\mathbf{b}^{(k)}$ are respectively the trainable weight matrix and bias for the k-th layer. Afterwards, taking the final low-dimensional embedding $\mathbf{y}_i^{(K)}$ as the input, the decoder derives the output $\hat{\mathbf{x}}_i$ by reversing the calculation process of encoder. The goal of the autoencoder is to minimize the reconstruction error of the outputs and the inputs in order to learn representative embeddings $\mathbf{y}_i^{(k)}$ of the inputs. The loss function $\mathcal{L}_{AE}$ for AE can be formulated as:

$$\begin{aligned} \mathcal{L}_{AE} &= \sum_{i=1}^{n} \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|_2^2 + \mathcal{L}_{reg} \\ \mathcal{L}_{reg} &= \frac{1}{2}\sum_{k=1}^{K}\left(\left\|\mathbf{W}^{(k)}\right\|_F^2 + \left\|\hat{\mathbf{W}}^{(k)}\right\|_F^2\right) \end{aligned} \tag{2}$$

where $\mathcal{L}_{reg}$ is the $\mathcal{L}2$-norm regularization to prevent overfitting. By minimizing the reconstruction loss $\mathcal{L}_{AE}$, the output $\hat{\mathbf{x}}_i$ tries to recover the information of the input data in the latent space. In such case, for each product, instead of using its one-hot encoding as the input, we propose to fully leverage the information within the adjacency matrix $\mathbf{A}$. Specifically, we set $\mathbf{x}_i = \mathbf{a}_i$ where $\mathbf{a}_i$ denotes the i-th row of $\mathbf{A}$. Since $\mathbf{x}_i$ carries the neighborhood information of product v_i, the reconstruction loss will force the model to reconstruct such information in $\hat{\mathbf{x}}_i$ from the latent vector $\mathbf{y}_i^{(K)}$. As a result, $\mathbf{y}_i^{(K)}$ will be capable of preserving the second-order proximity in the latent space. Besides, due to the sparsity of the PC graph, i.e., the number of zero elements in A is far more than that of non-zero elements, we devise an additional penalty on the reconstruction loss to lay more emphasis on the accurate reconstruction of non-zero elements in $\hat{\mathbf{x}}_i$. Hence, we reformulate the loss function in Eq. (2) as $\mathcal{L}_{sp}$ for preserving the second-order proximity:

$$\mathcal{L}_{sp} = \sum_{i=1}^{n} \|(\hat{\mathbf{x}}_i - \mathbf{x}_i) \odot \mathbf{c}_i\|_2^2 + \mathcal{L}_{reg} \tag{3}$$

where $\odot$ denotes the Hadamard product, $\mathbf{c_i} = \{c_{ij}\}_{j=1}^{n}$ contains the penalty weights. If $a_{ij} = 1$, $c_{ij} = \tau$, else $c_{ij} = 1$ Note $\tau > 1$ is the penalty coefficient to be tuned. By minimizing $\mathcal{L}_{sp}$, substitutable vertices with similar neighborhood structures will be mapped closely via the learned embedding vector $\mathbf{y}_i^{(K)}$, thus enabling the AE to accurately reconstruct the input adjacency matrix $\mathbf{A}$. Hence, the second-order proximity will be retained by the product embedding $\mathbf{y}_i^{(K)}$.

3.2 Supervised Learning on Negative First-Order Proximity

While the current product embedding $\mathbf{y}_i^{(K)}$ of product v_i only incorporates the second-order proximity among substitutable products, now we introduce a supervised pairwise constraint to exploit the negative first-order proximity in the PC graph. Intuitively, in the latent space, for product v_i, given its substitute v_j and complement v_t, a reasonable substitute product embedding approach should ensure that v_i is always closer to v_j than v_t in the latent space. With the availability of the ground truth (see Section 4.1) for both substitutable and complementary products, we formulate the following pairwise constraint to preserve the negative first-order proximity:

$$d(\mathbf{y}_i^{(K)}, \mathbf{y}_j^{(K)}) < d(\mathbf{y}_i^{(K)}, \mathbf{y}_t^{(K)}), \forall v_i \in V, \forall v_j \in V_i^{sub}, \forall v_t \in V_i^{com} \tag{4}$$

