

Multi-Faceted Knowledge-Driven Pre-Training for Product Representation Learning

Denghui Zhang¹, Yanchi Liu, Zixuan Yuan¹, Yanjie Fu¹, Haifeng Chen¹, and Hui Xiong¹, *Fellow, IEEE*

Abstract—As a key component of e-commerce computing, product representation learning (PRL) provides benefits for a variety of applications, including product matching, search, and categorization. The existing PRL approaches have poor language understanding ability due to their inability to capture contextualized semantics. In addition, the learned representations by existing methods are not easily transferable to new products. Inspired by the recent advance of pre-trained language models (PLMs), we make the attempt to adapt PLMs for PRL to mitigate the above issues. In this article, we develop KINDLE, a Knowledge-driven pre-trainiNG framework for proDuct representation LEarning, which can preserve the contextual semantics and multi-faceted product knowledge *robustly* and *flexibly*. Specifically, we first extend traditional one-stage pre-training to a two-stage pre-training framework, and exploit a deliberate knowledge encoder to ensure a smooth knowledge fusion into PLM. In addition, we propose a multi-objective heterogeneous embedding method to represent thousands of knowledge elements. This helps KINDLE calibrate knowledge noise and sparsity automatically by replacing isolated classes as training targets in knowledge acquisition tasks. Furthermore, an input-aware gating network is proposed to select the most relevant knowledge for different downstream tasks. Finally, extensive experiments have demonstrated the advantages of KINDLE over the state-of-the-art baselines across three downstream tasks.

Index Terms—Product representation learning, product search, product matching, product classification, pre-trained language models

1 INTRODUCTION

NOWADAYS, e-commerce has become an integral part of our lives. According to the global sale statistics,¹ e-commerce is responsible for around \$3.5 trillion in 2019, and is expected to hit \$4.9 trillion by 2021. Among numerous data mining approaches for e-commerce, product representation learning (PRL) serves as a fundamental component, which aims to learn the distributional representations in a latent space for thousands of products. The latent representations have the advantages of dimensionality reduction, automatic feature learning, etc., which makes them useful for many downstream tasks, including product matching, search, and categorization [2], [22], [24].

Despite the prevalence, existing product representation learning approaches suffer from two noteworthy limitations: (i) *Insufficient ability in capturing contextualized semantics to deal with the polysemy problem*. The meaning of a word may vary in

different contexts. For instance, Fig. 1 shows a real example at Amazon.com, where the word *Monitor* appears in two different product titles, i.e., “Baby Monitor...” and “Dell... Monitor”. The former refers to webcam or camera while the latter is closer to *display* or *screen*. Such cases challenge the existing PRL approaches [2], [36], [41] that utilize similar ideas of word2vec [13] to learn product semantics, as the static word embedding cannot model the word sense dynamically from the context. These approaches may generate similar representations for two distinct products because they share some words, which actually have very different meanings in two products. (ii) *Lack of transferability from existing products to new products*. Most existing PRL approaches train a fixed embedding matrix for existing products, they cannot generalize well to new products, especially Out-of-Distribution (OOD) samples. Yet, for many e-commerce platforms where high volumes of new items are offered for sales everyday, stable and fast transferability is critical to the success of reliable services.

More recently, the pre-trained language models (PLMs) such as BERT and GPT-3, also known as contextualized word embeddings [6], [17], [20], have achieved great success in a broad range of natural language processing tasks. In contrast to the traditional word embedding methods, PLMs can greatly alleviate the polysemy problem as they encode semantic knowledge into a Transformer network, which takes the whole sequence as inputs and the word sense is conditioned on the contexts. Besides, the paradigm of pre-training and fine-tuning also enables better transferability for new data. Based on its merits, we make the attempt to adapt PLMs to the scenario of PRL and generate deep contextualized product representations. However, it is a non-trivial task due to the following challenges:

- *Highlighting the key information of a product under the PLM framework*. A natural way of generating representation is to feed the product title into a PLM and average all

1. <https://www.digitalcommerce360.com/article/global-ecommerce-sales/>

- Denghui Zhang, Zixuan Yuan, and Hui Xiong are with Rutgers University, Newark, NJ 07103 USA. E-mail: {dzhzhangai, yuanzx33033}@gmail.com, hxiong@rutgers.edu.
- Yanchi Liu and Haifeng Chen are with NEC Laboratories, America, Princeton, NJ 08540 USA. E-mail: {yanchi, haifeng}@nec-labs.com.
- Yanjie Fu is with the University of Central Florida, Orlando, FL 32816 USA. E-mail: yanjie.fu@ucf.edu.

Manuscript received 3 October 2021; revised 29 May 2022; accepted 6 August 2022. Date of publication 29 August 2022; date of current version 5 June 2023.

This work was supported by the National Science Foundation under Awards 1814510 and 2040799.

(Corresponding authors: Yanchi Liu and Hui Xiong.)

Recommended for acceptance by Z. Wang.

Digital Object Identifier no. 10.1109/TKDE.2022.3200921

Category	Product title
Video Monitors	Baby <i>Monitor</i> with Remote Pan-Tilt-Zoom Camera and 3.2" LCD Screen
Computer Monitors	Dell SE2419Hx 23.8" IPS Full HD (1920x1080) <i>Monitor</i>

Fig. 1. An example of the polysemy problem.

embeddings from the last layer of the Transformer. However, such flat representation lacks awareness of priority over key terms. As shown in Fig. 2, identifying key information (e.g., product type, accessories) of a product is critical for humans to distinguish different products, yet a hard task for machines. Therefore, how to highlight the “main points” under PLM is crucial for accurate product representation learning, however still remains unsettled.

- *Incorporating multi-faceted knowledge into PLMs.* As shown in Fig. 3, e-commerce platforms, like Amazon, eBay, and Walmart, contain heterogeneous product knowledge, such as product brand, product category, associated products, etc. In the literature, they have been used to enhance product representation and alleviate the vocabulary gap problem [29], [36], [41]. For example, people who search for “Dell Monitor” may also be interested in “Docking Station” although they are not literally similar. However, directly incorporating product knowledge into PLMs by multi-task learning can cause two kinds of discrepancy issues: (i) *Language and Knowledge Discrepancy*, that is, the discrepancy between language modeling and product knowledge preserving may cause discrepant optimizing direction for the underlying neural network. (ii) *Intra-knowledge Discrepancy*, i.e., multi-faceted product knowledge (e.g., attribute, category knowledge) is *heterogeneous*, directly preserving all together may also cause dispersed training objectives.

- *Handling the noise and sparsity issues of knowledge.* In most cases, product knowledge in e-commerce websites relies on data contributed by retailers, thus tends to be *noisy* and *sparse* [7]. Specifically, it happens due to the following reasons: (i) Inconsistent word usage. Different retailers often use synonyms (e.g., hood, hoodie) or abbreviation (e.g., Chocolate versus Choc) to refer the same concept. (ii) Missing attribute value. Retailers may not always list all structured fields including necessary attributes and categories. (iii) Diverse user behaviors. Some knowledge is purely driven by user behavior (e.g., product associations like co-buy), inevitably affected by outliers.

In this article, we propose KINDLE to address these challenges. Specifically, our model is novel in four aspects. (i) We extend the typical pre-training to two separate stages, i.e., language acquisition and knowledge acquisition, and use an extra knowledge encoder to preserve product knowledge

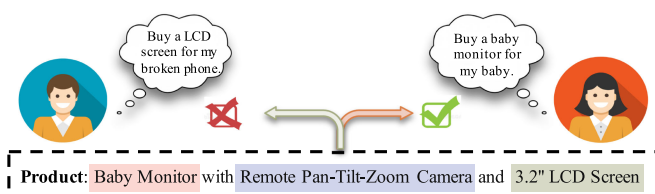


Fig. 2. An example of different parts of a product title. The existing PRL methods pay equal attention to different parts of the product title and become less effective when applied to tasks such as product matching and search.

