Multi-Label Zero-Shot Product Attribute-Value Extraction

Jiaying Gong
Virginia Tech
Blacksburg, U.S.
gjiaying@vt.edu

Hoda Eldardiry
Virginia Tech
Blacksburg, U.S.
hdardiry@vt.edu

ABSTRACT

E-commerce platforms should provide detailed product descriptions (attribute values) for effective product search and recommendation. However, attribute value information is typically not available for new products. To predict unseen attribute values, large quantities of labeled training data are needed to train a traditional supervised learning model. Typically, it is difficult, time-consuming, and costly to manually label large quantities of new product profiles. In this paper, we propose a novel method to efficiently and effectively extract unseen attribute values from new products in the absence of labeled data (zero-shot setting). We propose HyperPAVE, a multi-label zero-shot attribute value extraction model that leverages inductive inference in heterogeneous hypergraphs. In particular, our proposed technique constructs heterogeneous hypergraphs to capture complex higher-order relations (i.e. user behavior information) to learn more accurate feature representations for graph nodes. Furthermore, our proposed HyperPAVE model uses an inductive link prediction mechanism to infer future connections between unseen nodes. This enables HyperPAVE to identify new attribute values without the need for labeled training data. We conduct extensive experiments with ablation studies on different categories of the MAVE dataset. The results demonstrate that our proposed HyperPAVE model significantly outperforms existing classification-based, generation-based large language models for attribute value extraction in the zero-shot setting.

CCS CONCEPTS

• **Computing methodologies → Information extraction**.

KEYWORDS

attribute value extraction; zero-shot learning; heterogeneous hypergraph; inductive link prediction

ACM Reference Format:
Jiaying Gong and Hoda Eldardiry. 2024. Multi-Label Zero-Shot Product Attribute-Value Extraction. In *Proceedings of the ACM Web Conference 2024 (WWW '24), May 13–17, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3589334.3645649

WWW '24, May 13–17, 2024, Singapore, Singapore

ACM ISBN 979-8-4007-0171-9/24/05.
https://doi.org/10.1145/3589334.3645649

1 INTRODUCTION

Product attribute value extraction (AVE) aims to extract attribute-value pairs (i.e. <color: red>) from e-Commerce product descriptions, which provides a better search and recommendation experience for customers. Existing studies on AVE mainly focus on supervised-learning models such as sequence labeling [29, 73], extractive question answering [57, 61] and multi-modal learning [20, 41, 60, 62] models. These supervised learning models are trained to only predict seen attribute value pairs. However, new products with unseen attribute-value pairs enter the market every day in real-world e-commerce platforms. It is time-consuming and costly to manually label large quantities of new products for training.

Some recent works focus on open mining models [70, 77] to directly extract attribute values from product titles or descriptions. However, these approaches can not discover attribute values that are not explicitly mentioned in the text. In other words, these open mining models can not extract values that never appear in the product profile. To extract unseen attribute values, these open mining models use self-supervised learning, but they still need a high-quality seed attribute set bootstrapped from existing resources. Besides these open mining models, some generative large language models (LLM) are fine-tuned to autoregressively decode unseen attribute values from the input text. However, fine-tuning such LLM (i.e. T5 [50]) requires a lot of time and computing resources.

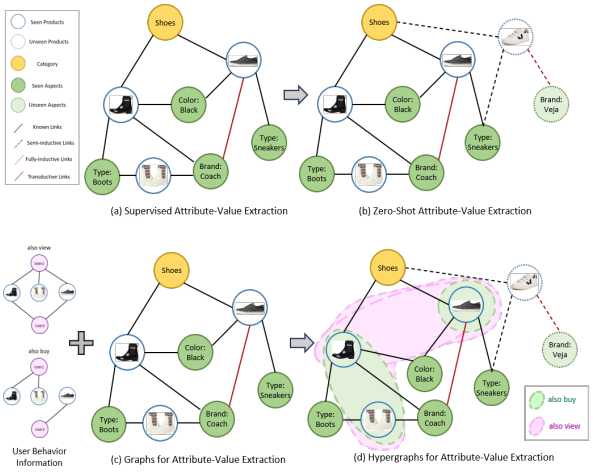

Figure 1: An example of zero-shot product attribute-value extraction by semi-inductive link predictions.