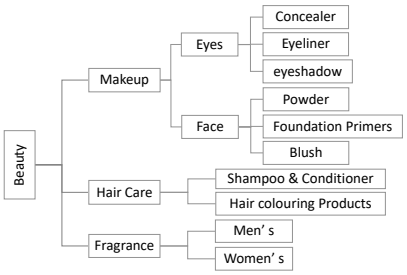


Figure 2: Part of the product tree collected from Amazon Beauty products (the complete tree is too large to show here).

where V_i^{sub} and V_i^{com} are the set of labelled substitutes and complements for vertex v_i, respectively.

To pose the pairwise constraint in Eq. (4) on the learning process of embedding $\mathbf{y}_i^{(K)}$, we adopt an energy-based learning approach [LeCun *et al.*, 2006] to quantify the distance between v_i and its substitutes or complements. Mathematically, we employ the Euclidean distance to compute the pairwise energy between v_i and v_j, denoted by $d(\mathbf{y}_i^{(K)}, \mathbf{y}_j^{(K)}) = \left\| \mathbf{y}_i^{(K)} - \mathbf{y}_j^{(K)} \right\|_2$. Then, the loss function for supervised learning on negative first-order proximity can be defined as follows:

$$\mathcal{L}_{fp} = \sum_{i,j,t \in L} \left(\left\| \mathbf{y}_i^{(K)} - \mathbf{y}_j^{(K)} \right\|_2^2 + \exp(- \left\| \mathbf{y}_i^{(K)} - \mathbf{y}_t^{(K)} \right\|_2) \right) \quad (5)$$

where L is the set of valid triplets defined in Eq. (4). In short, $\mathcal{L}_{fp}$ penalizes the pairwise energy (i.e., distance), which forces the energy of substitutable product pairs to be lower than that of complementary product pairs. Equivalently, the model discourages any first-order proximity between two substitutes in the latent space, thus maintaining the negative first-order proximity for substitutable products.

3.3 Tree-Based Semantic Similarity Modelling

To further capture the semantic similarity between substitutes, we construct a category tree from the data. An example of the category tree built upon the Amazon Beauty dataset is shown in Figure 2. Specifically, each node in the tree represents a category tag in the hierarchy. Therefore, based on the strategy in [Li *et al.*, 2003] which have been proven effective on measuring tree-based semantic similarity, the semantics of two different products can be compared according to their exact positions in the category tree.

For the comparison between vertices v_i and v_j, we combine the length l_{ij} of the shortest path between them with the depth d_{ij} of their lowest common ancestor in a non-linear way. On that basis, we propose a loss function to model the semantic similarity between the embeddings $\mathbf{y}_i^{(K)}$ and $\mathbf{y}_j^{(K)}$ of two substitutes, which incurs a penalty when two substitutable products under the similar category are mapped far from each other in the embedding space:

$$\mathcal{L}_{ss} = \sum_{i \in V} \sum_{j \in V_i^{sub}} sim(v_i, v_j) \left\| \mathbf{y}_i^{(K)} - \mathbf{y}_j^{(K)} \right\|_2$$
$$sim(v_i, v_j) = \exp(-\theta l_{ij}) * \frac{\exp(\mu d_{ij}) - \exp(-\mu d_{ij})}{\exp(\mu d_{ij}) + \exp(-\mu d_{ij})} \quad (6)$$

where θ and μ control the strength of l_{ij} and d_{ij}, respectively.