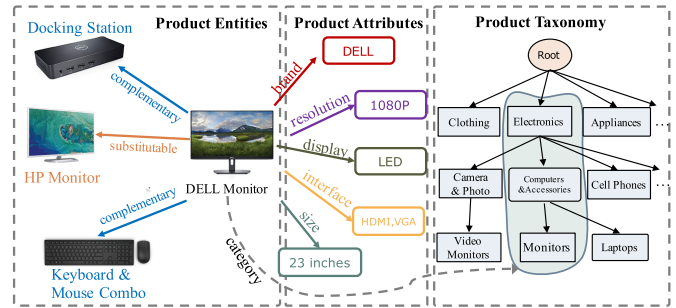


Fig. 3. A snapshot of Product Knowledge. The central entity represents a product. Surroundings are associated products, attributes (e.g., brand name, size), and categories.

alone. In this way, we alleviate the language and knowledge discrepancy issue. (ii) To highlight the key information of a product, a hierarchical Skeleton Attention (SA) compatible with PLM is proposed to capture the main points. (iii) During pre-training, the knowledge encoder along with Skeleton Attention first generates local product representations, which capture individual knowledge facets. Then we propose an input-aware gating network to fuse local representations into final representations during fine-tuning stage. It ensures automatically selecting relevant knowledge facets in different downstream tasks and mitigating the intra-knowledge discrepancy issue. (iv) To alleviate the noise and sparsity issues of product knowledge, we use heterogeneous embeddings instead of isolated class labels to represent knowledge elements in knowledge acquisition tasks. In this way the knowledge interrelatedness, i.e., label correlations, can be captured. Such interrelatedness of knowledge catalyzes self-calibration to its noise and sparsity, enabling a more robust learning process.

We conduct extensive experiments on three downstream tasks and their zero-shot counterparts. Our framework consistently outperforms state-of-the-art baselines in terms of various metrics (improves averagely 4.5% on product matching, 8.7% on personalized product search, and 4.7% on product classification). Ablation studies also show that different components of KINDLE are contributing and the model achieves the best performance when all the components are activated.

2 PRELIMINARIES

2.1 Problem Statement

In this article, we focus on *title-based product representation learning*. Formally, given a product represented by its title $p = \{w_i\}_{i=1}^n$, we aim to learn a model \mathcal{K} (based on PLMs) that maps the product title p into a dense representation $\mathcal{K}(p)$ which encodes essential information. Following PLMs, we adopt the paradigm of pre-training and fine-tuning. During pre-training, we leverage multiple resources to help $\mathcal{K}(p)$ encode product semantic information and additional multi-faceted product knowledge. To apply in downstream tasks such as product matching, search, classification, etc., \mathcal{K} will further be fine-tuned on task-specific datasets to encode task-related knowledge.

2.2 Multi-Faceted Product Knowledge

In this article, we consider three facets of product knowledge and represent them by Product Knowledge Graph (PKG). As

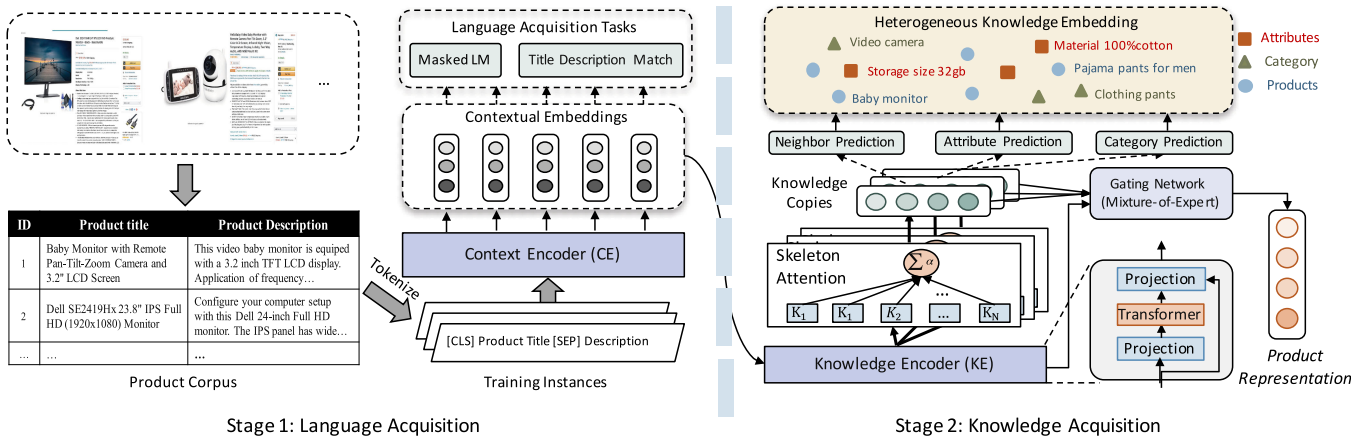


Fig. 4. Overview of KINDLE, a two-stage knowledge-driven pre-training framework. The first stage focuses on language acquisition while the second is responsible for multi-faceted knowledge acquisition.

shown in Fig. 3, three types of knowledge are loosely connected by a central product while inter-knowledge correlations are not presented. Besides, they differ vastly from each other in terms of volume and internal structure, thus being heterogeneous. Formal definitions of them are given below.

Definition 1. Neighbor Community Knowledge. Given a product p in PKG, we consider $E_p = \{p_i\}_{i=1}^m$, i.e., the set of surrounding products (similar or associated) as the neighbor community knowledge. Similar to social networks where we can learn about a user through his friends, a product can also be depicted and enriched by its associated products.

Definition 2. Attribute Knowledge. Given a product p in PKG, we consider the corresponding attribute set $A_p = \{a_i\}_{i=1}^l$ as the attribute knowledge. It provides more fine-grained semantic knowledge for product representations.

Definition 3. Category Knowledge. Given a product p and a pre-defined category hierarchy \mathcal{T} , we consider all categories it belongs to as the category knowledge, corresponding to nodes in \mathcal{T} . We distinguish category from attributes because there are rich structural correlations between different categories in \mathcal{T} and we preserve such structural priors by optimizing latent category representations with Poincaré Embedding [15].

2.3 Downstream Tasks

In this article, we mainly focus on three downstream tasks that rely on learning product representations first, namely, product matching, product classification, personalized product search. Detailed definitions of them can be found in Section 4.

3 PRE-TRAINING METHODOLOGY

In this section, we introduce our methodology in detail. We start with an overview of the proposed KINDLE framework. Then, we present the details of the underlying components.

3.1 Framework Overview

As shown in Fig. 4, KINDLE consists of two sequential stages: *language acquisition* and *knowledge acquisition*. In the first stage of pre-training, we rely on the language suite (consisting of Context Encoder and two language acquisition

tasks) to learn *contextual semantics* of the product domain. In the second stage, Context Encoder is fixed and its output is first transferred to Knowledge Encoder. Then followed by multiple Skeleton Attention layers, we generate local product representations (i.e., Knowledge Copies (KCs)), each capturing one facet of product knowledge. KCs are trained by heterogeneous embedding guided knowledge acquisition tasks to actually obtain *multi-faceted knowledge*. Final product representation is generated during fine-tuning stage by combining all KCs through a gating network, which is able to adjust weights according to the input product content.

3.2 Language Acquisition Suite

The language acquisition suite serves for modeling contextual semantics, consisting of input representation mapping, extended vocabulary, Context Encoder, and two language acquisition tasks. Please note that the language acquisition suite is optimized only during the first stage of pre-training.

3.2.1 Input Representation and Vocabulary

Given an input sequence (consisting of a product title $p = \{w_i\}_{i=1}^n$ and a description $d = \{w_i\}_{i=1}^m$), we first tokenize each word into smaller tokens (e.g. *headphone-2head, phone*) and use WordPiece embedding [34] to generate token embeddings $\mathcal{S} = \{\text{Tok}_i\}_{i=1}^{n+m+2}$ (two special tokens [CLS] and [SEP] are inserted to the start and middle positions respectively). For token vocabulary, we use the one of BERT since we adopt it as the backbone of Context Encoder. To deal with novel words in the product domain, we expand the vocabulary with 1,000 most frequent out-of-vocabulary (OOV) words in our corpus by directly adding them as tokens. Table 1 shows representative words of different product categories. Finally each token embedding is added with a position embedding and a segment embedding [5] to form the input representation $\mathcal{S}_I = \{\mathbf{E}_i\}_{i=1}^{n+m+2}$.