To address the above challenges, we propose HyperPAVE, a multi-label zero-shot attribute value extraction model that leverages inductive inference in heterogeneous hypergraphs to recognize unseen (new) attribute-value pairs (aspects) for which there is no

available labeled training data. Motivated by the inductive graph learning, which shows the superiority of GNN to inductively adapt to infer unseen nodes [15, 68], we build inductive heterogeneous hypergraphs employing inductive link prediction mechanisms to infer missing or future connections (e.g., from new 'product' node to unseen 'aspect' node). The top part of Figure 1 shows an example comparison between supervised (Figure 1a) and zero-shot (Figure 1b) attribute value extraction. Existing works formulate relation propagation as a transductive link prediction task (Figure 1a), where links can only be predicted between seen nodes (products and aspects) [3, 43]. To recognize unseen (new) aspects for new products, negative links are added in the original graph and the model is trained to predict whether an edge exists between two nodes based on the node features. HyperPAVE aims to learn the connections between both the nodes' features that are obtained from the fine-tuned LLM-based encoder and the complex graph structure. Motivated by the success of combining inductive GNNs and pre-trained BERT models [28], HyperPAVE is designed to enhance the inductive hypergraph-based model with fine-tuned BERT contextual embeddings for each node. Then, HyperPAVE is updated with zero-shot products and aspects with fine-tuned contextual embeddings, where message-passing is conducted directly on the updated graph, ensuring the inductive inference ability.

In addition, given the complexity of product data, it is important to design a model that can capture the heterogeneous, interconnected, and higher-order representation of both product data and user behavior data. Therefore, our proposed model HyperPAVE consists of various types of nodes including 'category', 'product', and 'aspect'. The product node records information including both product titles and descriptions. To fully express the semantic information for attribute-value pairs, the aspect nodes record detailed attribute-value descriptions generated by a generator. The proposed hypergraph representation uses higher-order relations to capture complex and interconnected user behavior information (e.g., 'also buy', 'also view') and product inventory information (e.g., 'product has aspects', 'category includes products'). The bottom part of Figure 1 shows an example comparison between graph-based (Figure 1c) and hypergraph-based (Figure 1d) attribute value extraction. To capture complex interconnected user behavior information, instead of using multiple graphs (one for each behavior e.g., "also buy" and "also view"), we construct hypergraphs using hyperedges to represent user behavior information as higher-order relations. Compared to using several different graphs to capture complex relations, using a hypergraph (1) can include more (i.e. user behavior) information for the final node representation, (2) does not need to include user nodes in the graph, and (3) relations are not limited to binary connections. The contributions are summarized as:

- We propose a multi-label zero-shot model HyperPAVE to extract unseen attribute values for new products without labeled training data. HyperPAVE leverages an inductive link prediction mechanism combined with a fine-tuned BERT encoder to obtain unseen contextual node features.
- We build heterogeneous hypergraphs with higher-order relations to capture the complex and interconnected user behavior and structured product inventory information.
- Extensive experiments on the public dataset MAVE demonstrate that HyperPAVE significantly outperforms the classification model, generative LLMs, and graph-based models in zero-shot learning. Besides, HyperPAVE also shows the effectiveness and efficiency of training.