3.4 Objective Function of SPEM

Finally, to simultaneously encode the second-order proximity, negative first-order proximity and semantic similarity when learning embeddings for substitutable products from the PC graph, the loss function of SPEM is defined as the combination of Eq. (3), Eq. (5) and Eq. (6):

$$\mathcal{L} = \alpha \mathcal{L}_{sp} + \beta \mathcal{L}_{fp} + \gamma \mathcal{L}_{ss} \quad (7)$$

where α, β and γ are hyper-parameters that control the weights of different components.

4 Experiments

In this section, we evaluate the SPEM by competing against state-of-the-art baselines on real datasets from Amazon.

4.1 Datasets

We use seven publicly available Amazon product review datasets collected by [McAuley *et al.*, 2015], and the statistics of the datasets are listed in Table 1. For each category, we construct a PC graph with the following procedure. An edge between two products is established if they have been co-purchased by at least t common users, where the value of t is determined by the user-item interaction density of each dataset. In particular, we set t to 3 for Beauty, Baby and Video Games, and 5 for the remaining datasets. Following [McAuley *et al.*, 2015], we label the substitutable and complementary product pairs. Specifically, [McAuley *et al.*, 2015] defines four types of relationships between product *X* and *Y*: (1) users viewed *X* also viewed *Y*; (2) users viewed *X* eventually bought *Y*; (3) users bought *X* also bought *Y*; (4) users bought *X* also bought *Y* simultaneously. According to [McAuley *et al.*, 2015], we refer to (1) and (2) as substitute relationship while (3) and (4) as complement relationship.

We randomly choose 85% of the labelled substitutable product pairs as the training set, and use the remaining pairs as the test set. For each substitute product pair (*X*, *Y*) in the training dataset, we randomly choose a node *Z* from *X*'s neighboring nodes, and then add the triplet (*X*, *Y*, *Z*) to L as the supervised information in the loss function $\mathcal{L}_{fp}$.

4.2 Comparison Methods

We compare SPEM with the following four baseline methods in substitutable product inference.

Category-Tree (CT): Category-Tree is built using the category attributes in the datasets. Hence, we can easily compute the semantic similarity between two products. Note that CT only depends on semantic similarity to discover substitutes.

CML [Hsieh *et al.*, 2017]: CML learns a joint metric space to encode item-item relationships by pulling the related pairs closer and pushing the irrelevant pairs further apart in the latent space.

Sceptre [McAuley *et al.*, 2015]: Sceptre combines topic modelling and supervised link prediction to predict substitutable relationships among products from their textual reviews and descriptions.

LVA [Rakesh *et al.*, 2019]: LVA links two variational autoencoders conditioned on the observed links among items. It is the state-of-the-art model for discovering substitutes.

Category	#Users	#Items	#Reviews
Apps for Android	1.32M	61K	2.63M
Beauty	1.21M	259K	2.02M
Baby	532K	64K	915K
Cell Phone and Accessories	2.26M	347k	3.45M
Electronics	4.25M	498K	11.4M
Kindle Store	1.41M	431K	3.21M
Video Games	827K	51K	1.32M

Table 1: Dataset statistics for selected categories on Amazon.

Category	α	β	γ	dimensions in each layer
Apps for Android	0.6	0.04	1	48537-1000-250
Beauty	0.6	0.07	0.7	10427-1000-100
Baby	0.1	0.15	0.1	9595-1000-100
Cell Phone and Accessories	0.6	0.1	1	17918-1000-100
Electronics	0.6	0.1	1	55122-1000-100
Kindle Store	0.6	0.1	1	47985-1000-100
Video Games	0.6	0.09	0.1	12013-1000-100

Table 2: Parameter settings in different categories

To further investigate the significance of each model component, we design three variants of SPEM as follows:

SPEM-SS: SPEM-SS removes the supervised component from the full SPEM.

SPEM-FS: SPEM-FS removes the second-order proximity from the full SPEM.

SPEM-SF: SPEM-SF does not consider the effects of the semantic similarity.