3.2.2 Context Encoder

Context Encoder (CE) takes the tokenized, vectored sequences as input and generates contextualized word embeddings. We use pre-trained BERT² as the backbone to build CE for

2. <https://huggingface.co/bert-base-uncased>

TABLE 1
High-Frequency OOV Words in Product Corpus

Departments	High-frequency words
Appliances	humidifier, dryer, whirlpool, cooktop, dispenser, bake, cfm, thermostat, heater
Automotive	oem, bumpers, durability, stickers, gmc, adhesive, waterproof, plated, relays
Clothing, Shoes and Jewelry	bracelets, sandal, durability, cushioning, sneaker, adidas, necklaces, legging
Electronics	adapter, charger, warranty, interconnects, headphones, headset, wifi, hdd
Grocery and Gourmet Food	diagnose, calories, gluten, gourmet, soybean, starch, hydrogenated, ounces, caffeine
Home and Kitchen	dishwasher, tablecloth, ornament, polyester, stylish, handmade, rugs, figurine, tumbler
Office Products	toner, indexes, envelopes, mfc, deskjet, pencils, laserjet, bookmark, eraser
Sports and Outdoors	hoodie, neoprene, pant, sweatshirts, mans, womens, wicking, zippered, breathability
Toys and Games	jigsaw, bandai, assorted, playset, quadcopter, pvc, monstercard, hasbro, bobbleheads

two notable benefits: (1) Inheriting rich language semantics of BERT obtained from massive Wikipedia articles. (2) Easy to adapt to our product domain and downstream tasks by post-training and adding task-specific layers. Formally, given an initialized input sequence $S_I = \{\mathbf{E}_i\}_{i=1}^{n+m+2}$, Context Encoder maps S_I to the contextualized embedding sequence $S_T = \{\mathbf{T}_i\}_{i=1}^{n+m+2}$. While each internal layer of Context Encoder is empowered by self-attention [31], each output word embedding \mathbf{T}_i is dependent on the entire input sequence $S_I = \{\mathbf{E}_i\}_{i=1}^{n+m+2}$, such design enables the output embeddings to be “contextualized”.

3.2.3 Language Acquisition Tasks

To preserve product semantics in CE, it is pre-trained by two *language-acquisition* tasks.

Task 1: Masked Language Model (MLM). MLM is a fill-in-the-blank task, where the model uses the context tokens around the mask token to predict what masked token should be (e.g., “Baby [MASK] with Remote Pan-Tilt-Zoom Camera” $\xrightarrow{\text{predict}}$ “Monitor”). When it converges, the model learns contextual semantics of each token and the last layer of Transformer is considered as contextual embeddings. Given an input sequence, we randomly mask 15% (the ratio is empirically borrowed from BERT) of tokens and reconstruct them using the last layer, we refer readers to [6] for detailed objective function of MLM.

Task 2: Title Description Matching (TDM). In addition to MLM, BERT uses the Next Sentence Prediction (NSP) task to enhance high-level semantic learning. Whereas, the notion of next sentence does not apply for product corpus as product title or description usually consists of one sentence. Hence, we propose TDM, a new sentence-level task in which we use the global classification token ([CLS]) of the last layer, to predict whether the input product title matches the

description (i.e., refer to the same product). Accordingly, we modify the input a little during pre-training, i.e., the input product title is paired with its correct product description for 50% of the time (labeled as Match). And for the rest 50% of the time, we replace the correct product description with a corrupted description that is randomly selected from a different category (labeled as NotMatch). The objective function of TDM is summarized as

$$q_i = \frac{\exp(\mathbf{T}_{CLS,i}^\top \mathbf{w}_m)}{1 + \exp(\mathbf{T}_{CLS,i}^\top \mathbf{w}_m)} \quad (1)$$

$$\mathcal{L}_{TDM} = - \sum_{i \in \mathcal{D}} [y_i \log(q_i) + (1 - y_i) \log(1 - q_i)], \quad (2)$$

where \mathbf{w}_m is the parameter of binary classifier, q_i denotes the probability of that the i^{th} title and description match. y_i is ground truth label (1 or 0) of matching. \mathcal{D} denotes the training corpus. The two tasks of MLM and TDM are trained together by multi-task learning with equal weights (0.5, 0.5).

3.3 Knowledge Acquisition Suite

In the second pre-training stage, we propose a knowledge acquisition suite to preserve multi-faceted product knowledge, consisting of a Knowledge Encoder (KE), Skeleton Attention layers, and three knowledge-acquisition tasks as shown in the right part of Fig. 4. In this stage, only parameters of knowledge acquisition suite are optimized while the Context Encoder (CE) is fixed. This ensures a smooth knowledge fusion without interfering the language preserving function of the language acquisition suite.

3.3.1 Knowledge Encoder

We continue to use the product corpus as input fed into CE and transfer the output to KE. As shown in right bottom of Fig. 4, KE consists of two projection layers and multiple Transformer layers. The projection layer aims to project input to “knowledge space” from “semantic space”, the Transformer layers store knowledge in the self-attentions and keep compatible with CE. A skip connection is applied across two projection layers to avoid losing contextual embedding information. Only contextual embeddings of the product title (i.e., $\{\mathbf{T}_i\}_{i=1}^n$) are forwarded to KE to generate *knowledge-informed* embeddings (i.e., $\{\mathbf{K}_i\}_{i=1}^n$). We discard the product description because the title already contains the most necessary information, and our problem setting is using the title to represent a product, which is more applicable when online retailers do not provide product descriptions.

3.3.2 Skeleton Attention

To address the issue of highlighting key information of products, we propose a novel attention method applied on the output of KE to generate intermediate product representations. Our attention mechanism is featured with *hierarchical structure* and *multi-faceted knowledge-guidance*:

- We use a two-layer hierarchical structure to form the attention, i.e., phrase-level attention and word-level attention. In this way, it automatically learns to attend

informative phrases in product title as well as informative words in phrase, i.e., what we consider as the “skeleton” of a product.

- We leverage multiple duplicates of the attention layer to generate intermediate representations, called Knowledge Copies (KCs). Each representation is pre-trained with a knowledge acquisition task, thus the corresponding attention weights is guided by one facet of product knowledge, and multi-faceted knowledge is stored in different duplicates of the attention.

Word-Level Attention. Given the embeddings generated by KE (i.e., $\{\mathbf{K}_i\}_{i=1}^n$, corresponding to words in product title), the first layer of Skeleton Attention is word-level attention, which learns a attention score over each word within a phrase. Specifically, we first obtain phrase boundary index by chunking product titles into phrases³. Then within each phrase, we calculate attention over each word as

$$\mathbf{u}_{ij} = \tanh(\mathbf{W}_w \mathbf{K}_{ij} + \mathbf{b}_w) \quad (3)$$

$$\alpha_{ij} = \frac{\exp(\mathbf{u}_{ij}^\top \mathbf{h}_w)}{\sum_{k'} \exp(\mathbf{u}_{i,j'}^\top \mathbf{h}_w)}, \quad \mathbf{v}_i = \sum_k \alpha_{ij} \mathbf{K}_{ij}, \quad (4)$$

where \mathbf{K}_{ij} denotes the embedding of the j^{th} word in the i^{th} phrase, we first feed it through a one-layer perceptron to get \mathbf{u}_{ij} as a hidden representation. Next we measure the importance of the word as the correlation between \mathbf{u}_{ij} and a word-level latent embedding \mathbf{h}_w , then get a normalized importance (attention) weight α_{ij} through a softmax function. \mathbf{h}_w is randomly initialized and jointly learned during the training process. Finally, we compute the phrase embedding \mathbf{v}_i by summing up all the words (i.e., $\{\mathbf{K}_{1j}, \mathbf{K}_{2j}, \dots\}$) within it based on the attention weights.

Phrase-Level Attention. After we get the intermediate phrase embeddings for phrases in product title, we obtain the local product representations in a similar way

$$\mathbf{u}_i = \tanh(\mathbf{W}_v \mathbf{v}_i + \mathbf{b}_v) \quad (5)$$

$$\beta_i = \frac{\exp(\mathbf{u}_i^\top \mathbf{h}_p)}{\sum_k \exp(\mathbf{u}_k^\top \mathbf{h}_p)}, \quad \mathbf{p} = \sum_i \beta_i \mathbf{v}_i, \quad (6)$$

where we first feed the phrase embedding \mathbf{v}_i through a one-layer MLP to get \mathbf{u}_i as a hidden representation of \mathbf{v}_i , then we measure the importance of the phrase as the correlation between \mathbf{u}_i and the phrase-level latent embedding \mathbf{h}_p , and get a normalized importance score β_i through a softmax function. Finally, we compute the product embedding \mathbf{p} as a weighted sum of the phrase embeddings (i.e., $\{\mathbf{v}_1, \mathbf{v}_2, \dots\}$) based on the attention weights.

We leverage three duplicates of the skeleton attention to generate three local representations (i.e., $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$), what we also call as “Knowledge Copies” as they are guided by three knowledge acquisition tasks to obtain corresponding knowledge.