2 RELATED WORKS

2.1 Attribute Value Extraction

Attribute value extraction (AVE) aims at extracting attribute-value pairs (aspects) based on the product information. Early works use rule-based methods with domain-specific dictionaries to match target attribute value pairs [19, 48, 66]. With the development of neural networks, some studies view AVE as a sequence labeling problem [29, 52, 73, 79]. Then, question-answering-based models are built to treat attributes as questions and values as answers [57, 61, 69]. Multimodal fusion utilizing product images as visual features are learned to integrate visual semantics for products [10, 20, 36, 41, 42, 60, 62, 80]. Some studies formulate AVE as a multi-label classification task to extract multiple aspects for the products [5, 11, 21]. To handle unseen attribute values, open mining models [70, 77] extract aspects directly from the text with limited/weak supervision, and generation models [58] decode aspects as target sequences. However, all of these approaches (1) require large quantities of labeled data for training and (2) miss higher-order relations between products, such as 'also buy' or 'also view' products.

2.2 Zero-shot Learning

Zero-shot learning has been widely applied in the field of computer vision (CV) [47] and natural language processing (NLP) [2]. Existing works for zero-shot learning in information extraction can be roughly divided into three categories: (1) Embedding-based models, where representations of both seen and unseen classes are learned based on the auxiliary information such as class information [1, 55] and other external information [22, 37]. However, high-quality external knowledge is required for training the model, resulting in an increase in training time and resources. (2) Generative-based models, where augmented samples are generated for unseen classes by generation models (i.e. GAN [45], VAEs [31], and GPT-2 [49]) based on the samples of seen classes. Then, the zero-shot learning problem is converted into a conventional supervised learning problem [9, 23]. However, these models suffer from the noise of augmented samples and performance highly depends on generative models. (3) Graph-based models, where GNNs [56] are directly used to predict unseen classes by inductive link prediction [2]. Most studies view this problem as zero-shot knowledge graph completion [18] or zero-shot item recommendation [15]. Attentive GCN is used to transfer features from seen classes to unseen classes [24]. Ontologies or topologies are utilized to augment ZSL by capturing relationships between classes [7, 17]. Motivated by this, we build a product heterogeneous hypergraph to identify unseen aspects with inductive inference ability while capturing higher-order relations.

2.3 Heterogeneous Hypergraph

Hypergraphs are generalizations and extensions of ordinary graphs, where hyperedges can accommodate an arbitrary number of nodes to capture the higher-order relations [76]. To handle different types

of nodes and edges, heterogeneous hypergraphs are learned by attention mechanisms [13, 30, 33, 40], wavelets [59], and variational auto-encoder [14, 39]. Though, all of these works are widely applied for social networks [34, 64], academic citations [65, 67, 75], biological networks [25, 44] or product recommendation in e-commerce [4, 8, 38, 71], heterogeneous hypergraphs are never applied to attribute value extraction in e-commerce. Different from the above hypergraphs that build hyperedges by close neighbors or meta-paths, we construct e-commerce related hyperedges by using user behavior and product inventory data to capture higher-order relations among categories, products, and aspects, to recognize unseen attribute values for new products.

3 METHODOLOGY

3.1 Problem Definition

In this section, we introduce the problem statement and some necessary definitions and notations for heterogeneous hypergraphs and multi-label zero-shot learning.

Problem Statement. Let $D = \{c_i, p_i, a_i\}$ denote a corpus of e-commerce product records, where c_i, p_i, a_i represent sub-category, product and attribute value pair (aspect), respectively. We use C, P, and A to denote the sets of sub-categories, products, and aspects. Hence, the task of attribute value extraction can be formulated as follows: Input: The product records D. Output: A model to estimate the probability that a new product p in sub-category c will have the unseen attribute value a. The goal of attribute value extraction is to learn a model $\mathcal{M}(p_i, a_j) \rightarrow \hat{y}\,[0, 1]$ to score the probability that a product p_i has the attribute value a_j based on $\mathcal{G}$, which includes all the relations from user behavior and product inventory information. Given several different graphs (i.e. user behavior graphs, product inventory graphs, etc.), we first build a heterogeneous hypergraph $\mathcal{G}$ to capture the higher-order and non-binary relations contained in $\mathcal{G}$. Then, we aim to learn the representations for nodes on a heterogeneous hypergraph $\mathcal{G}$ for an inductive link prediction task.