4.3 Evaluation Protocol

Given a pair of products, we require all comparison methods to infer whether there exists substitutable relationship between them. To measure the inference accuracy, we adopt $Hits@k$ that is widely used to evaluate rank-based methods [Yin *et al.*, 2017; Yin *et al.*, 2018; Wang *et al.*, 2018]. For each substitute product pair (v_i, v_j) in the test set, we take the following steps for evaluation. (1) We randomly sample ζ products with which product v_i is irrelevant. Note that in our experiments, ζ is set to 500. (2) We compute the distance between v_i and ζ products in latent space. (3) We sort these $\zeta + 1$ products in ascending order. Let I_j denote the position of v_j in the ranked list. (4) If $I_j \leq k$, we get a hit, otherwise, we get a miss. The hit may vary with the change of k and ζ, thus k and ζ should be fixed for all comparison methods to ensure fairness. $Hits@k$ is then defined by the average of total hits over the whole test set. We vary the value of k by $\{10, 20, 30, 50, 100\}$.

4.4 Parameter Settings

We adopt a three-layer network for both the encoder and decoder of SPEM, i.e., $K = 3$, and the hidden dimension of each layer is reported in Table 2. The hyper-parameters of α, β and γ are also listed in Table 2. Note that these parameters are tuned to obtain the best performance in our model. For θ and μ in $\mathcal{L}_{ss}$, the optimal values are $\theta = 0.2$ and $\mu = 0.6$, following [Li *et al.*, 2003]. To train SPEM, we set τ, batch size and learning rate as 5, 16 and 0.01 following [Wang *et al.*, 2016].

Category	Method	k@10	k@20	k@30	k@50	k@100
Apps for Android	CT	0.73	0.75	0.77	0.80	0.83
	LVA	0.51	0.57	0.63	0.71	0.79
	CML	0.64	0.69	0.73	0.75	0.77
	Sceptre	0.29	0.39	0.39	0.41	0.47
	SPEM	**0.84**	**0.85**	**0.85**	**0.85**	**0.86**
Beauty	CT	0.92	0.93	0.93	0.94	0.94
	LVA	0.53	0.57	0.63	0.70	0.74
	CML	0.88	0.94	0.96	0.96	**0.97**
	Sceptre	0.33	0.50	0.64	0.68	0.75
	SPEM	**0.96**	**0.96**	**0.97**	**0.97**	**0.97**
Baby	CT	-	-	-	-	-
	LVA	0.47	0.53	0.58	0.63	0.70
	CML	0.72	0.80	0.85	0.86	0.89
	Sceptre	0.25	0.35	0.44	0.59	0.80
	SPEM	**0.89**	**0.90**	**0.90**	**0.91**	**0.92**
Cell Phones and Accessories	CT	0.43	0.45	0.46	0.49	0.55
	LVA	0.32	0.37	0.44	0.47	0.54
	CML	0.42	0.45	0.47	0.49	0.52
	Sceptre	0.18	0.25	0.34	0.41	0.51
	SPEM	**0.56**	**0.58**	**0.60**	**0.62**	**0.66**
Electronics	CT	0.14	0.15	0.17	0.21	0.32
	LVA	0.62	0.70	0.74	0.78	0.82
	CML	0.65	0.68	0.70	0.71	0.72
	Sceptre	0.57	0.64	0.70	0.75	0.80
	SPEM	**0.77**	**0.78**	**0.79**	**0.80**	**0.83**
Kindle Store	CT	0.13	0.16	0.18	0.21	0.24
	LVA	0.46	0.49	0.51	0.55	0.59
	CML	0.28	0.31	0.35	0.37	0.42
	Sceptre	0.41	045	0.48	0.53	0.57
	SPEM	**0.48**	**0.50**	**0.51**	**0.55**	**0.60**
Video Games	CT	0.18	0.20	0.22	0.26	0.33
	LVA	0.55	0.62	0.74	0.83	0.89
	CML	0.62	0.67	0.70	0.72	0.74
	Sceptre	0.42	0.60	0.71	0.80	0.87
	SPEM	**0.91**	**0.91**	**0.92**	**0.93**	**0.93**

Table 3: $Hits@k$ of all comparison methods on seven datasets (the hierarchy data of Baby is not available). We bold the highest result in each dataset.

