Product Knowledge Graph Embedding for E-commerce

Da Xu*
Chuanwei Ruan*
Walmart Labs
Sunnyvale, California, USA
{Da.Xu,Chuanwei.Ruan}@walmartlabs.com

Evren Korpeoglu, Sushant Kumar,
Kannan Achan
Walmart Labs
Sunnyvale, California, USA
{EKorpeoglu,SKumar4,KAchan}@walmartlabs.com

ABSTRACT

In this paper, we propose a new product knowledge graph (PKG) embedding approach for learning the intrinsic product relations as product knowledge for e-commerce. We define the key entities and summarize the pivotal product relations that are critical for general e-commerce applications including marketing, advertisement, search ranking and recommendation. We first provide a comprehensive comparison between PKG and ordinary knowledge graph (KG) and then illustrate why KG embedding methods are not suitable for PKG learning. We construct a self-attention-enhanced distributed representation learning model for learning PKG embeddings from raw customer activity data in an end-to-end fashion. We design an effective multi-task learning schema to fully leverage the multi-modal e-commerce data. The Poincaré embedding is also employed to handle complex entity structures. We use a real-world dataset from *grocery.walmart.com* to evaluate the performances on knowledge completion, search ranking and recommendation. The proposed approach compares favourably to baselines in knowledge completion and downstream tasks.

CCS CONCEPTS

• **Information systems** → **Data mining**; **Web searching and information discovery**; **Retrieval models and ranking**.

KEYWORDS

Knowledge graph; Relation learning; Representation learning; Search ranking; Recommendation; Information retrieval

ACM Reference Format:
Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar,, and Kannan Achan. 2020. Product Knowledge Graph Embedding for E-commerce. In *The Thirteenth ACM International Conference on Web Search and Data Mining (WSDM '20), February 3–7, 2020, Houston, TX, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3336191.3371778

1 INTRODUCTION

Understanding the relations among products as product knowledge play pivotal roles in the rapidly developing e-commerce world. Product relations, including `complement` (`co-buy`), `co-view` and `substitute`, are central for marketing, advertising and recommendation. Several papers have devoted to learning and inferring product relations from either predefined product relation graph [13] or customer records [36]. Besides the above relations, the interactions between products and natural language is gaining increasing attention [2]. Firstly, product descriptions provide valuable side information for various tasks. Secondly, customers often engage with products via search activities. We use the `search` and `describe` relation to summarizing interactions between natural language and products. On top of the descriptions, products are often grouped into hierarchical categories shown in Figure 1a, which motivates the `IsA` relationship. In this paper, we focus on the above six key relations for product knowledge, which should satisfy most e-commerce applications. Notice that these relations can be subdivided into more fine-grained levels depending on the use case. For example, `complement` can be divided into `AddOn`, `AccessoryTo`, `PartOf`, and `describe` summarizes `HasAttribute`, `Brand`, `Name`, etc.

By treating products, words and category labels as entities and relations as edges, the multi-relation product knowledge can be efficiently summarized by product knowledge graph like Figure 1b. This motivates us to compare our work for learning PKG with the well-established work in knowledge graph (KG) learning, especially knowledge graph embedding [30]. KG embedding methods learn the representation (embedding) of entities and relations in a lower-dimensional continuous vector space. The inherent structure of the KG is geometrically preserved in the vector space, and the embeddings can simplify manipulations while remaining useful for downstream applications such as knowledge completion. The prevalent KG embedding models including *TransE* [5], *TransH* [32], *TransR* [10] and *TransD* [8] measure the plausibility of observed facts in KG with translational distance, while *RESCAL* [17], *DistMult* [34], *HolE* [16] and *ComplEx* [23] use latent semantic similarity. Subsequent work improves the model complexities to further explore additional information and structural properties of KG [30].

However, most KG embedding models rely on the crucial assumption that all the facts in knowledge base are established with high plausibility, which is hardly valid when learning PKG embedding. In e-commerce data, other than the obvious `describe` and `IsA` relation, the `complement`, `co-view`, `substitute` and `search` relations all need to be extracted from customer records by information retrieval methods. For example, two `coffee` products that are very similar in the title and brand may not `substitute` each other because one is *decaffeinated* and the other is *caffeinated*. The customer substitution records, in this case, is a better source for informing the `substitute` relation. Just like many other machine learning applications in e-commerce, learning PKG embedding cannot circumvent the **noise** and **sparsity** issues in the customer-product

*Both authors contributed equally to this research.

WSDM '20, February 3–7, 2020, Houston, TX, USA

ACM ISBN 978-1-4503-6822-3/20/02…$15.00
https://doi.org/10.1145/3336191.3371778

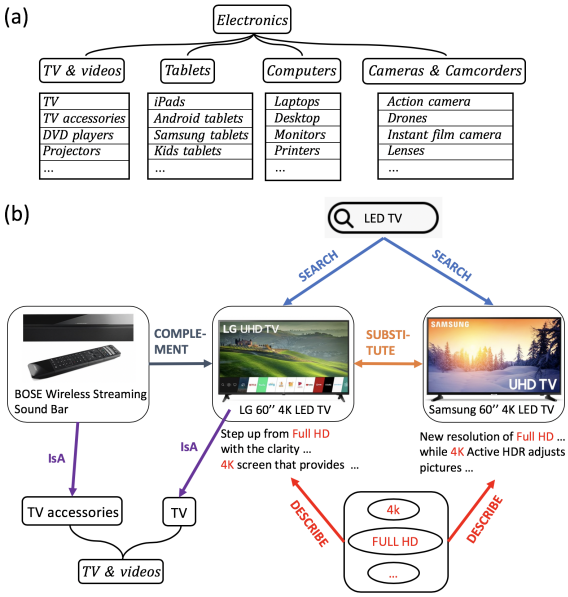

Figure 1: Visual illustrations. (a) Example of product category hierarchy. (b) Sketched product knowledge graph.

interaction data. If retrieving target relations is not complicated enough, the irregular tree-structured product categories pose further challenges since they are inherently hard to be embedded into Euclidean spaces. Last but not least, even the product descriptions are sometimes filled with **noise** words that are irrelevant to the `describe` relation.

Firstly, to break down the **noise** issues, we point out that our learning objectives are intrinsically discrete event-sequence learning similar to that of the neural machine language translation [3]. In the following example, we try to retrieve the `complement` relation for *toothpaste* and *toothbrush* from a purchase sequence:

$$[\ldots, \text{soap, detergent, toothbrush, towel} \rightarrow \text{toothpaste}, \ldots].$$

Among the products purchased prior to *toothpaste*, the *toothbrush* should be recognized with highest importance such that the input product sequence 'translates' to *toothpaste*. Same argument applies to product description, where for an ice cream described by:

$$\{\text{The strawberry ice cream featured by Haagan-Dazs is the marriage of sweet summer strawberries to cream and } \ldots\},$$

the flavor and brand should receive the highest attentions such that the description 'translates' to *strawberry Haggan-Dazs ice cream*. It becomes clear at this point that our goal is aligned with the attention mechanism proposed in neural machine translation literature [11, 25], where a sentence is represented by a weighted sum of the individual word representation. The attention weight assigned to each entry reflects their relative importance when translating the next token. Therefore, we propose an adapted self-attention network as the relation extractor to effectively retrieve target relations from the noisy product description and customer activity data.