3.3.3 Knowledge Acquisition Tasks

To preserve the multi-faceted product knowledge, we train KCs (i.e., $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$) with three knowledge acquisition tasks,

i.e., *Neighbor Prediction, Attribute Prediction, and Category Prediction*. In order to overcome the sparsity and noise issues of product knowledge, we propose a heterogeneous embedding model to represent different facets of knowledge. For each task, we use the corresponding heterogeneous knowledge embeddings as training targets of KCs. Compared to representing knowledge elements as isolated class labels, applying the knowledge embedding preserves the label correlation and helps calibrate noise and sparsity.

Heterogeneous Knowledge Embeddings. We follow three intuitions for generating the knowledge embeddings of neighbor products, attributes, categories.

Intuition1: Products that share similar attributes, categories should be close in the embedding space. This intuition helps alleviate the noise issue in product associations which are generated from user behaviors, i.e., making truly associated products close to each other.

Intuition2: Attributes, categories that cover similar sets of products should be close in the embedding space. The intuition helps mitigate the synonym and missing value issues. For example, for chocolate products, two retailers may use “Chocolate” and “Choc” as the category name respectively, but as long as two synonyms cover similar sets of products, their embeddings will be close.

Intuition3: Category embeddings should preserve the hierarchical structure information. As mentioned previously, there are rich structural correlations among categories, preserving such information improves category representations.

To fulfill the above intuitions, we propose three objective functions respectively and jointly optimize them. Let $\mathbf{n}, \mathbf{a}, \mathbf{c}$ denotes the embeddings of neighbor products, attributes, categories.

$$\mathcal{O}_1 = \sum_{(n_i, a_j) \in \mathcal{G}_P} \sum_{(n_i, c_j) \in \mathcal{G}_P} \log p(a_j | n_i) + \log p(c_j | n_i) \quad (7)$$

$$\mathcal{O}_2 = \sum_{(n_i, a_j) \in \mathcal{G}_P} \sum_{(n_i, c_j) \in \mathcal{G}_P} \log p(n_j | a_i) + \log p(n_j | c_i) \quad (8)$$

$$\log \sigma(\mathbf{n}_j^\top \mathbf{a}_i) + \sum_{z=1}^Z \mathbb{E}_{n_i \sim P_i(n)} [\log \sigma(-\mathbf{n}_i^\top \mathbf{a}_i)], \quad (9)$$

where $p(n_j | a_i) = \exp(\mathbf{n}_j^\top \mathbf{a}_i) / \sum_{n_j \in \mathcal{P}} \exp(\mathbf{n}_j^\top \mathbf{a}_i)$ denotes the probability of product n_j given attribute a_i , it follows second-order proximity [28] in network embedding. \mathcal{G}_P denotes product knowledge graph. \mathcal{P} denotes the product set. For efficient optimization, we replace $p(n_j | a_i)$ with Eq. (9), i.e., using negative sampling [13] to approximate original softmax function, where $\sigma(x) = 1/(1 + \exp(-x))$ is the sigmoid function. $p(a_j | n_i)$, $p(a_j | c_i)$, and $p(c_j | a_i)$ are calculated in the same way.

$$\mathcal{O}_3 = \sum_{i \in \mathcal{C}} \sum_{j \in \text{child}(i)} p(c_j | c_i) \quad (10)$$

$$p(\mathbf{c}_j | \mathbf{c}_i) = \frac{\exp(-d_{\text{Pointcare}}(\mathbf{c}_j, \mathbf{c}_i))}{\sum_{c \in \mathcal{C}} \exp(-d_{\text{Pointcare}}(\mathbf{c}, \mathbf{c}_i))} \quad (11)$$

$$d_{\text{Pointcare}}(\mathbf{c}_i, \mathbf{c}_j) = \text{arcosh} \left(1 + 2 \frac{\|\mathbf{c}_i - \mathbf{c}_j\|^2}{(1 - \|\mathbf{c}_i\|^2)(1 - \|\mathbf{c}_j\|^2)} \right). \quad (12)$$

3. We use AllenNLP toolkit to fulfill this, <https://demo.allennlp.org/constituency-parsing>

\mathcal{O}_3 optimizes the distances of all parent-child category pairs, $p(\mathbf{c}_j | \mathbf{c}_i)$ denotes softmax normalized distance of c_j and c_i .

$d_{\text{Pointcaré}}(\cdot, \cdot)$ denotes the distance metric used in Pointcaré embedding [15] which is the key to preserve structural correlations.

We leverage a simple yet effective multi-task learning strategy described in [23] to jointly maximize $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3$, it sample each tasks based on the size of the task data.

Knowledge Acquisition Tasks: Neighbor Prediction, Attribute Prediction, Category Prediction. With the heterogeneous knowledge embeddings generated, we use them as training targets and optimize a hinge loss of distance between KCs and their targets. For instance, the loss function of Neighbor Prediction task is defined as

$$\mathcal{L}_{NP} = - \sum_{i \in \mathcal{D}} \sum_{j \in N(i)} \max(0, 1 + \langle \mathbf{p}_{1,i}, \mathbf{n}_j \rangle - \langle \mathbf{p}_{1,i}, \mathbf{n}_j^- \rangle), \quad (13)$$

where $\mathbf{p}_{1,i}$ denotes the generated first KC (knowledge copy) of the i^{th} input product, \mathcal{D} denotes the corpus. $N(i)$ denotes the neighbor products of i , \mathbf{n}_j represents the pre-trained embedding for neighbor product j , and \mathbf{n}_j^- is random negative sample. $\langle \cdot, \cdot \rangle$ denotes the L2 distance. To be noted, only KCs and the knowledge suite are updated while knowledge embeddings (\mathbf{n}_j) are fixed. For the tasks of Attribute Prediction and Category Prediction, similarly, we calculate \mathcal{L}_{AP} and \mathcal{L}_{CP} for $\mathbf{p}_2, \mathbf{p}_3$ by replacing \mathbf{n}_j with \mathbf{a}_j and \mathbf{c}_j respectively,

$$\mathcal{L}_{AP} = - \sum_{i \in \mathcal{D}} \sum_{j \in A(i)} \max(0, 1 + \langle \mathbf{p}_{1,i}, \mathbf{a}_j \rangle - \langle \mathbf{p}_{1,i}, \mathbf{a}_j^- \rangle) \quad (14)$$

$$\mathcal{L}_{CP} = - \sum_{i \in \mathcal{D}} \sum_{j \in C(i)} \max(0, 1 + \langle \mathbf{p}_{1,i}, \mathbf{c}_j \rangle - \langle \mathbf{p}_{1,i}, \mathbf{c}_j^- \rangle). \quad (15)$$

3.3.4 Final Representation by Mixtures of Experts (MoE)

Given the knowledge-guided local representations $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$, we propose to combine them coherently to generate the final product representation. The intuitions are: (1) The same type of knowledge may have different gain effect in different instances of product (e.g., for those products that already contains attribute information like “Material 100%cotton” in title, attribute knowledge may bring limited improvements). (2) The same knowledge may contribute differently (more or less) to different downstream tasks. We utilize the Mixtures of Experts (MoE) model to fulfill the intuitions. As shown in Fig. 4, we apply a softmax gating network on the output of Knowledge Encoder ([CLS] token) to calculate three normalized scalars g_1, g_2, g_3 , which are then used as the weights summing KCs

$$g_i = \frac{\exp(\boldsymbol{\eta}_i^T \mathbf{K}_{CLS})}{\sum_j \exp(\boldsymbol{\eta}_j^T \mathbf{K}_{CLS})} \approx p(\mathbf{p}_i | \boldsymbol{\eta}, \mathbf{K}_{CLS}), \quad i = 1, 2, 3, \quad (16)$$

where $\boldsymbol{\eta}_i$ denotes the gating parameter for the i^{th} knowledge copy, \mathbf{K}_{CLS} denotes the output [CLS] token of KE, and $\boldsymbol{\eta}_i$ and \mathbf{K}_{CLS} have the same dimensions. Final product representation \mathbf{p} is calculated as a weighted sum of the gated local representations, i.e., $\mathbf{p} = \sum_{i=1}^3 g_i \mathbf{p}_i$. To be noted, in pre-training phase, only parameters behind $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ are optimized while parameters related to \mathbf{p} are fixed. That is to say,

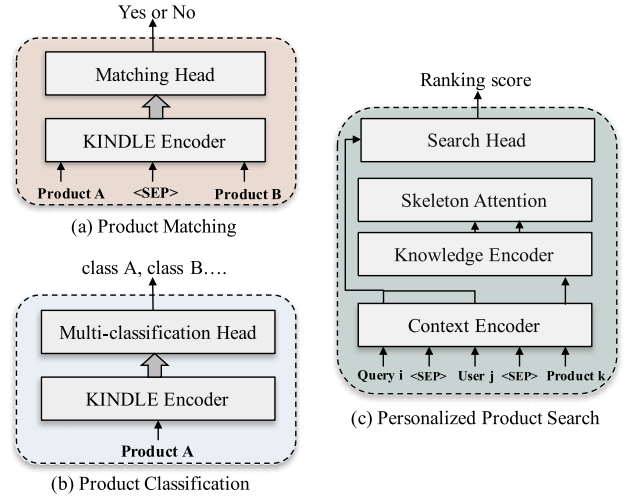


Fig. 5. Fine-tuning KINDLE on three downstream tasks.

we only calculate final representation \mathbf{p} and update other parameters during the fine-tuning phase.