4.5 Experimental Results

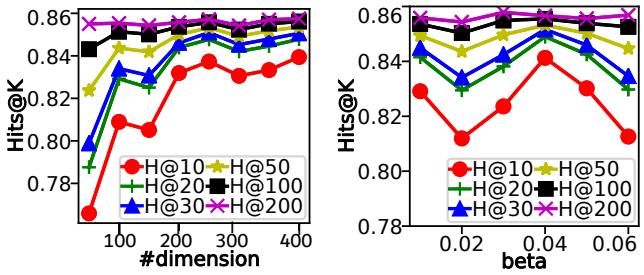
In this section, we report the comparison results of our SPEM model, the baselines and four variants of SPEM.

Substitute Products Prediction

The evaluation results of $Hits@k$ on all datasets of the comparison methods are presented in Table 3. From the results, we make the following observations. (1) SPEM outperforms all other methods across different product categories. It also illustrates that the $Hits@k$ of our method is consistently the highest with increasing k, which demonstrates that our model is capable of inferring substitutable relationship between two vertices. Particularly, our model has significant improvements over baselines on the three categories Beauty, Baby and Video Games (over 85%). (2) There is no winner among baselines across all categories. SPEM and CML achieve much higher prediction accuracy than LVA and Sceptre, indicating the benefits of behaviour-based methods over review-based methods. Besides, LVA outperforms Sceptre. The reason is that VAE mitigates the problems associated with LDA. For instance, LDA cannot work well on short text which lacks contextual information. (3) CT drops behind other three methods in Electronics, Kindle Store and Video Games, while it is better in Apps for Android and Beauty. It shows that the performance of CT heavily depends on the categories of products. Meanwhile, it indicates the benefit of exploiting the second-order proximity and negative first-order proximity.

Method	k@10	k@20	k@30	k@50	k@100
SPEM-SS	0.78	0.81	0.82	0.83	0.85
SPEM-FS	0.74	0.78	0.79	0.82	0.84
SPEM-SF	0.80	0.83	0.83	0.84	0.85
SPEM	**0.84**	**0.85**	**0.85**	**0.85**	**0.86**

Table 4: Hits@k for ablation studies on Apps for Android. The best results are bold.



(a) #Dimension (b) β

Figure 3: Sensitivity w.r.t. dimension and β

Ablation Study

We further compare SPEM with its three variants and only report the comparison results on the Apps for Android category in Table 4 due to the space limit. From the table, We have the following observations: (1) SPEM significantly outperforms the three variants on different categories, indicating the benefits brought by each component, respectively. For example, the results gap between SPEM and SPEM-FS validates the benefit of second-order proximity; (2) We find that the contribution of each component to improving prediction accuracy is varied. The second-order proximity is the most important component. The reason is that the high second-order proximity describing common neighborhood nodes implies that two products can be replaced in the PC graph, which is a stronger indicator of substitute products than other two components.

4.6 Parameter Sensitivity

We investigate the parameter sensitivity of SPEM, including the number of embedding dimensions and the value of hyper-parameter β, which controls the degree between supervised and unsupervised learning. We report $Hits@k$ on the category of Apps for Android.

Figure 3a reports the results of accuracy ($Hits@k$) w.r.t. different embedding dimensions. We can observe that the accuracy is in a fluctuating growth trend initially and raises to a peak. Then, the accuracy declines slightly and becomes stable when the number of dimensions continuously increases. The reason is that the latent vectors are capable of capturing most of the useful information. Overlarge dimension can increase noises and consume more computing resources, and has negative effect on the results. Overall, an appropriate number of the latent dimension is important to learn the nodes representation, but SPEM is not sensitive when it is larger than 250.