Secondly, we resort to the multi-task learning with multi-modal data to handle the **sparsity** issue. It has become a common practice for industrial applications to leverage the useful information across related tasks to make up for the data sparsity in individual task [12, 29, 37]. In e-commerce, available data sources often include customer view, purchase, search, substitution records, as well as product descriptions and hierarchical category information. The tasks of learning/retrieving different product knowledge relations can be naturally brought together under the *propagation rule* that the substitutable or similar products are more likely to have similar `complement`, `co-view`, `substitute` products as well as descriptions and search terms. This observation motivates us to design the multi-task learning schema leveraging the *propagation rule*.

Last but not least, we propose to handle the tree-structured hierarchical category by embedding them onto the **Poincaré ball** [15]. Several papers including [33] have discussed methods for including structured entities in KG embedding. However, they still operate on Euclidean space. While Euclidean space lacks the flexibility to embed tree-structured data properly, some hyperbolic spaces have favorable manifold structures [6]. We briefly introduce the technical background for Poincaré embedding in Section 5.

We provide comprehensive comparisons between PKG and ordinary KG in Section 3 and illustrate the critical insight of how *translation* can be carried out via distributed representations in Section 4. We introduce our PKG embedding approach in Section 6. In Section 7, we first demonstrate how KG embedding methods can fail when directly applied to PKG with a real-world e-commerce dataset from *grocery.walmart.com*. We then show the superior performances of the proposed approach with product knowledge completion tasks and downstream e-commerce tasks.

2 CONTRIBUTION

We conclude several major contributions of this paper as follow.

- We propose the combination of product relations and PKG for e-commerce with comprehensive illustrations.
- We provide a systematic comparison between PKG and KG.
- We thoroughly explain the complex semantics of relations for PKG and show how to deal with the sophisticated relations using distributed representation.
- We propose a self-attention-based representation learning model for automatically retrieving relations and learning embeddings for PKG from both user activity data and product information, in an end-to-end fashion.
- We demonstrate the meaningfulness and usefulness of the outcome with real-world e-commerce dataset on knowledge completion, search ranking and recommendation tasks.

3 PRODUCT KNOWLEDGE GRAPH V.S. KNOWLEDGE GRAPH

The recent advances in KG embedding, which owes much to several publicly available KG database such as Freebase, DBpedia and YAGO, have led to successful applications in semantic parsing, information extraction, question answering and other NLP tasks [30]. PKG, though important for e-commerce, has received far less attention in existing literature. In this section, we list several critical components of KG embedding and compare them with PKG.

Data source. KG databases consist of established facts in the triplet form of *(head entity, relation, tail entity)*. The data source for constructing PKG have multiple modalities, including product catalog information (e.g. description, categories), raw user-product interaction records and others.

Model assumption. The core assumption for KG embedding is that the observed facts in KG database are well-established and plausible. It is not the case for PKG, where observations are much noisier and the facts have not been established.

Quantity of relation types. KG databases often contain thousands of relation types. In PKG, as we discussed before, the major relations can be adequately summarized by `complement`, `co-view`, `substitute`, `describe`, `search` and `IsA`.

Semantics of relation. Ordinary KG has semantically simple and unambiguous relations such as `BornIn`, `DirectorOf`, `HasWife`. Relations in PKG are semantically more complicated, as we illustrate with the below examples of `complement` for `TV`:

- (`Remote control`, `complement`, `TV`): accessory;
- (`TV mount frame`, `complement`, `TV`): structural attachment;
- (`Audio speaker`, `complement`, `TV`): enhancement;
- (`HDMI Cable switcher`, `complement`, `TV`): add-on.

In other words, product relations in PKG has much richer semantic meanings since products are designed over a broad range of purposes in the real world. Thus it is impractical to represent the relations merely by translation or/and project operations as their capacity of expressing complex semantic meanings are limited. Distributed representations, on the other hand, is another worthy option. Also, the relations in PKG are all N-to-N, as opposed to the many 1-to-N and 1-to-1 relations in ordinary KG [5].

Additional information. Both KG and PKG can use entity types, attributes and textual descriptions as additional information to enhance performances, and various methods has been developed for such purposes in KG embedding literature.

Logic rules. The first order Horn clause , e.g. `HasWife` $\Rightarrow$ `HasSpouse`, motives the logical inference in KG and is exploited by knowledge acquisition and inference. It also helps refine KG embedding [31]. Unfortunately, relations for PKG can often disobey the Horn clause. On the other hand, PKG do enjoy the *propagation rule* that substitutable products are more likely to have similar relations with other entities.

Downstream tasks. Most often KG embedding are applied in KG completion tasks [5]. Relation extraction and question answering are two other major directions. As for PKG, product knowledge completion is notably more important due to the **sparsity** issue in e-commerce data. Relation extraction and question answering can also find their counterparts in e-commerce settings, such as user understanding and searching. A key downstream application for PKG is recommender system. Although several work have proposed KG-enhanced recommendation methods, their application mainly focus on news and movie/book recommendation [27, 28, 35].

4 RELATION TRANSLATION VIA DISTRIBUTED REPRESENTATION

The distributed representation of words are learnt under the hypothesis that words which appear in a similar context have a similar representation. Translation models such as TransE are initially inspired by the linear analogies observed in the distributed representation (word embedding) outcome of *word2vec* [14], e.g. *king is to men as queen is to women*. In Euclidean space, the relation of linear analogy corresponds to vector translation: $\mathbf{z}_{\text{king}} - \mathbf{z}_{\text{men}} \approx \mathbf{z}_{\text{queen}} - \mathbf{z}_{\text{women}}$, where $\mathbf{z}_x$ denotes the embedding for entity (relation) x. The above observations motivate people to represent the relation `royal` by a translation vector $\mathbf{z}_{\text{royal}}$ such that:

$$\begin{cases} \mathbf{z}_{\text{king}} \approx \mathbf{z}_{\text{men}} + \mathbf{z}_{\text{royal}} \\ \mathbf{z}_{\text{queen}} \approx \mathbf{z}_{\text{women}} + \mathbf{z}_{\text{royal}}. \end{cases}$$

Indeed, for ordinary KG where relations are established in fine-grained level, it is convenient and straightforward to define a translation vector (operating on relation-specific spaces) for each relation. However, for PKG where the relations are much more complicated, it is impractical to expect a single translation vector $\mathbf{z}_{\text{complement}}$, even equipped with relation-specific projections, to express various product complementary semantics (e.g. functional complete, structural attachment, enhancement) at the same time.

The linear analogy observed in distributed representation of words, on the other hand, is capable of expressing complex relation semantics. A recent work proves from a principled manner that *word2vec* is also learning relational translation in the form of linear analogies [1]. The embeddings are intrinsically optimized to recover the patterns such as *king is to men as queen is to women*. We state a simplified version of theorem below.