4 FINE-TUNING ON DOWNSTREAM TASKS

We present how to apply KINDLE on several downstream tasks of product representation learning, i.e., product matching, personalized product search, and product classification.

4.1 Product Matching

As shown in Fig. 5a, given two products (e.g., *Kingston 133x high-speed 4GB compact flash card ts4gcf133, 21.5 MB per sec data transfer rate, vs, Kingston ts4gcf133 4GB compact flash memory card (133x)*), this task aims to predict whether they refer to the same product [19]. It is considered as the *entity matching* task in product domain. For each product pair p_i, p_j , we first feed them into Context Encoder of KINDLE, then we feed their contextual embeddings to the knowledge suite to get their knowledge-aware representations $\mathbf{p}_i, \mathbf{p}_j$ respectively. Finally a simple binary classifier is applied on the concatenation of them (i.e., $[\mathbf{p}_i, \mathbf{p}_j]$) to perform prediction.

4.2 Product Classification

As shown in Fig. 5b, given an input product title p , this task aims to retrieve all the categories in a predefined hierarchy \mathcal{T} that p belongs to. We use the final representation of KINDLE along with multiple flat binary classifiers to perform prediction.

4.3 Personalized Product Search

As shown in Fig. 5c, given input query q_i , user profile u_j , this task aims to return the products that the user is most likely interested in, ranked by scores. Similarly, given a training triple (q_i, u_j, p_k) , we first obtain the representation of them via KINDLE respectively (i.e., $\mathbf{q}_i, \mathbf{u}_j, \mathbf{p}_k$). To be noted, only product representation (\mathbf{p}_k) is obtained via the entire pipeline (Context Encoder + knowledge suite) while $\mathbf{q}_i, \mathbf{u}_j$ are averaged contextual embeddings. Then we minimize the personalized search loss defined in [2] for fine-tuning (equivalent to maximizing the likelihood of observed user-query-item triples)

TABLE 2
Statistics of the Datasets

Datasets	Pre-training	P-M	P-S	P-C
# Products	1M	7,851	61,324	21,039
# Attributes	52,582	—	1,332	1,594
# Categories	5,891	—	267	133
# Attributes/item	7.2	—	—	—
# Categories/item	6.5	—	—	—
# Neighbors/item	27.8	—	—	—
# (q, u, p) triples	—	—	562,345	—
# (p_i, p_j) pairs	—	82,420	—	—

$$\begin{aligned}
\mathcal{L}_{ps} &= - \sum_{(i,j,k) \in \mathcal{D}_{ps}} \log P(\mathbf{p}_k | \mathbf{u}_j, \mathbf{q}_i) \\
&= - \sum_{(i,j,k) \in \mathcal{D}_{ps}} \log \sigma(\mathbf{p}_k \cdot (\gamma \mathbf{u}_j + (1 - \gamma) \mathbf{q}_i)) \\
&\quad + z \cdot \mathbb{E}_{\mathbf{p}'_k \sim P_k} [\log \sigma(-\mathbf{p}'_k \cdot (\gamma \mathbf{u}_j + (1 - \gamma) \mathbf{q}_i))], \quad (17)
\end{aligned}$$

where z is the number of negative samples and P_k is the uniform noise distribution for negative sampling. γ is a hyperparameter controlling the weight of user-side and query-side information.

5 EXPERIMENTS

We conduct a series of experiments to validate the effectiveness of KINDLE on three downstream tasks (i.e., *product matching*, *personalized product search*, *product classification*) as well as their zero-shot versions for new products. Details of each task and the corresponding KINDLE based fine-tuning architecture are presented in the Section 4. In addition, we propose a new task, *zero-shot knowledge recovery* to verify the knowledge acquisition ability of KINDLE, and real examples are presented for better understanding.

5.1 Experimental Settings

5.1.1 Tasks and Datasets

We create the pre-training dataset leveraging resources publicly available. Then we use existing benchmark datasets or create new datasets for downstream tasks. Table 2 presents the statistics of all the datasets.

Pre-Training Data. The pre-training dataset consists of a product corpus (1M product titles and descriptions) and a product knowledge graph (1M products, 52,582 attributes and 5,891 categories). Each product has 7.2 attributes, 6.5 categories and 27.8 neighbors in average, revealing the noisy and sparse nature of the heterogeneous product knowledge. Fig. 6 shows the length distributions of product title and description in our dataset, which are similar across different categories.

Downstream Task Data. (i) *Product Matching (P-M)* aims to predict whether two given products refer to the same product, we use the benchmark dataset described in [19], which extracts more than seven thousands products from various online retailer sources. It consists of 82,420 training pairs (36,908 positive + 45,512 negative), and 2,200 testing pairs (600 positive + 1,600 negative). We use 10% of the training

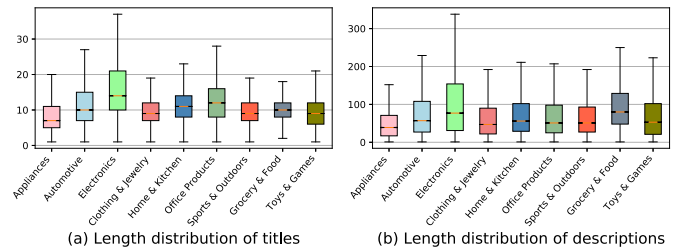


Fig. 6. Length distributions of product titles and descriptions under different categories.

data as validation set. (ii) *Personalized Product Search (P-S)* aims to predict a ranking score for a product given a query and user profile, we follow the same procedure in work [2] to create a dataset using Amazon *Electronics* subset, consisting of 61,324 products and 562,345 (user, query, product) triples. We divide them into train/validation/test set with the ratio of 7:1:2. (iii) *Product Classification (P-C)* aims to return all the categories that a given product belongs to, we create an evaluation dataset by extracting Amazon product metadata, consisting of 21,039 product titles and 133 fine-grained categories. We divide them into train/validation/test set with the ratio of 7:1:2.

5.1.2 Evaluation Metrics and Baselines

Product Matching. We use Precision, Recall and F-1 to evaluate the matching results. We compare our pre-trained model with a state-of-the-art entity matching baseline *Magellan* [11], and four strong deep learning solutions described in *Deep-Matcher*, i.e., *SIF*, *RNNs*, *Attention*, *Hybrid* [14].

Personalized Product Search. We adopt several ranking metrics following prior work for evaluation of this task, i.e., mean reciprocal rank (MRR), normalized discounted cumulative gain (NDCG@10), and the proportion of correct result ranked in the Top 10 (Hit@10). We compare with state-of-the-art baseline as well as strong baselines in prior work of this field. The full names of compared baselines are: Transformer-based Embedding Model (*TEM* [4]), Zero Attention Model (*ZAM* [1]), Attention-based Embedding Model (*AEM* [1]), Hierarchical Embedding Model (*HEM* [2]), Query Embedding Mode (*QEM*).

Product Classification. We adopt multi-class classification metrics, i.e., Accuracy, Micro-F1, and Macro-F1 to evaluate this task. We compare with strong DL baselines for title classification including Long Short Term Memory networks (*LSTM*), *Attentive LSTM*, Hierarchical Attention Network (*HAN*).

For all the three tasks, we also compare with *vanilla BERT* and *BERT-K*. The former is post-trained on our corpus without considering any product knowledge, the latter has second knowledge pre-training stage supervised by multi-task knowledge classification (without considering the training discrepancy, knowledge noise, sparsity issues). We average the output of BERT to generate product representation and use the same task-specific architectures as KINDLE for fine-tuning.