DEFINITION 1 (Heterogeneous Hypergraph): A heterogeneous hypergraph can be defined as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{T}_v, \mathcal{T}_e, W\}$, where $\mathcal{V} = \{v_1, v_2, \cdots, v_N\}$ is the node set, and $\mathcal{T}_v$ is the node type set. $\mathcal{E} = \{e_1, e_2, \cdots, e_M\}$ is the hyperedge set, and $\mathcal{T}_e$ is the hyperedge type set, where $|\mathcal{T}_v| + |\mathcal{T}_e| > 2$. N and M represent the maximum numbers of hyperedge nodes and edges. $W = diag(w_{e_1}, w_{e_2}, \cdots, w_{e_M})$ denotes the diagonal matrix representing the hyperedge weight. We use incidence matrix $H \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{E}|}$ to represent relationships between nodes and hyperedges, with entries defined as:

$$H(v, e) = \begin{cases} 1 & \text{if } v \in e \\ 0 & \text{if } otherwise. \end{cases} \tag{1}$$

$D_v \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ and $D_e \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{E}|}$ are the diagonal matrices representing the degree matrix of nodes and hyperedges, where $D_v(i, i) = \sum_{e\in\mathcal{E}} W(e)H(i, e)$ and $D_e(i, i) = \sum_{v\in\mathcal{V}} H(v, i)$. The normalized hypergraph adjacency matrix $A \in \mathbb{R}^{\mathcal{V}\times\mathcal{V}}$, representing the connection relationship between nodes, is defined as:

$$A = D_v^{-1/2} H W D_e^{-1} H^T D_v^{-1/2} \tag{2}$$

DEFINITION 2 (Zero-Shot Learning in Graph): For multi-label zero-shot attribute-value (aspect) prediction, let $A^s = \{a_1^s, \cdots, a_m^s\}$ and $A^u = \{a_1^u, \cdots, a_m^u\}$ denote the node sets of seen and unseen aspects, where $A^s \cap A^u = \emptyset$. Only A^s is included in the training graph $\mathcal{G}_{tr}$ and only A^u is included in the testing graph $\mathcal{G}_t$. Product p_i with any a_i^u will be removed from $\mathcal{G}_{tr}$ to $\mathcal{G}_t$, to ensure all unseen aspect nodes are not in the training graph $\mathcal{G}_{tr}$. Details for multi-label zero-shot sampling are introduced in Algorithm 1.

3.2 Multi-Label Zero-Shot Data Sampling

Multi-label zero-shot data sampling includes (1) data splitting to ensure that there is **no overlap** of aspect and product nodes in training and validation/testing sets, and (2) negative sampling to balance the dataset. For data splitting, we first randomly generate N aspect nodes A_N as unseen attribute values. Then, we remove both the nodes A_N and their corresponding products P_M as unseen products, and all edges on A_N and P_M from the original graph $\mathcal{G}$, where $N \neq M$. This step ensures that the zero-shot products and attribute values are never shown in the training graph. We update the validation and testing graphs with the zero-shot nodes and links separately so that there's no overlap of zero-shot nodes and links between the validation and testing sets. To balance the dataset, we do negative sampling and add negative links for all training, validation, and testing graphs. Details for multi-label zero-shot data sampling are shown in Algorithm 1.