Theorem. *If "entity y_1 is to entity x_1 as entity y_2 is to x_2", then:*

$$\mathbf{z}_{y_1} = \mathbf{z}_{x_1} + (\mathbf{z}_{y_2} - \mathbf{z}_{x_2}) + \epsilon$$

where ϵ is the translation error which can depend on the entities and model parameters.

A significant consequence of the theorem is that for each relation (such as `complement`), the underlying relational semantics, however complicated, can be constructed via entity embeddings that are learned in a similar fashion to *word2vec*. For instance, instead of explicitly define a translation vector for `AccessoryTo`, which is a special case of `complement`, we can expect the model to learn from customer purchase records that: $\mathbf{z}_{\text{AccessoryTo}} \equiv \mathbf{z}_{\text{Xbox}} - \mathbf{z}_{\text{handle}}$ such that $\mathbf{z}_{\text{remote control}} + \mathbf{z}_{\text{AccessoryTo}} \approx \mathbf{z}_{\text{TV}}$. We leave the detail of training the entity embeddings to Section 6.

5 RELATED WORK

In this section, we walk through several KG embedding models that we use for building baselines, the background of distributed representation, self-attention and Poincare embedding. To be consistent with original literature we use $\mathbf{r}$ to represent relation embedding.

KG embedding models. Let $\mathbf{h}$ and $\mathbf{t}$ denote head and tail entity of a triplet (`head entity`, `relation`, `tail entity`). *TransE* uses the distance function for learning from the triplets: $d_r(\mathbf{h}, \mathbf{t}) = \sum -\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|$, where $\|.\|$ can be either ℓ_1 or ℓ_2 norm. *TransH* further considers relation-specific entity embeddings, i.e $\mathbf{h}^r_\perp$ and $\mathbf{t}^r_\perp$, defined as entities projected onto the relation-specific hyperplanes. The distance is defined similarly as: $d_r(\mathbf{h}, \mathbf{t}) = \sum -\|\mathbf{h}^r_\perp + \mathbf{r} - \mathbf{t}^r_\perp\|$. *TransR* shares a similar idea to *TransH* but with the difference that instead of projecting entities onto hyperplanes, *TransR* considers relation-specific linear subspaces. *TransD* puts further constraints on the

parametrization of relation-specific linear subspaces to achieve higher efficiency.

RESCAL uses the semantic matching distance functions: $d_r(\mathbf{h},\mathbf{t}) = \sum \mathbf{h}^\top \mathbf{M}_r \mathbf{t}$, where M_r is a matrix associated with relation r. *DistMult* simplifies *RESCAL* by restricting M_r to be diagonal. The *Holographic Embedding* (*HolE*) employs the circular correlation operation $\star$ to combine the advantages of *RESCAL* and *DistMult*, where the distance function is defined via $d_r(\mathbf{h},\mathbf{t}) = \sum \mathbf{r}^\top(\mathbf{h}\star\mathbf{t})$.

All the above methods operates on the real space, and to extend the embedding approach of *DistMult* to complex-valued vectors, *ComplEx* defines distance as: $d_r(\mathbf{h},\mathbf{t}) = \sum \mathrm{Re}\big(\mathbf{h}^\top \mathrm{diag}(\mathbf{r})\bar{\mathbf{t}}\big)$, where Re(.) extracts the real part and $\bar{\mathbf{t}}$ is the conjugate of $\mathbf{t}$.

Distributed representation of *word2vec*. *Word2vec* uses the skip-gram model to learn the distributed representation of words. The score function is defined as the total log-probability of observing the words given their contexts: $S = \sum_i \sum_{j\in \mathrm{Context}(i,c)} \log p(e_i|e_j)$, where $\mathrm{Context}(i,c)$ is the set of neighbours of entity i within a window of size c. Each probability term is computed with softmax function such that

$$p(e_i|e_j) = \frac{\exp\big((\mathbf{z}_i^O)^\top(\mathbf{z}_j^I)\big)}{\sum_k \exp\big((\mathbf{z}_k^O)^\top(\mathbf{z}_j^I)\big)}, \tag{1}$$

where $\mathbf{Z}^I$ and $\mathbf{Z}^O$ are the "input" and "output" word embeddings. To avoid computing the summation term over the whole vocabulary, hierarchical softmax and negative sampling are often employed as approximation methods for the computational efficiency [14].

Self-attention. The key idea behind the attention mechanism is that only part of the input sequence is relevant to the output, and the model should pay more attention on the relevant part. As an add-on component, attention mechanism has been widely applied to image captioning and machine translation. Recently, the purely attention-based sequence-to-sequence *Transformer* model achieves state-of-the-art performances in machine translation tasks [25]. After discarding the RNN structures for modelling sequences, *Transformer* relies heavily on the self-attention module. The authors also demonstrate the improved model interpretation and computation efficiency for the *Transformer* [25].

Poincaré embedding. Embedding entities in the Euclidean vector spaces does not account for their latent hierarchical structures. Hyperbolic space such as the Poincaré Ball, on the other hand, can represent hierarchy structures and similarity more parsimoniously. We refer the readers to [6] for a comprehensive introduction on hyperbolic embedding. The main difference induced by operating on Poincare Ball lies in the distance metric, which is defined as:

$$d_{\text{Poincaré}}(e_i,e_j) = \mathrm{arcosh}\big(1 + 2\frac{\|\mathbf{z}_i - \mathbf{z}_j\|_2^2}{(1-\|\mathbf{z}_i\|_2^2)(1-\|\mathbf{z}_j\|_2^2)}\big). \tag{2}$$

Loss functions developed upon (2) can be optimized via Riemannian optimization methods according to the Riemannian structure of Poincaré Ball. Efficient implementation of stochastic gradient descent on Riemannian manifolds has also been developed [4].

6 METHODOLOGY

When learning PKG embedding according to the proposed approach, we use a generic e-commerce dataset that consists of session-based purchase and view sequences, product substitution records, the search-and-click records, product descriptions and hierarchical category labels. We point out that the proposed approach is still applicable when one or more of the above data modalities are unavailable. The notations are summarized in Table 1.

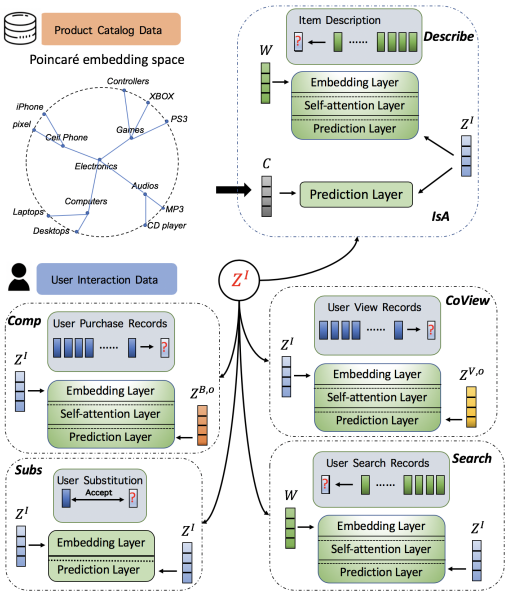

Figure 2: Architecture for learning PKG embedding.