Internal Baselines. Besides task-specific baselines, we also compare with several internal baselines of KINDLE for ablation study. They are pre-trained following the same two-stage

TABLE 3
Performance on Product Matching

		Matching Metrics (%)		
Model		Precision	Recall	F1
Baselines	Magellan	61.51	76.77	68.33
	DeepMatcher (SIF)	61.73	77.54	69.23
	DeepMatcher (RNNs)	62.98	78.96	70.12
	DeepMatcher (Attention)	63.22	79.35	70.37
	DeepMatcher (Hybrid)	63.73	79.76	70.84
	BERT	65.51	81.23	72.52
	BERT-K	65.78	81.42	72.35
KINDLE	KE	67.20	81.23	73.73
	SA	68.51	82.53	74.86
	MoE	69.21	83.34	75.62

The best performance is highlighted in boldface.

TABLE 4
Performance on Personalized Product Search

		Ranking Metrics (%)		
Model		MRR	NDCG@10	Hit@10
Baselines	QEM	24.52	25.31	28.67
	HEM	27.86	27.91	30.14
	AEM	28.04	28.23	30.55
	ZAM	29.31	30.51	31.22
	TEM	31.53	32.95	31.98
	BERT	32.85	33.94	33.02
	BERT-K	33.12	33.99	32.88
KINDLE	KE	37.57	38.04	37.98
	SA	39.57	39.15	39.37
	MoE	41.11	40.02	42.49

framework, but employ different ways to generate the final product representation in fine-tuning stage: (1) p_1, p_2, p_3 , three local representations are used to represent products directly, they incorporate neighbor, attribute, category knowledge respectively. (2) *KE*, averaging the output of Knowledge Encoder while Skeleton Attention (SA) and Mixture of Experts are not used. (3) *SA*, averaging SA’s output (i.e., p_1, p_2, p_3) by weights learned from a softmax function. To be noted, our full version of KINDLE is achieved by averaging p_1, p_2, p_3 with Mixture of Experts (*MoE*).

5.1.3 Hyper-Parameters

Pre-Training Parameters. For each internal baseline of KINDLE, we start pre-training with the first stage, i.e., language acquisition tasks. Then we fix the language model and start the second stage pre-training, i.e., optimizing the knowledge suite supervised by knowledge acquisition tasks and pre-trained knowledge embeddings. We use BERT_{base, uncased} (12 layers, 768 hidden dimensions, 12 heads) to initialize Context Encoder. Token masking ratio for MLM is 15%, title description corruption ratio for TDM is 50%. The transformer in Knowledge Encoder is set to be lighter, with 4 layers, 768 hidden dimensions, 6 heads, and the projection layer is equivalent to 2-layer perceptron with the same input and output dimensions as KE. For both stages, we set the maximum

TABLE 5
Performance on Product Classification

		Classification Metrics (%)		
Model		Accuracy	Mi-F1	Ma-F1
Baselines	LSTM	75.84	84.90	80.87
	Attentive LSTM	76.34	85.22	82.47
	HAN	77.37	86.90	82.47
	BERT	78.24	86.98	82.65
	BERT-K	78.55	86.56	82.28
KINDLE	KE	81.15	89.76	84.65
	SA	82.23	90.14	84.97
	MoE	83.38	91.45	86.16

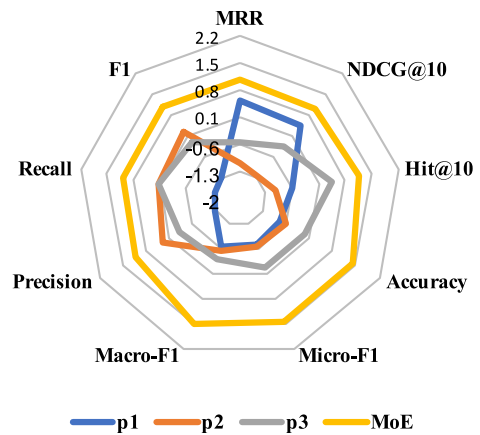


Fig. 7. The effect of incorporating individual facet of product knowledge and combining all facets. We show their performances on three tasks (nine metrics) together in a radar map. For a clear comparison, all the values are first standardized.

epochs as 10, batch size 32, learning rate $1e-5$, also, use early stopping to avoid overfitting.

Fine-Tuning Parameters. For all the external baselines, we either use the best parameter configuration described in previous papers (P-M and P-S tasks) or obtain the best configuration using our validation set (P-C task). For fine-tuning KINDLE on each task, please refer to the Section 4 for implementation details. We choose the batch size, learning rate and epochs from {16, 32, 64}, { $5e-6, 1e-5, 2e-5, 5e-5$ } and {2, 4, 6, 8, 10}. We pick the best parameters on the validation set and report the corresponding test results. We found the setting that works best across most tasks and models is 32 batch size, 4 or 6 epochs and a learning rate of $2e-5$. All results are reported as averages of 10 runs.

5.2 Experimental Results

5.2.1 Overall Performance and Ablation Study

Tables 3, 4, and 5 show the results of product matching, personalized product search and product classification respectively. And Fig. 7 shows the effects after incorporating different product knowledge and fusing them all in KINDLE. The key observations are: (1) *Transformer based models* (i.e., TEM, BERT) achieved the best performances among external baselines, indicating the effectiveness of contextualized semantics. (2) Compared with BERT, our internal baselines with a *second-stage knowledge acquisition pre-training*, achieved

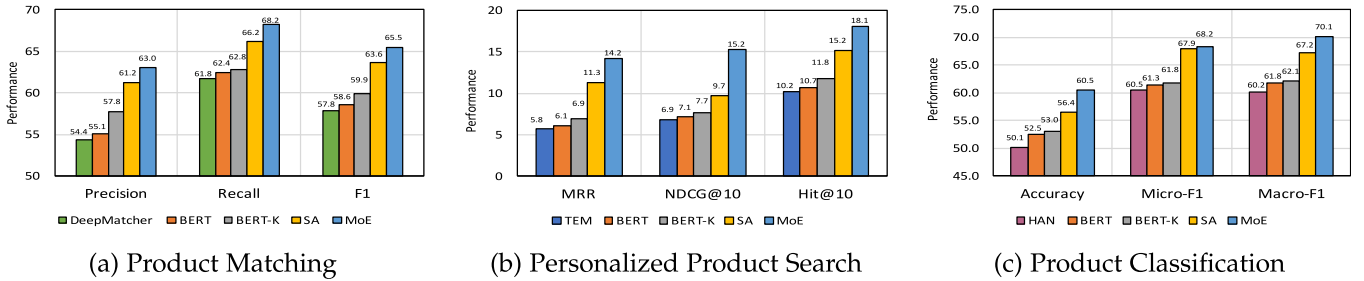


Fig. 8. Performance on the zero-shot settings of downstream tasks.

further improvement (5% accuracy on product matching, 24% MRR on product search and 6% accuracy on product classification over the second best), which validated the effectiveness of product knowledge for PRL. However, BERT-K did not get much improvements even after incorporating knowledge by multi-task learning, indicated addressing the learning discrepancy issue is critical to successful knowledge acquisition. (3) Not all product knowledge contributed equally to the performance. For product search, p_1 (Neighbor knowledge) achieved the most improvements over baselines, p_3 (Category) followed, p_2 (Attribute) got very limited improvements. In product matching, p_2 (incorporated attribute knowledge) obtained clear improvements over BERT. In product classification, however, p_3 (incorporated category knowledge) obtained obvious improvements. (4) Compared with combining both Knowledge Encoder and Skeleton Attention, solely using Knowledge Encoder did not get notable improvements, indicating that by capturing key information, our attention mechanism actually helps encoding product knowledge in a more effective way. We conjecture this is due to there exist certain correlations between key information in product title (e.g., *Baby Monitor with Remote Pan-Tilt-Zoom Camera and 3.2" LCD Screen*) and product knowledge (e.g., category: *video monitor*). (5) Furthermore, for all tasks, combining different product knowledge by MoE achieved the best result in internal baselines, indicating that the dynamic weights generated by input-aware gating enables more robust fusion of heterogeneous multi-faceted knowledge.

5.2.2 Transferability on New Products

To validate the transferability on unseen products, we proposed the *zero-shot versions* of the three downstream tasks. Specifically, for each task, we create 5,000 zero-shot testing samples (denoted by \mathcal{D}_Z), in which not only all the samples but also the related products are never appeared in the pre-training data (\mathcal{D}_P) and task-specific training data (\mathcal{D}_F). In addition, 30% of them are OOD samples from other domains. Then we directly measure the performance of all models (pre-trained on \mathcal{D}_P , fine-tuned on \mathcal{D}_F) on \mathcal{D}_Z without further fine-tuning. Fig. 8 shows the performance comparison between the best baselines and our model. We observed that BERT outperformed non-BERT models, and KINDLE-MoE obtained the best results in all tasks. We conclude that (1) Contextualized transformer-based models have better transferability than other deep models; (2) Incorporating product knowledge can further improve the transferability of product representation; (3) Compared with averaging SA, Mixture-of-Experts is a better way to deal

with the heterogeneity of product knowledge, this is consistent with what we observed in the regular setting of downstream tasks.