Figure 3b shows how the value of β affects the prediction accuracy. Obviously the optimal value for β is 4, where the model has the best performance when working with other optimal parameters listed in Section 4.4.

5 Related Work

Product relationship inference. The most closely related works are Sceptre and LVA, which extract features from reviews. However, heavily relying on the product reviews is a serious drawback, especially when reviews are insufficient. Moreover, LDA is known to have poor performance on short texts. While VAE training often results in a degenerate local optimum known as "posterior collapse" where the model learns to ignore the latent variable and the approximate posterior mimics the prior [He *et al.*, 2019], which is serious in text modelling. Instead of using reviews, PC graph was constructed based on user behaviours to mine the substitutable relationship among products.

Network embedding (NE). NE solves the problem of nodes representation in low-dimension space. DeepWalk [Perozzi *et al.*, 2014] is empirically effective by combination of random walk and skip-gram. LINE [Tang *et al.*, 2015] designs two loss functions to preserve the first-order and second-order proximity. [Wang *et al.*, 2016] proposes a semi-supervised deep model, which has multiple layers of non-linear functions to capture the network structure. Despite the success of these NE approaches, they are all proposed to preserve the first-order or the second-order proximity. SPEM is the first graph embedding model that explores to preserve the second-order proximity and negative first-order proximity based on the characteristics of substitutable relationship.

Semantic similarity. Semantic similarity measurement plays an important role in information retrieval and natural language processing [Li *et al.*, 2003]. In hierarchy, [Rada *et al.*, 1989] proves that the minimum number of edges from one node to another is a metric to measure the conceptual distance. [Jiang and Conrath, 1997] extends the work to utilize weighted links to compute the similarity between nodes. However, these methods fail to handle the large and general semantic nets. To this end, [Li *et al.*, 2003] proposes a similarity measure that considers shortest path length and depth of the ancestor in the hierarchy, achieving better performance.

6 Conclusions

In this paper, we proposed a novel model, namely SPEM, to discover the embedding of substitutable products. We first proposed a semi-supervised deep Autoencoder to preserve the second-order proximity, then we used a supervised pairwise constraint including substitutable and complementary information to maintain the negative first-order proximity. A tree-based semantic similarity function is also employed to preserve the functional affinity between substitutes. Moreover, we conducted extensive experiments to evaluate the performance of SPEM in real-world datasets , which demonstrate significant improvements over state-of-the-art baselines.

Acknowledgments

This work was supported by ARC Discovery Project (Grant No.DP190101985, DP170103954).

References

[Aggarwal *et al.*, 2014] Charu C Aggarwal, Mansurul A Bhuiyan, and Mohammad Al Hasan. Frequent pattern mining algorithms: A survey. In *Frequent Pattern Mining*, pages 19–64. 2014.

[Blei *et al.*, 2003] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *JMLR*, pages 993–1022, 2003.

[Chen *et al.*, 2018] Hongxu Chen, Hongzhi Yin, Weiqing Wang, Hao Wang, Quoc Viet Hung Nguyen, and Xue Li. Pme: projected metric embedding on heterogeneous networks for link prediction. In *SIGKDD*, pages 1177–1186, 2018.

[Chen *et al.*, 2019] Tong Chen, Hongzhi Yin, Hongxu Chen, Rui Yan, Quoc Viet Hung Nguyen, and Xue Li. Air: Attentional intention-aware recommender systems. In *ICDE*, 2019.

[Guo *et al.*, 2019] Lei Guo, Hongzhi Yin, Qinyong Wang, Tong Chen, Alexander Zhou, and Quoc Viet Hung Nguyen. Streaming session-based recommendation. In *SIGKDD*, 2019.

[He *et al.*, 2019] Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. Lagging inference networks and posterior collapse in variational autoencoders. In *ICLR*, 2019.

[Hsieh *et al.*, 2017] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. Collaborative metric learning. In *WWW*, pages 193–201, 2017.