Notation	Description
$d \in \mathbb{N}$	entity embedding dimension
$l \in \mathbb{N}$	the maximum sequence length for predicting target entity
$\mathcal{I}$,$\mathcal{W}$,$\mathcal{C}$	item (product), word and category label set
$\mathcal{B}$, $\mathcal{V}$, $\mathcal{S}$	customer session-based co-buy, co-view and substitution acceptance records, for instance, $\mathcal{B}_i = (\mathcal{I}_{i_1}, \mathcal{I}_{i_2}, \ldots, \mathcal{I}_{\lvert\mathcal{B}_i\rvert})$, $\mathcal{S}_j = (\mathcal{I}_{j_1}, \mathcal{I}_{j_2})$
$\mathcal{Q}$, $\mathcal{D}$, $\mathcal{L}$	query (search), description, category labels for products, i.e. $\mathcal{Q}_{\mathcal{I}_j} \subset \mathcal{W}$, $\mathcal{D}_{\mathcal{I}_j} \subset \mathcal{W}$, $\mathcal{L}_{\mathcal{I}_j} \subset \mathcal{C}$
$\mathbf{Z}^I \in \mathbb{R}^{\lvert\mathcal{I}\rvert \times d}$	product "input" embedding similar to that of *word2vec*, e.g. $\mathbf{Z}^I_{\mathcal{I}_j}$ denotes product "input" embedding for item $\mathcal{I}_j$
$\mathbf{Z}^{B,O}, \mathbf{Z}^{V,O} \in \mathbb{R}^{\lvert\mathcal{I}\rvert \times d}$	product "output" embeddings for modelling co-buy and co-view records
$\mathbf{W} \in \mathbb{R}^{\lvert\mathcal{W}\rvert \times d}$	word entity embeddings
$\mathbf{C} \in \mathbb{R}^{\lvert\mathcal{C}\rvert \times d}$	category entity embeddings
$\mathbf{P} \in \mathbb{R}^{l \times d}$	positional encoding matrix for self-attention

Table 1: Notation

6.1 Modelling `substitute` Relation

According to the *propagation rule* mentioned in Section 1 and 3, the `substitute` relation is the key to bring together the various

relations for PKG embedding. Recall the distributional hypothesis which states that the contextually similar words have similar representations. The *propagation rule* implies, and it is indeed observed in *word2vec* outcome, that similar words lay closer to each other in the word "input" embedding space. In analogy, substitutable products are also expected to have similar product "input" embeddings. When the customer substitution records are available, we can directly model the product "input" embeddings $\mathbf{Z}^I$. Since the `substitute` relation is symmetric, which means if A can substitute B then B can also substitute A, for each substitution accepted by customers we define the substitution score as:

$$S_{\text{sub}} = \sum_{(e_1,e_2)\in\mathcal{S}} \log p(e_1,e_2) \equiv \sum_{(e_1,e_2)\in\mathcal{S}} \log \frac{\exp\left((\mathbf{Z}^I_{e_1})^\intercal \mathbf{Z}^I_{e_2}\right)}{\sum_{i\in\mathcal{I}} \exp\left((\mathbf{Z}^I_{i})^\intercal \mathbf{Z}^I_{e_2}\right)}. \tag{3}$$

We shall see in the next sections that $\mathbf{Z}^I$ builds the bridge between modelling various relations by exploiting the *propagation rule*.

6.2 Self-attention Mechanism for `complement`, `co-view`, `describe` and `search` Relations

To extract `complement`, `co-view`, `search` and `describe` relations from the noisy customer purchase, view, search activities and product descriptions respectively, we use self-attention module with problem-specific modifications. For notation simplicity, in this part (Section 6.2) we use $\mathbf{Z}^O$ to denote product "output" embedding as a whole. Also, the sequence length l may vary across tasks.

Embedding layer for self-attention. The embedding layer takes an ordered sequence of entities (products or words) as input. To model positional information, self-attention uses positional encoding such that each position k maps to a vector $\mathbf{P}_k \in \mathbb{R}^d$. The entity sequence is truncated at the maximum length l, which we denote by $\mathbf{e} = (e_1, \ldots, e_l)$. The embedding layer takes the sum of entity embeddings (either $\mathbf{Z}^I$ or $\mathbf{Z}^O$) and their corresponding position encoding from $\mathbf{P}$, and the output is given by:

$$\mathbf{E}^I_{\mathbf{e}} = [\mathbf{Z}^I_{e_1} + \mathbf{P}_1, \ldots, \mathbf{Z}^I_{e_1} + \mathbf{P}_l]^\intercal \in \mathbb{R}^{l\times d}$$

and $\mathbf{E}^O_{\mathbf{e}} = [\mathbf{Z}^O_{e_1} + \mathbf{P}_1, \ldots, \mathbf{Z}^O_{e_1} + \mathbf{P}_l]^\intercal \in \mathbb{R}^{l\times d}$.

Self-attention layer. We use the scaled dot-product attention as building block, which is defined as:

$$\text{Attn}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\Big(\frac{\mathbf{Q}\mathbf{K}^\intercal}{\sqrt{d}}\Big)\mathbf{V}, \tag{4}$$

where $\mathbf{Q}$ represents the "queries", $\mathbf{K}$ the "keys" and $\mathbf{V}$ the "values". Since each row of $\mathbf{Q},\mathbf{K},\mathbf{V}$ correspond to an entity, the dot-product attention layers outputs the weighted sum of the entity embeddings in $\mathbf{V}$, where the weights reflects the pairwise "query-key" interactions in the entities sequence.

Given our distributed representation setting, it is intuitive to consider using $\mathbf{E}^I$ as the "queries" and $\mathbf{E}^O$ as the "keys" since they embed the contextual and positional relatedness information of the entity pairs. We also choose to use $\mathbf{E}^I$, the entity "input" embedding with positional encoding, as the "values" $\mathbf{V}$, for reasons which we will later explain. So the output of our attention layer is given by: $\mathbf{H} = \text{Attn}(\mathbf{E}^I, \mathbf{E}^O, \mathbf{E}^I)$. To see how self-attention layer is capable of assigning weights to entities such that the output only focus on related part of the sequence, we point out that $\mathbf{H}$ can be alternatively expressed as:

$$\mathbf{H}_i = \sum_j \alpha_{ij} \times \mathbf{E}^I_j,$$

with the weights $\alpha_{ij} \equiv (\mathbf{E}^I_i)^\intercal \mathbf{E}^O_i$ capturing the contextual and positional relation between entity e_i and e_j.