5.2.3 Zero-Shot Knowledge Recovery

We propose this new task to validate the knowledge acquisition ability of KINDLE. Formally, given a product title, we aim to retrieve its most likely neighbor products, attributes, categories from the pre-defined knowledge sets. We compare pre-trained KINDLE with a simpler variant KINDLE-C (C indicates classification). Both models are only pre-trained in an unsupervised fashion, not fine-tuned in any tasks. While KINDLE uses pre-trained heterogeneous knowledge embeddings as targets to guide the knowledge acquisition tasks, and KINDLE-C treats each knowledge element as a classification label and use a classification loss to optimize p_1, p_2, p_3 , i.e., it does not consider the relatedness between different knowledge elements. Specifically, we sample 10,000 products that have never appeared in our pre-training data and feed their titles into both models to obtain three local representations p_1, p_2, p_3 respectively. Then we use them to retrieve three types of knowledge respectively. For KINDLE, we calculate the L2 distance between p_i and all knowledge elements as the retrieve score while for KINDLE-C we use the classifier scores over all elements. Based on the scores, we retrieve and rank all knowledge elements to report the Mean Rank (the lower represents better) and Hit@10 of ground truth knowledge in Fig. 9. KINDLE outperformed KINDLE-C consistently, indicating our heterogeneous knowledge embeddings successfully captured the relatedness between knowledge elements to alleviate the noise and sparsity issues. We show two real examples of knowledge recovery in Table 6. KINDLE got more robust results than KINDLE-C, i.e., less wrong answers, usually at least correlated or neutral answers.

5.3 System Deployment

We deploy KINDLE on a *Business Matching* platform of NEC, which aims to find business partners for companies according to their business need and product information.

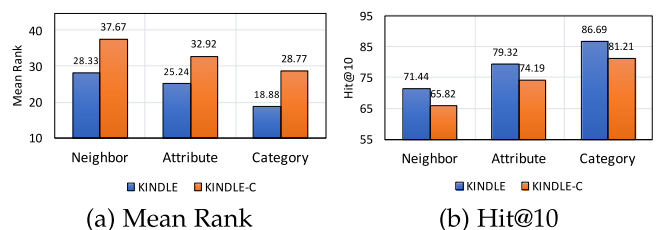




Fig. 9. Performance on zero-shot knowledge recovery.

TABLE 6
Two Knowledge Recovery Examples

Products	Method	Recovered knowledge elements
 Fujifilm X-E3 Mirrorless Digital Camera XF23mmF2 R WR Kit - Silver	KINDLE	Attributes: (1) *Brand Fujifilm ; (2) *Optical Sensor Size APS-C ; (3) Material Waterproof . Categories: (1) *Electronics ; (2) *Camera&Photo ; (3) Lenses ;
	KINDLE-C	Attributes: (1) Light Source Type LED ; (2) Material Stainless steel ; (3) *Brand Fujifilm . Categories: (1) *Electronics ; (2) Accessories ; (3) Simulated Cameras .
 Sboly Single Serve Coffee Maker Brewer for K-Cup Pod & Ground Coffee Thermal Drip Instant Coffee Machine with Self Cleaning Function	KINDLE	Attributes: (1) Material BPA Free Plastic ; (2) *Capacity 1 Cup ; (3) *Brand Sboly . Categories: (1) *Home & Kitchen ; (2) *Single-Serve Brewers ; (3) Electronics .
	KINDLE-C	Attributes: (1) *Color Black ; (2) Flavor Dark Roast ; (3) *Brand Sboly . Categories: (1) *Home & Kitchen ; (2) Coffee ; (3) Electronics .

We return the Top 3 retrieved elements for each type of knowledge. Bold, red, blue, and orange indicate correct, wrong, highly correlated, and neutral answers (annotated by three volunteers).

We apply our model to learn *contextualized* representations of needs and *knowledge-aware* representations of products respectively. Since the number of needs and products of a company may vary from 1 to more than 100, during training time, we run 100 duplicates of our model in parallel on 10 GTX-1080-Ti GPUs to generate representations efficiently. Given a query company, to avoid large latency on returning top-k results, we employ Maximum Inner Product Search (MIPS) algorithms using running time and storage space that scale sub-linearly with the number of candidates [21], [25]. Both real-world historical ground-truth and manually annotated matching data shows that compared with existing deep matching baselines, KINDLE achieves 9.8% and 15.2% improvements on Precision in normal setting and zero-shot setting, respectively.

6 RELATED WORK

Our work is related to *unsupervised representation learning* and *e-commerce computing*. Ever since the advent of word2vec [13], a varieties of unsupervised representation learning methods have been proposed for a broad range of domains and applications [8], [12], [16], [28], [37]. In e-commerce, item2vec [3] is the first work that generalized skip-gram based representation learning to product recommendation. Following it, more advanced embedding model have been proposed for recommendation [30], [41], [42]. For personalized product search, [2] proposed a series of models that jointly optimizes product, query and user representations [1], [2]. Other following work [38], [39] jointly learns embeddings for product search and recommendation to achieve improvements for both tasks. A major limitation of these prior work is that they do not consider contextualized semantics. Although recent work [4], [9] leverage transformer and BERT for search and ranking, none of them utilize rich product knowledge. Product knowledge has shown effectiveness [35] in product-related tasks, whereas none existing work considers both knowledge and contextual embeddings. Our work is the first unified work to incorporate contextual word embeddings and product knowledge for product representation learning.

Our work is also related to *Pre-trained Language Models (PLMs)*. Recently, the emergence of PLMs [6], [17], [20] has dominated the progress of natural language processing. Compared with traditional word embedding [13], PLMs learn to represent words based on the entire input context

to deal with word polysemy, thus captures semantics more accurately. Besides, it inherits strong transfer learning ability from the paradigm of pre-training and fine-tuning. Following PLMs, many endeavors (e.g., SpanBERT [10], ERNIE [27], etc.) have been made for further optimization. On the other hand, to enable PLMs with world knowledge, several attempts [18], [26], [32], [33], [40] have been made to inject knowledge from external KGs into BERT. Most of these work is based on the “BERT+entity linking” paradigm, however, it is not suitable for product representation learning as retailer-based product knowledge are usually heterogeneous and noisy. Our knowledge pre-training suite is designed to be flexible and robust to overcome these issues.

7 CONCLUSION

In this work, we proposed a novel knowledge-driven pre-training framework for learning product representations. It integrated multi-faceted product knowledge into language model smoothly by extending PLM with a knowledge acquisition stage. Multi-objective heterogeneous knowledge embeddings were also utilized in knowledge acquisition tasks to calibrate knowledge noise and sparsity issues. Moreover, an input-aware gating network was utilized to re-weight different knowledge facets for different tasks. Experiments on three downstream tasks demonstrated the effectiveness of our framework.

REFERENCES

- [1] Q. Ai, D. N. Hill, S. V. N. Vishwanathan, and W. B. Croft, “A zero attention model for personalized product search,” in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, 2019, pp. 379–388.
- [2] Q. Ai, Y. Zhang, K. Bi, X. Chen, and W. B. Croft, “Learning a hierarchical embedding model for personalized product search,” in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2017, pp. 645–654.
- [3] O. Barkan and N. Koenigstein, “Item2Vec: Neural item embedding for collaborative filtering,” in *Proc. IEEE 26th Int. Workshop Mach. Learn. Signal Process.*, 2016, pp. 1–6.
- [4] K. Bi, Q. Ai, and W. B. Croft, “A transformer-based embedding model for personalized product search,” 2020, *arXiv:2005.08936*.
- [5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” 2018, *arXiv:1810.04805*.
- [6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics*, 2019, pp. 4171–4186.