Algorithm 1: Multi-label Zero-shot Data Sampling

```
Input  : Graph G with categories nodes C, product nodes P
         and aspect nodes A, unseen aspect number N
Output : Train graph G_tr, val graph G_v, test graph G_t
Initialize G_tr, G_v, G_t
for i in Random(N) do
    P_i = get_node(G, A_i)
    link_pos = get_edge(G, P_i, A_i)
    link_neg = Sampling(get_complement(link_pos))
    G.remove(A_i, P_i, link_pos)
    if i//2=0 then
        G_v.update(A_i, P_i, link_pos, link_neg)
    else
        G_t.update(A_i, P_i, link_pos, link_neg)
    G_tr = G.add_negatives()
return G_tr, G_v, G_t
```

3.3 Overall Framework

Figure 2 shows our proposed framework HyperPAVE with three main components: a) hypergraph construction, b) heterogeneous hypergraph relation learning, and c) inductive link prediction. We introduce each component in detail below.

3.3.1 Heterogeneous Hypergraph Construction. As shown in Figure 2(a), there are three types of nodes: categories, products, and attribute values (aspects), and four types of hyperedges: 'also view', 'also buy', 'product with all aspects' and 'category with all products and aspects', which are constructed from two main data sources: user behavior information and product inventory information as:

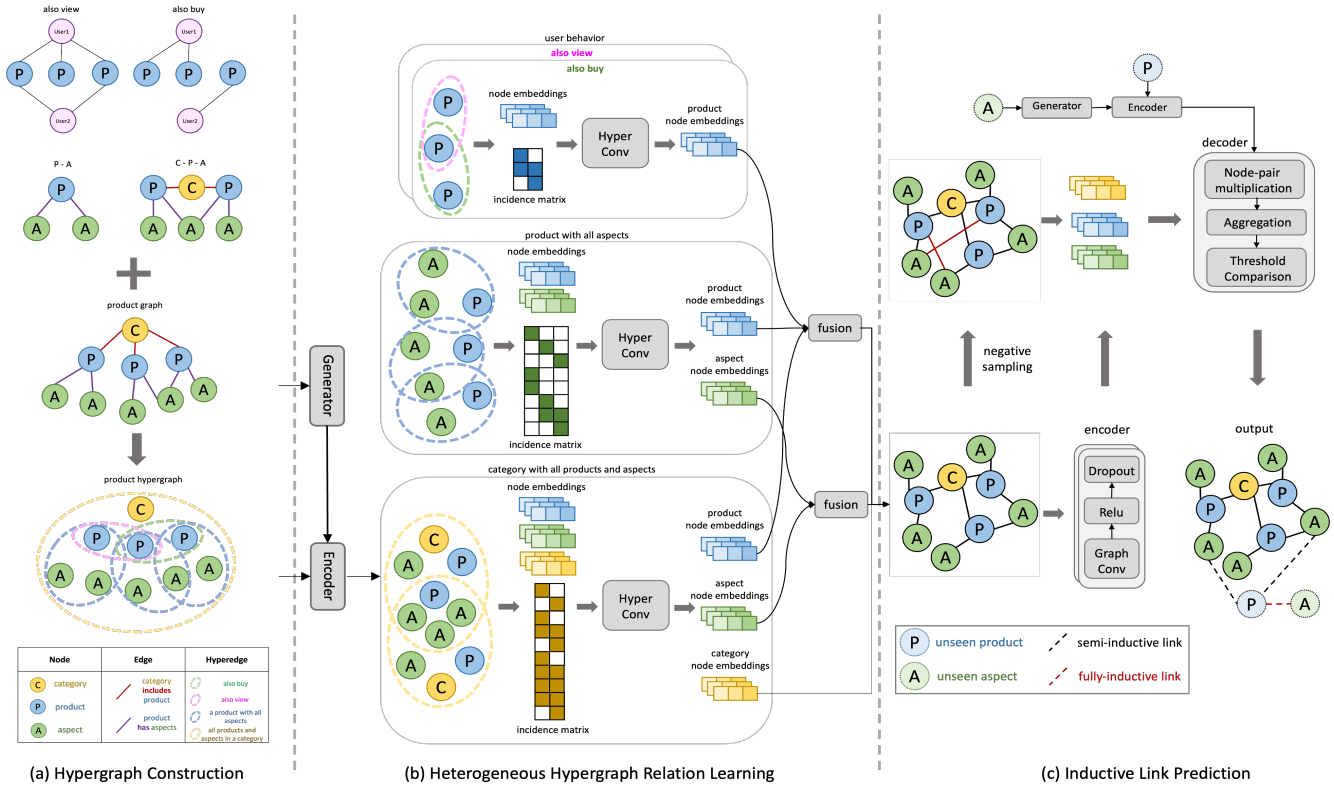

Figure 2: Overall framework of our proposed model HyperPAVE. The framework includes three key components: (a) Hypergraph Construction (b) Heterogeneous Hypergraph Relation Learning and (c) Inductive Link Prediction.

1) User Behavior Data. User behaviors have multiple types related to item-to-item relationships: people who bought X also bought Y ('also buy') and people who viewed X also viewed Y ('also view'). To well handle different user behaviors, we construct two types of hyperedges $\mathcal{T}_e^u = \{\mathcal{E}^V, \mathcal{E}^B\}$, where $\mathcal{E}^V$ represents 'also view' and $\mathcal{E}^B$ represents 'also buy'. For example, given the record of user1 in 'also view' graph shown in Figure 2(a), we construct a hyperedge $\mathcal{E}_i^V = \{p_1, p_2, \cdots, p_n\} \in \mathcal{E}^V$ to model the interactions between users and products. That is, each hyperedge in $\mathcal{E}^V$ corresponds to one user. These hyperedges are homogeneous because all nodes represent products.

2) Product Inventory Data. Product inventory data refers to the existing product information records, including category, product, attribute values, etc. We construct hyperedges $\mathcal{E}^P$ to connect all attribute values to one product (P-A) and hyperedges $\mathcal{E}^C$ to connect all product information to one sub-category (C-P-A). For example, given a product p_i, we construct a hyperedge $\mathcal{E}_i^P = \{p_i, a_1, a_2, \cdots, a_n\} \in \mathcal{E}^P$ to indicate the relationships between product and its attribute values. These heterogeneous hyperedges record the non-binary relations among categories, products, and attribute values. To summarize it, we obtain hyperedge sets as:

$$\mathcal{T}_e = \left\{\mathcal{E}^V, \mathcal{E}^B, \mathcal{E}^P, \mathcal{E}^C\right\} \tag{3}$$

3.3.2 Heterogeneous Hypergraph Relation Learning.

Embedding Module. As shown in Figure 2(b), a heterogeneous hypergraph encoder first initializes the node embeddings. Since the attribute values (aspects) may lose contextual information due to the simple format, GPT-2 [49] is adopted as the text generator to generate more detailed descriptions for attribute values. For example, the attribute value: 'connectivity: wireless' can be elaborated to a more detailed explanation: 'connectivity is wireless communication between the user's device, which has an independent, physical signal to the user'. We then adopt a pre-trained language model BERT [12] as all nodes' input encoder to generate the initial contextual representation. For the product node, we construct a string [CLS;t;SEP;d] by concatenating the product title and description as the input, where CLS and SEP are special tokens. The initial output representation for the category node c_i, product node p_i and aspect node a_i can be formulated as follows:

$$h_{v_{c_i}} = tanh(W \cdot f_\varnothing(c_i) + b) \tag{4}$$

$$h_{v_{p_i}} = tanh(W \cdot f_\varnothing(t_i, d_i) + b) \tag{5}$$

$$h_{v_{a_i}} = tanh(W \cdot f_\varnothing(g_\varnothing(a_i)) + b) \tag{6}$$

where $f_\varnothing$ is BERT encoder, $g_\varnothing$ is GPT-2 generator, c is category, t is product title, d is product description, a is 'attribute value', W and b are trainable weights and bias. To simplify the notations, we use h_{v_i} to denote the initial feature embeddings of all different nodes.