However, taking direct inner-product between $\mathbf{E}^I$ and $\mathbf{E}^O$ does not take account of the interactions between different latent dimensions. It can impair the expressiveness of the self-attention layer. Instead, we add a two-layer point-wise feed-forward network to the entity "input" and "output" embeddings before passing them to the dot-product attention, i.e.

$$\begin{aligned} &\text{FFN}(\mathbf{E}_i) \equiv \text{ReLU}(\mathbf{E}_i\boldsymbol{\Theta}_1 + \mathbf{b}_1)\boldsymbol{\Theta}_2 + \mathbf{b}_2, \\ &\mathbf{F}^I_i = \text{FFN}(\mathbf{E}^I_i),\ \mathbf{F}^O_i = \text{FFN}(\mathbf{E}^O_i),\ \forall i \in \{1,\ldots,l\}. \end{aligned} \tag{5}$$

where $\boldsymbol{\Theta}_1, \boldsymbol{\Theta}_2 \in \mathbb{R}^{d\times d}$ are parameter matrices and $\mathbf{b}_1, \mathbf{b}_2 \in \mathbb{R}^d$ are bias terms. So the complete self-attention layer is given by:

$$\mathbf{H} = \text{Attn}(\mathbf{e}) \equiv \text{Attn}(\mathbf{F}^I_{\mathbf{e}}, \mathbf{F}^O_{\mathbf{e}}, \mathbf{E}^I_{\mathbf{e}}), \mathbf{H} \in \mathbb{R}^{l\times d}. \tag{6}$$

Prediction layer. When modelling with customer purchase and view data, it is straightforward to use the previously purchased/viewed product sequences to predict next purchase/view product. Similar to that of *word2vec* in (1), we estimate the $p(e_{l+1}|e_1,\ldots,e_l)$ where e can be either word token or product.

To compute $p(e_{l+1}|e_1,\ldots,e_l)$, we need an aggregated representation of the context sequence $\mathbf{e} = (e_1,\ldots,e_l)$. By plugging in our self-attention layer as sequence aggregator, we obtain:

$$\begin{aligned} \log p(e_{l+1}|\mathbf{e}) &= (\mathbf{Z}^O_{e_{l+1}})^\intercal \text{Attn}(\mathbf{e}) - \log \sum_{i\in\mathcal{I}} \exp\left((\mathbf{Z}^O_i)^\intercal \text{Attn}(\mathbf{e})\right) \\ &= (\mathbf{Z}^O_{e_{l+1}})^\intercal \sum_{j=1}^{l} \alpha_{ij}\mathbf{Z}^I_{e_j} - \log \sum_{i\in\mathcal{I}} \exp\left((\mathbf{Z}^O_i)^\intercal \sum_{j=1}^{l} \alpha_{ij}\mathbf{Z}^I_{e_j}\right) + C, \end{aligned} \tag{7}$$

where C is some constant term unrelated to learning entity embeddings. We point out that (7) is an extension of (1) where the context sequence is aggregated by the attention weights that are part of our self-attention layer. The inner product terms in (7) also explains why we use the entity "input" embedding $\mathbf{Z}^I$ as the "value" matrix in the dot-product attention. By doing so, we are still modelling with the inner products between the product "input" and "output" embeddings ($\mathbf{Z}^I$ and $\mathbf{Z}^O$).

The description and search data can be modelled with the same prediction setting. Given the desription/search words, we predict the target product. Since $\mathbf{Z}^I$ carries the substitutable product information, under the *propagation rule* products that are closer in terms of $\mathbf{Z}^I$ should have similar descriptions and search terms. Therefore, we use the word embedding to predict product "input" embedding:

$$\log p(e_{l+1}|\mathbf{e}) = (\mathbf{Z}^I_{e_{l+1}})^\intercal \sum_{j=1}^{l} \alpha_{ij}\mathbf{W}^I_{e_j} - \log \sum_{i\in\mathcal{I}} \exp\left((\mathbf{Z}^I_i)^\intercal \sum_{j=1}^{l} \alpha_{ij}\mathbf{W}^I_{e_j}\right) + C, \tag{8}$$

where the entity sequence $\mathbf{e}$ is from search or description.

Score functions. With the outputs from the prediction layer, the score functions can be given in a straightforward manner, e.g.

$$S_{\text{complement}} = \sum_{(e_{l+1},\mathbf{e})\in\mathcal{B}} \log p(e_{l+1}|\mathbf{e}). \tag{9}$$

The score functions of $S_{\text{co-view}}$, S_{describe}, S_{search} are also computed according to the data of $\mathcal{V}$, $\mathcal{D}$, $\mathcal{S}$ in the same fashion. We do not write them down to avoid unnecessary repetitions.

6.3 Poincaré Embedding for Category Hierarchy and IsA Relation

To learn the distributed representations of the hierarchical categories, which are originally tree-structured symbols, we employ the Poincaré embedding for the best practice. When a category symbol c_1 is a child node of c_2, their distance in terms of $d_{\text{Poincaré}}$ is supposed to be small. To be consistent with the score functions for learning other relations, we also estimate $p(c_1|c_2)$ as a classification task with softmax function:

$$p(c_1|c_2) = \frac{\exp\big(-d_{\text{Poincaré}}(c_1, c_2)\big)}{\sum_{c\in C}\exp\big(-d_{\text{Poincaré}}(c, c_2)\big)}. \tag{10}$$

Notice that the category label embddings $\mathbf{C}$ are independent from the product and word embeddings in other tasks. Therefore, it is convenient to pre-train the Poincaré embeddings for the category labels and then pass them to the final task of modelling IsA relation.

Now that the category embeddings $\mathbf{C}$ are fixed, the IsA relation can be directly formulated as a multi-class classification problem, where we use product embedding to predict their category labels. Again we use the product "input" embedding $\mathbf{Z}^I$ for the same reason as using $\mathbf{Z}^I$ for modelling describe and search. For consistency we also use softmax function in the score function S_{IsA}:

$$S_{\text{IsA}} = \sum_{i\in|\mathcal{I}|}\sum_{j\in\mathcal{L}_i}\log\frac{\exp\big(\mathbf{C}_j^\top\mathbf{Z}_i^I\big)}{\sum_{c\in C}\exp\big(\mathbf{C}_c^\top\mathbf{Z}_i^I\big)}. \tag{11}$$

We point out that the (log-)normalization terms in (3), (7), (8), (10) and (11) can all be efficiently approximated by negative sampling.

6.4 Multitask Training

Although the score functions associated with each task have simple forms and unequivocal interpretations, it is unclear how to combine them into an overall score for optimal results. Despite that many methods have been proposed for multi-task learning with deep neural networks such as *GradNorm* [7], *MGDA* [21] and *uncertainty weighting* [9], they assume a shared network structure across tasks. In our setting, however, only the product "input" embedding is shared in all tasks. Also, most of the above approaches require computing all the gradients during each update, which is infeasible in our case due to the vast number of free parameters in our method. Similarly, the various searching algorithms for detecting the best-weighted combination of individual score functions [38] are also computationally impractical due to the large scale of real-world e-commmerce data. Interestingly, a recent work observes that there is no clear consensus on correctly training multi-task model in an NLP setting similar to ours [22].