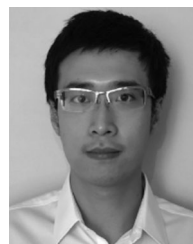
- [7] X. L. Dong et al., "AutoKnow: Self-driving knowledge collection for products of thousands of types," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 2724–2734.
- [8] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 855–864.
- [9] S. Han, X. Wang, M. Bendersky, and M. Najork, "Learning-to-rank with BERT in TF-ranking," 2020, *arXiv:2004.08476*.
- [10] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy, "SpanBERT: Improving pre-training by representing and predicting spans," *Trans. Assoc. Comput. Linguistics*, vol. 8, pp. 64–77, 2020.
- [11] P. Konda et al., "Magellan: Toward building entity matching management systems," *Proc. VLDB Endowment*, vol. 9, no. 12, pp. 1197–1208, 2016.
- [12] M. Li, D. Zhang, Y. Jia, Y. Wang, and X. Cheng, "Link prediction in knowledge graphs: A hierarchy-constrained approach," *IEEE Trans. Big Data*, vol. 8, no. 3, pp. 630–643, Jun. 2022.
- [13] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. 26th Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.
- [14] S. Mudgal et al., "Deep learning for entity matching: A design space exploration," in *Proc. Int. Conf. Manage. Data*, 2018, pp. 19–34.
- [15] M. Nickel and D. Kiela, "Poincaré embeddings for learning hierarchical representations," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6338–6347.
- [16] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2014, pp. 701–710.
- [17] M. Peters et al., "Deep contextualized word representations," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics*, 2018, pp. 2227–2237.
- [18] M. E. Peters et al., "Knowledge enhanced contextual word representations," 2019, *arXiv:1909.04164*.
- [19] A. Pimpeli, R. Peeters, and C. Bizer, "The WDC training dataset and gold standard for large-scale product matching," in *Proc. Companion Proc. World Wide Web Conf.*, 2019, pp. 381–386.
- [20] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," *OpenAI*, 2018.
- [21] P. Ram and A. G. Gray, "Maximum inner-product search using cone trees," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2012, pp. 931–939.
- [22] P. Ristoski, P. Petrovski, P. Mika, and H. Paulheim, "A machine learning approach for product matching and categorization," *Semantic Web*, vol. 9, no. 5, pp. 707–728, 2018.
- [23] V. Sanh, T. Wolf, and S. Ruder, "A hierarchical multi-task approach for learning embeddings from semantic tasks," in *Proc. 33rd AAAI Conf. Artif. Intell.*, 2019, pp. 6949–6956.
- [24] K. Shah, S. Kopru, and J. D. Ruvini, "Neural network based extreme classification and similarity models for product matching," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics*, 2018, pp. 8–15.
- [25] A. Shrivastava and P. Li, "Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS)," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 2321–2329.
- [26] Y. Sun et al., "ERNIE 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation," 2021, *arXiv:2107.02137*.
- [27] Y. Sun et al., "Enhanced representation through knowledge integration," 2019, *arXiv:1904.09223*.
- [28] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 1067–1077.
- [29] C. Van Gysel, M. de Rijke, and E. Kanoulas, "Mix'n match: Integrating text matching and product substitutability within product search," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, 2018, pp. 1373–1382.
- [30] F. Vasile, E. Smirnova, and A. Conneau, "Meta-Prod2Vec: Product embeddings using side-information for recommendation," in *Proc. 10th ACM Conf. Recommender Syst.*, 2016, pp. 225–232.
- [31] A. Vaswani et al., "Attention is all you need," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
- [32] R. Wang et al., "K-adapter: Infusing knowledge into pre-trained models with adapters," 2020, *arXiv:2002.01808*.
- [33] X. Wang et al., "KEPLER: A unified model for knowledge embedding and pre-trained language representation," *Trans. Assoc. Comput. Linguistics*, vol. 9, pp. 176–194, 2021.
- [34] Y. Wu et al., "Google's neural machine translation system: Bridging the gap between human and machine translation," 2016, *arXiv:1609.08144*.
- [35] D. Xu, C. Ruan, J. Cho, E. Korpeoglu, S. Kumar, and K. Achan, "Knowledge-aware complementary product representation learning," in *Proc. 13th Int. Conf. Web Search Data Mining*, 2020, pp. 681–689.
- [36] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan, "Product knowledge graph embedding for e-commerce," in *Proc. 13th Int. Conf. Web Search Data Mining*, 2020, pp. 672–680.
- [37] Z. Yuan, H. Liu, R. Hu, D. Zhang, and H. Xiong, "Self-supervised prototype representation learning for event-based corporate profiling," in *Proc. 35th AAAI Conf. Artif. Intell.*, 2021, pp. 4644–4652.
- [38] H. Zamani and W. B. Croft, "Joint modeling and optimization of search and recommendation," 2018, *arXiv:1807.05631*.
- [39] H. Zamani and W. B. Croft, "Learning a joint search and recommendation model from user-item interactions," in *Proc. 13th Int. Conf. Web Search Data Mining*, 2020, pp. 717–725.
- [40] D. Zhang et al., "Domain-oriented language modeling with adaptive hybrid masking and optimal transport alignment," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2021, pp. 2145–2153.
- [41] Y. Zhang, Q. Ai, X. Chen, and W. B. Croft, "Joint representation learning for top-n recommendation with heterogeneous information sources," in *Proc. ACM Conf. Inf. Knowl. Manage.*, 2017, pp. 1449–1458.
- [42] K. Zhao, Y. Li, Z. Shuai, and C. Yang, "Learning and transferring ids representation in e-commerce," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 1031–1039.



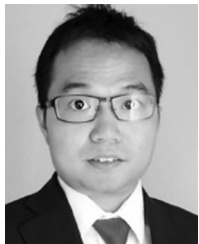
Denghui Zhang received the BE degree from the University of Science and Technology, Beijing, China, in 2015, the MS degree from the Chinese Academy of Sciences, China, 2018. He is currently working toward the PhD degree with the Information System Department, Rutgers University. His research interests include data mining, business analytics, natural language processing, and representation learning.



Yanchi Liu received the PhD degree in information technology from Rutgers, the State University of New Jersey. He is currently a research staff member in NEC Labs America. His research interests include data mining, artificial intelligence, and natural language processing. He has published prolifically in top conferences/journals such as KDD, IJCAI, WWW, *IEEE Transactions on Cybernetics*. He also served as senior program committee/program committee members among top conferences and journals in the data mining and artificial intelligence communities.

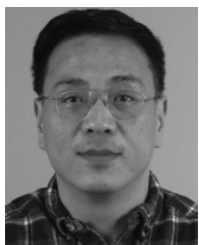


Zixuan Yuan received the BS degree from Tongji University, 2015, and the MS degree from Rutgers University. He is currently working toward the PhD degree with the Management Science and Information System Department, Rutgers University, supervised by Prof. Hui Xiong. His research focuses on data-driven financial applications, social network modeling, and representation learning.



Yanjie Fu received the BE degree from the University of Science and Technology of China, in 2008, and the ME degree from Chinese Academy of Sciences, in 2011, and the PhD degree from Rutgers, the State University of New Jersey, in 2016. He is an assistant professor with the Department of Computer Science, University of Central Florida. His research interests include data mining and Big Data analytics. He has research experience in industry research labs, such as Microsoft Research Asia and IBM Thomas J. Watson Research Center.

He has published prolifically in refereed journals and conference proceedings, such as *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Mobile Computing*, *ACM Transactions on Knowledge Discovery from Data*, *ACM SIGKDD*, *AAAI*, *IJCAI*, *VLDB*, *WWW*.



Haifeng Chen received the BEng and MEng degrees in automation from Southeast University, China, in 1994 and 1997 respectively, and the PhD degree in computer engineering from Rutgers University, New Jersey, in 2004. He is currently the head of Data Science and System Security Department, NEC Laboratory America, Princeton, New Jersey. His research focus is on Big Data analytics, AI, software and system security, smart service, and platforms. He has co-authored numerous publications in top conference and journals, and has more than 70 patents granted or applied.



Hui Xiong (Fellow, IEEE) received the BE degree in automation from the University of Science and Technology of China (USTC), Hefei, China, the MS degree in computer science from the National University of Singapore (NUS), Singapore, and the PhD degree in computer science from the University of Minnesota - Twin Cities, in 2005. He is currently a distinguished professor with Rutgers, the State University of New Jersey, where he received the 2018 Ram Charan Management Practice Award as the Grand Prix winner from the

Harvard Business Review, RBS Dean's Research Professorship (2016), two-year early promotion/tenure (2009), the Rutgers University Board of Trustees Research Fellowship for Scholarly Excellence (2009), the ICDM Best Research Paper Award (2011), Dean's Award for Meritorious Research (2010, 2011, 2013, 2015) with Rutgers Business School, the 2017 IEEE ICDM Outstanding Service Award (2017), and the AAAI-2021 Best Paper Award (2021). For his outstanding contributions to data mining and mobile computing, he was elected an ACM distinguished scientist, in 2014, and an AAAS fellow, in 2020.

▷ **For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.**