[Jiang and Conrath, 1997] Jay J Jiang and David W Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. 1997.

[Jo and Oh, 2011] Yohan Jo and Alice H Oh. Aspect and sentiment unification model for online review analysis. In *WSDM*, pages 815–824, 2011.

[Kingma and Welling, 2013] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2013.

[Koren and Bell, 2015] Yehuda Koren and Robert Bell. Advances in collaborative filtering. In *Recommender Systems Handbook*, pages 77–118. 2015.

[LeCun *et al.*, 2006] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 2006.

[Li *et al.*, 2003] Yuhua Li, Zuhair A Bandar, and David McLean. An approach for measuring semantic similarity between words using multiple information sources. *TKDE*, pages 871–882, 2003.

[Li *et al.*, 2017] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. Neural attentive session-based recommendation. In *CIKM*, pages 1419–1428, 2017.

[Mas-Colell *et al.*, 1995] Andreu Mas-Colell, Michael Dennis Whinston, Jerry R Green, et al. *Microeconomic Theory*. Oxford University Press New York, 1995.

[McAuley *et al.*, 2015] Julian McAuley, Rahul Pandey, and Jure Leskovec. Inferring networks of substitutable and complementary products. In *SIGKDD*, pages 785–794, 2015.

[Perozzi *et al.*, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *SIGKDD*, pages 701–710, 2014.

[Rada *et al.*, 1989] Roy Rada, Hafedh Mili, Ellen Bicknell, and Maria Blettner. Development and application of a metric on semantic nets. *IEEE transactions on systems, man, and cybernetics*, pages 17–30, 1989.

[Rakesh *et al.*, 2019] Vineeth Rakesh, Suhang Wang, Kai Shu, and Huan Liu. Linked variational autoencoders for inferring substitutable and supplementary items. In *WSDM*, pages 438–446, 2019.

[Sarwar *et al.*, 2001] Badrul Munir Sarwar, George Karypis, Joseph A Konstan, John Riedl, et al. Item-based collaborative filtering recommendation algorithms. In *WWW*, pages 285–295, 2001.

[Tang *et al.*, 2015] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *WWW*, pages 1067–1077, 2015.

[Titov and McDonald, 2008] Ivan Titov and Ryan McDonald. Modeling online reviews with multi-grain topic models. In *WWW*, pages 111–120, 2008.

[Wang *et al.*, 2016] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *SIGKDD*, pages 1225–1234, 2016.

[Wang *et al.*, 2018] Qinyong Wang, Hongzhi Yin, Zhiting Hu, Defu Lian, Hao Wang, and Zi Huang. Neural memory dtreaming recommender networks with adversarial training. In *SIGKDD*, pages 2467–2475, 2018.

[Wu and Yan, 2017] Chen Wu and Ming Yan. Session-aware information embedding for e-commerce product recommendation. In *CIKM*, pages 2379–2382, 2017.

[Yin *et al.*, 2016] Hongzhi Yin, Xiaofang Zhou, Bin Cui, Hao Wang, Kai Zheng, and Quoc Viet Hung Nguyen. Adapting to user interest drift for poi recommendation. *TKDE*, pages 2566–2581, 2016.

[Yin *et al.*, 2017] Hongzhi Yin, Weiqing Wang, Hao Wang, Ling Chen, and Xiaofang Zhou. Spatial-aware hierarchical collaborative deep learning for poi recommendation. *TKDE*, pages 2537–2551, 2017.

[Yin *et al.*, 2018] Hongzhi Yin, Lei Zou, Quoc Viet Hung Nguyen, Zi Huang, and Xiaofang Zhou. Joint event-partner recommendation in event-based social networks. In *ICDE*, pages 929–940, 2018.

[Zheng *et al.*, 2009] Jiaqian Zheng, Xiaoyuan Wu, Junyu Niu, and Alvaro Bolivar. Substitutes or complements: Another step forward in recommendations. In *EC*, pages 139–146, 2009.