As a compromise, we choose the simple yet effective training method described in [20]. After each training epoch, a task is randomly selected, and a batch of dataset associated with this task is sampled for training. The sampling is done in a weighted fashion, so the probability of sampling a task is proportional to the relative size of each dataset. The sampling-then-training process is repeated until the metrics for each task stop improving on validation dataset.

Task	$p(t\|h)$ / $p(t\|\mathbf{h})$
substitute	$\propto \exp\big((\mathbf{Z}_t^I)^\top\mathbf{Z}_h^I\big), h, t \in \mathcal{I}$
complement	$\propto \exp\big((\mathbf{Z}_t^{B,O})^\top\mathbf{Z}_h^I\big), h, t \in \mathcal{I}$
co-view	$\propto \exp\big((\mathbf{Z}_t^{V,O})^\top\mathbf{Z}_h^I\big), h, t \in \mathcal{I}$
search describe	$\propto \exp\big((\mathbf{Z}_t^I)^\top\text{Attn}(\mathbf{h})\big), \mathbf{h} \subset \mathcal{W}, t \in \mathcal{I}$
Isa	$\propto \exp\big((\mathbf{Z}_t^I)^\top\mathbf{C}_h^I\big), h \in \mathcal{C}, t \in \mathcal{I}$
recommend	$\propto \exp\big((\mathbf{Z}_t^{B,O} + \mathbf{Z}_t^{V,O})^\top\text{Attn}(\mathbf{h})\big), \mathbf{h} \subset \mathcal{I}, t \in \mathcal{I}$

Table 2: Prediction for each task (relation) according to trained embeddings.

6.5 Prediction for Downstream Tasks

The downstream PKG tasks, such as knowledge completion, search ranking and recommendation, require prediction with the learned PKG embeddings. Task details are discussed in Section 7.4. When predicting or ranking the candidate tail entities t given head entity h (for complement, co-view, substitute) or entities $\mathbf{h}$ (for search, describe, recommendation), the objective is the $p(t|h)$ or $p(t|\mathbf{h})$ which we can compute according to Table 2.

7 EXPERIMENT AND RESULT

We design the experiments to answer the following questions:

Q1: Is the proposed multi-task learning schema reasonable?

Q2: Other than knowledge completion, how does the PKG embedding benefit downstream e-commerce tasks?

Q3: Why KG embedding methods fail to work when directly applied to raw e-comerce dataset?

Q4: If a product relation graph is available for KG embedding methods, can the proposed approach still outperform the baselines?

7.1 Dataset

We evaluate our approach on a real-world e-commerce dataset obtained from *grocery.walmart.com*, the largest online grocery shopping platform in the U.S. Product catalog information of the dataset are summarized in Table 3.

products	The dataset contains ~140,000 common grocery products covering a broad range from food to appliances
description	Each product is provided with a short description (containing name and brand) as shown on the website. Usually the descriptions have 20 - 100 words.
category hierarchy	Each product is assigned to a cateogry hierarchy in the form of {subcategory, category, department, super-department}, and there are 1,198 subcategories, 228 categories, 28 departments and 9 super-departments.

Table 3: Summary of product catalog data.

Session data. We are provided with ~40 million session records with the views, purchases, search queries and the products that are clicked according to the search queries.

Substitution data. When products went out of stock, substitutions are recommended for the customers where they can choose to accept or deny. The dataset consists of the accepted substitutions. Around 70,000 products have been substituted.

Preprocess. We remove products that have less than ten total appearances (purchase, view, searched, substitution), which leaves us ~100,000 products. Words with less than three appearances (description, search query) are removed. We point out that our approach is capable of handling **sparsity** issue, but to make sure that the baselines can work properly, we filter out infrequent entities.

Product relation graph (PRG). We are able to construct the product relation graph for `complement`, co-view and `substitute`. We first build a weighted graph such that the edge between node A and B is the number of session that these two products have been co-viewed, co-purchased or substituted. For instance, we use $\mathbf{X}$ to denote the resulting adjacency matrix for co-purchase, so

$$\mathbf{A}_{ij} = \#\{k|(\mathcal{I}_i, \mathcal{I}_j) \subset \mathcal{B}_k\}.$$

The normalized adjacency matrix can be computed via

$$\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}},$$

where $\mathbf{D}$ is a diagonal matrix with $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$. We then train a *biased random walk* with the normalized adjacency matrix and obtain top-K related neighbors for each product. After initial inspections we find that $K = 20$ gives reasonable results for most products. The parameters are then tuned such that the top-20 related products achieve best link prediction performance on a hold-out subset in terms of *hitting rate*. Then we treat the top-20 related products for each product as known facts and obtain the product relation graph $\mathbf{G}_{\text{buy}}$. $\mathbf{G}_{\text{view}}$ and $\mathbf{G}_{\text{subs}}$ can be constructed in the same way.

7.2 Baseline Methods

To answer **Q3** and **Q4**, we implement classic KG embedding methods on both raw data and data enhanced with $\mathbf{G}_{\text{buy}}$, $\mathbf{G}_{\text{view}}$ and $\mathbf{G}_{\text{subs}}$. When learning KG embedding from raw data, we directly enumerate all triplets from product catalog information, session data and substitution data, e.g.

$$\mathbf{X}_{\text{view}} = \{(\mathcal{I}_1, \texttt{co-view}, \mathcal{I}_2) \mid (\mathcal{I}_1, \mathcal{I}_2) \subset \mathcal{V}_i, 1 \le i \le |\mathcal{V}|)\}$$
$$\mathbf{X}_{\text{sub}} = \{(\mathcal{I}_1, \texttt{substitute}, \mathcal{I}_2) \mid (\mathcal{I}_1, \mathcal{I}_2) \in \mathcal{S}_i, 1 \le i \le |\mathcal{S}|)\}.$$

$\mathbf{X}_{\text{buy}}$, $\mathbf{X}_{\text{describe}}$, $\mathbf{X}_{\text{search}}$ and $\mathbf{X}_{\text{IsA}}$ are constructed in the same way.

Although the above construction mechanism creates a massive number of triplets, it is not clear how to effectively implement downsampling. Therefore, we only implement **TransE**, **TransD** and **DistMult** on this generated dataset (**No PRG**).

With the pre-trained product relation graph $\mathbf{G}_{\text{buy}}$, $\mathbf{G}_{\text{view}}$, $\mathbf{G}_{\text{subs}}$ replacing $\mathbf{X}_{\text{buy}}$, $\mathbf{X}_{\text{view}}$ and $\mathbf{X}_{\text{sub}}$, the number of triplets is considerably decreased in the enhanced dataset (**With PRG**). Therefore we train **TransE**, **TransH**, **TransR**, **TransD**, **RESCAL**, **DistMult** and **ComplEx** for comprehensive comparisons.

Since the KG embedding methods are not designed for recommendation, we further include **Factorization Machine** (**FM**) [18], **Bayesian Personalized Ranking** (**BPR**) [19], **Prod2vec** [24] and **Triple2vec** [26] as baselines. We choose these methods among others because they also learn latent representations of products.

7.3 Implementation Details

To be consistent with the original implementation, we also use stochastic gradient descent for the KG embedding models with learning rate selected among {0.001, 0.005, 0.01, 0.1}. The margin γ for the translation models are selected among {1, 2, 5, 10}. We also choose the ℓ_1 or ℓ_2 norm according to validation performance measured by *hitting rate* on the knowledge completion task that we describe in Section 7.4. All the above models are trained with negative sampling, where for each positive triplet we sample *three* negative (corrupted) triplets. All the implementations are in *Tensorflow*, where we use the open-source framework for knowledge embedding (*OpenKE*) as reference for the KG embedding baselines.

For the proposed approach, we also use stochastic gradient optimizer with learning rate selected among {0.001, 0.005, 0.01, 0.1}. After initial data analysis, we choose $l_{\text{buy}} = 20$, $l_{\text{view}} = 50$, $l_{\text{describe}} = 200$, $l_{\text{search}} = 10$ as the maximum sequence lengths for the self-attention mechanism in corresponding tasks. The Poincaré embedding for category hierarchy is pre-trained with the default setting proposed in [15]. We also use three negative samples for our approach in the same manner as that of *word2vec*.

Finally, we set the dimension of embeddings and latent factors to 100 for all methods. The recommendation baseline methods are also tuned for best performance on *hitting rate* in validation data.

7.4 Tasks and Evaluation Metrics

Knowledge completion. We focus on the link prediction task for knowledge completion and entity classification. While knowledge completion examines the **usefulness** or PKG embedding, entity classification examines their **meaningfulness**. For knowledge completion, we evaluate the prediction of tail entity when given head entity and relation. The relation is one of {`complement`, `co-view`, `substitute`}. We use top-10 hitting rate (**HIT@10**) and normalized discounted cumulative gain (**NDCG@10**) to evaluate the candidate rankings. For entity classification, we predict the *category* and *department* labels for products, using multi-class logistic regression with product "input" embedding as features. We report the **micro-F1** and **macro-F1** scores for the classification outcome.

Search ranking. We rank all products according to given search query and compute the top-10 recall (**R@10**) and mean average precision (**MAP@10**).

Recommendation. Since we only have session-based customer purchase records, we examine recommendation performance using with-in basket recommendation, i.e. given what the customer has purchases so far in the current and previous sessions, we predict the next impression (purchase or view) in the same session. We also report **HIT@10** and **NDCG@10**.

7.5 Training and Testing

The train-validation-test split is more involved because we are experimenting on several tasks with various baselines. Since the user activity data are timestamped, we split $\mathbf{X}_{\text{view}}$, $\mathbf{X}_{\text{buy}}$, $\mathbf{X}_{\text{sub}}$, $\mathbf{X}_{\text{search}}$ into 80%-10%-10% according to their chronological order. The product relation graphs are trained on $\mathbf{X}^{\text{train}}_{\text{view}}$, $\mathbf{X}^{\text{train}}_{\text{buy}}$, $\mathbf{X}^{\text{train}}_{\text{sub}}$ and validated on $\mathbf{X}^{\text{validate}}_{\text{view}}$, $\mathbf{X}^{\text{test}}_{\text{buy}}$ $\mathbf{X}^{\text{validate}}_{\text{sub}}$.

Task	Link prediction						Product classification			
Relation	complement		co-view		substitute		IsA (*category*)		IsA (*department*)	
Metric	**Hit@10** (a1)	**NDCG@10** (a2)	**Hit@10** (a3)	**NDCG@10** (a4)	**Hit@10** (a5)	**NDCG@10** (a6)	**micro-F1** (a7)	**macro-F1** (a8)	**micro-F1** (a9)	**macro-F1** (a10)
TransE (No PRG)	1.84	1.06	3.27	2.11	12.04	6.56	42.33	69.44	51.72	76.53
TransD (No PRG)	1.97	1.08	3.51	2.24	13.69	6.90	41.82	67.65	50.29	75.45
DistMult (No PRG)	3.47	1.88	6.58	3.41	20.64	9.96	53.75	74.69	61.42	80.73
TransE (With PRG)	3.65	1.82	6.95	3.90	30.22	13.41	45.43	74.93	55.89	81.81
TransH (With PRG)	4.13	1.79	6.88	2.89	30.37	13.56	41.94	64.09	50.12	72.95
TransR (With PRG)	6.06	2.35	8.17	3.43	31.25	14.88	46.37	72.74	53.95	74.11
TransD (With PRG)	4.26	1.95	7.03	2.97	20.71	9.86	50.36	71.02	59.62	82.43
RESCAL (With PRG)	1.64	0.97	1.63	0.87	12.46	5.76	62.89	86.27	72.27	90.97
DistMult (With PRG)	5.69	2.47	9.64	4.05	30.64	12.25	*68.25*	*94.23*	72.09	92.94
ComplEx (With PRG)	*7.81*	*3.36*	*12.38*	*5.77*	*31.25*	*12.60*	67.46	94.02	*72.54*	*97.71*
Our approach	***14.53***	***7.67***	***20.84***	***10.26***	***34.58***	***14.77***	***68.62***	***95.17***	***74.61***	***99.60***

Table 4: Testing performances on the knowledge completion tasks. The results are average over three runs and reported in %. The best performing method in each row is boldfaced, and the second best method in each row is underlined. The labels beneath the metric name corresponds to the labels in Figure 3.

Knowledge completion. We do the train-validation-test split on product relation graphs G_{buy}, G_{view} and G_{sub} into 80%-10%-10% with one condition that there is no isolated node in training graph. The proposed approach and baseline KG embedding methods are tested on G^{test}_{buy}, G^{test}_{view} and G^{test}_{sub} for `complement`, `co-view` and `substitute`. We also randomly select 10% of the products, mask all of their category information during training and predict their *category* and *department* in testing. For fair comparisons, when training our model, we remove all the related data from X_{view}, X_{buy}, X_{sub} and $X_{describe}$ that can cause information leak.

Search ranking. We test the performances on search ranking with X^{test}_{search}, on queries that have appeared in training dataset as well as new queries. For baseline KG embeddings methods, we take the average of entity word embeddings as query embedding.

Recommendation. For all the baselines considered, the within-session recommendation is trained on $X^{train}_{buy} \cup X^{train}_{view}$ and tested on $X^{test}_{buy} \cup X^{test}_{view}$.

7.6 Analysis on Multi-task Training

To show the effectiveness of our multi-task learning schema (**Q1**), we compute the correlations of the change in validation metric for each task during training, and compare the performance between: our training schema, training each task individually and multitask training with uniform task sampling. For any two tasks ($A \neq B$) with metric τ_A and τ_B, we compute $\rho_{A \to B} \equiv \text{corr}(\delta\tau_A, \delta\tau_B)$ according to the changes in τ_A and τ_B after each epoch trained for task A. The heatmap for ρ is provided in Figure 3, where we observe positive correlations among almost all tasks during the training. Specifically, the correlation between the `substitute` task and all other tasks are high. This indicates that our approach is benefiting from the *propagation rule*. The radar map in Figure 3 shows that the proposed training schema uniformly outperforms individual task training and the multi-task training under uniform task sampling.

7.7 Knowledge Completion and Downstream Tasks Performance

From Table 4, we see that the proposed approach outperforms all baselines in all tasks for knowledge completion, even when the KG embedding methods are enhanced with the pre-trained product knowledge graph (**Q4**). The fact that KG embeddings have subpar results when not using PRG suggests they rely heavily on well-established facts which are absent in the raw dataset (**Q3**). Notice that our approach exceeds in completing `complement` and `view` by significant margins. Since `complement` and `view` are the two relations which contain the most sophisticated semantics, we believe that our solution of using distributed representation performs better in capturing the richer relational semantics.

Model	**FM**	**BPR**	**prod2vec**	**triple2vec**	**Our approach**
Hit@10 (a11)	4.24	7.65	6.39	*11.30*	***13.72***
NDCG@10 (a12)	1.85	3.17	2.26	*4.83*	***5.79***

Task	**Encountered queries**		**New queries**	
Metric	**R@10** (a13)	**MAP@10** (a14)	**R@10** (a15)	**MAP@10** (a16)
TransE	8.74	3.26	5.11	2.62
TransH	10.43	4.28	6.85	2.79
TransR	15.82	6.77	10.33	4.09
TransD	13.69	6.17	9.50	4.24
RESCAL	7.71	2.98	4.25	1.93
DistMult	19.72	8.43	11.71	5.02
ComplEx	*21.58*	*10.04*	*12.46*	*5.15*
Our approach	***30.99***	***14.46***	***22.71***	***9.53***

Table 5: Testing performance (in %) on next-impression recommendation and the search ranking task for queries that have encountered in training and new queries. The results are averaged over three runs.

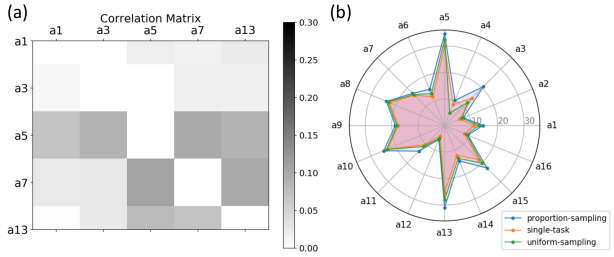

Figure 3: (a) Heatmap for the task correlations ρ (b) Test performances of different training schedules on all the tasks. The symbols for each task can be found beneath the metrics in Table 4 and 5. Micro-F1 and macro-F1 are divided by 4 for presentation purpose.

The results in Table 5 suggest that the PKG embeddings learned by our approach can also benefit downstream tasks such as search ranking and recommendation (**Q2**). In the search ranking task, our approach significantly surpasses baselines on both encountered queries and new queries. It is worth pointing out that for the next-impression prediction task, we also outperform the baseline models which are specially designed for recommendation.

8 CONCLUSION AND FUTURE WORK

We fully characterize the product knowledge graph and systematically compare it with the ordinary knowledge graph. To effectively learn PKG embedding with generic e-commerce dataset, we propose a self-attention-enhanced distributed representation learning method with an efficient multi-task training schema. The empirical results on the real-world data show that our approach outperforms KG embedding baselines in knowledge completion and delivers promising outcomes in downstream search ranking and recommendation. In the future, we will explore incorporating customer and customer knowledge (e.g. demographic information) into PKG to construct the customer-product knowledge graph that can stand out as the backbone for personalized e-commerce services.

REFERENCES

[1] Carl Allen and Timothy Hospedales. 2019. Analogies Explained: Towards Understanding Word Embeddings. In *International Conference on Machine Learning*. 223–231.

[2] Vito Walter Anelli, Pierpaolo Basile, Derek Bridge, Tommaso Di Noia, Pasquale Lops, Cataldo Musto, Fedelucio Narducci, and Markus Zanker. 2018. Knowledge-aware and conversational recommender systems. In *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM, 521–522.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[4] Silvere Bonnabel. 2013. Stochastic gradient descent on Riemannian manifolds. *IEEE Trans. Automat. Control* 58, 9 (2013), 2217–2229.

[5] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*. 2787–2795.

[6] Benjamin Paul Chamberlain, James Clough, and Marc Peter Deisenroth. 2017. Neural embeddings of graphs in hyperbolic space. *arXiv preprint arXiv:1705.10359* (2017).

[7] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2017. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. *arXiv preprint arXiv:1711.02257* (2017).

[8] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 687–696.

[9] Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7482–7491.

[10] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*.

[11] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* (2015).

[12] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1930–1939.

[13] Julian McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 785–794.

[14] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[15] Maximillian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. In *Advances in neural information processing systems*. 6338–6347.

[16] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic embeddings of knowledge graphs. In *Thirtieth Aaai conference on artificial intelligence*.

[17] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A Three-Way Model for Collective Learning on Multi-Relational Data.. In *ICML*, Vol. 11. 809–816.

[18] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International Conference on Data Mining*. IEEE, 995–1000.

[19] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 452–461.

[20] Victor Sanh, Thomas Wolf, and Sebastian Ruder. 2019. A hierarchical multi-task approach for learning embeddings from semantic tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 6949–6956.

[21] Ozan Sener and Vladlen Koltun. 2018. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*. 527–538.

[22] Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. 2018. Learning general purpose distributed sentence representations via large scale multi-task learning. *arXiv preprint arXiv:1804.00079* (2018).

[23] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*. 2071–2080.

[24] Flavian Vasile, Elena Smirnova, and Alexis Conneau. 2016. Meta-prod2vec: Product embeddings using side-information for recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 225–232.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[26] Mengting Wan, Di Wang, Jie Liu, Paul Bennett, and Julian McAuley. 2018. Representing and Recommending Shopping Baskets with Complementarity, Compatibility and Loyalty. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 1133–1142.

[27] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 417–426.

[28] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. DKN: Deep knowledge-aware network for news recommendation. In *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 1835–1844.

[29] Jingang Wang, Junfeng Tian, Long Qiu, Sheng Li, Jun Lang, Luo Si, and Man Lan. 2018. A multi-task learning approach for improving product title compression with user search log data. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[30] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering* 29, 12 (2017), 2724–2743.

[31] Quan Wang, Bin Wang, and Li Guo. 2015. Knowledge base completion using embeddings and rules. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

[32] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Twenty-Eighth AAAI conference on artificial intelligence*.

[33] Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2016. Representation Learning of Knowledge Graphs with Hierarchical Types.. In *IJCAI*. 2965–2971.

[34] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575* (2014).

[35] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 353–362.

[36] Yin Zhang, Haokai Lu, Wei Niu, and James Caverlee. 2018. Quality-aware neural complementary item recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM, 77–85.

[37] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1059–1068.

[38] Eckart Zitzler and Lothar Thiele. 1998. An evolutionary algorithm for multiobjective optimization: The strength pareto approach. *TIK-report* 43 (1998).