Table 1: Dataset statistics over ten categories. The number of hyperedges is reported in the format of: #nodes / #hyperedges.

Category	Number of Nodes			Number of Edges			Number of Hyperedges			
	#C	#P	#A	#CP	#PA	Density	P-P$_{\text{also view}}$	P-P$_{\text{also buy}}$	P-A	C-P-A
Arts	980	11,625	2,184	50,652	28,932	7.2×10^{-4}	970/624	1,448/1,248	13,809/11,643	14,789/979
Books	410	16,220	255	48.271	23,438	5.03×10^{-4}	1,247/1,433	2,432/2,550	16,475/16,222	16,885/409
Cellphones	145	8,499	1,484	27,620	20,329	9.35×10^{-4}	366/362	171/160	9,983/8,507	10,128/144
Giftcards	5	131	11	378	311	0.06	17/20	19/32	142/130	147/1
Grocery	742	18,315	4,686	75,362	47,745	4.37×10^{-4}	3,162/2,431	3,392/3,314	23,001/4,686	23,743/741
Industrial	433	3002	1573	12,429	8,453	1.67×10^{-3}	152/106	210/205	4,539/3,063	5,008/432
Pet	508	14,299	2,575	64,947	46,370	7.34×10^{-4}	1,614/1,670	820/600	16,874/14,675	17,382/507
Software	303	254	98	1,182	607	8.35×10^{-3}	19/20	2/1	352/287	655/302
Tools	975	34,076	7,538	143,683	101,475	2.7×10^{-4}	3,176/2,648	1,998/1,704	41,614/34,705	42,589/974
Videogames	910	731	353	4,446	2,152	3.32×10^{-3}	113/139	14/9	1,084/752	1,994/909

Message Passing Module. To support representation learning on the constructed heterogeneous hypergraphs in the previous step, we design a heterogeneous hypergraph relation learning module (shown in Figure 2(b) in HyperPAVE to explore the complex higher-order relationships based on many-to-many node message passing in the product graph by taking full advantage of the structure information in Figure 2(a). HyperPAVE learns node representations with two different aggregation functions:

$$h^l_{v_i} = AGGR^l_{edge}\left(h^{l-1}_{v_i}, \left\{h^l_{e_j} | \forall_{e_j} \in \mathcal{E}_i\right\}\right) \tag{7}$$

$$h^l_{e_j} = AGGR^l_{node}\left(\left\{h^{l-1}_{v_k} | \forall_{v_k} \in e_j\right\}\right) \tag{8}$$

where $AGGR$ is the aggregation function, $\mathcal{E}_i$ is the hyperedge sets connected to node v_i and $h^l_{e_j}$ is the representation of hyperedge e_j in layer l. Since not all the nodes in a hyperedge will contribute equally, the message passing is calculated from nodes to hyperedges:

$$\alpha^{e_i}_{v_i} = \frac{exp(LeakyReLU(w_1^T \cdot h^{l-1}_{v_i}))}{\sum_{v\in V_{e_i}} exp(LeakyReLU(w_1^T \cdot h^{l-1}_{v}))} \tag{9}$$

$$h^l_{e_i} = ||_{n=1}^{N} \sigma(\sum_{v\in V_{e_i}} \alpha^{e_i}_{v} \cdot h^{l-1}_{v}) \tag{10}$$

where $\alpha^{e_i}_{v_i}$ is the weight factor of node v_i to hyperedge e_i, V_{e_i} is the node set of hyperedges e_i, w_1^T is a trainable attention parameter, $||$ denotes concatenation with N heads, and σ is a non-linear function. $h^l_{e_i}$ is the lth layer of hyperedge representation. Similarly, the message passing from hyperedges to nodes is calculated as:

$$\alpha^{v_i}_{e_i} = \frac{exp(LeakyReLU(w_2^T \cdot (h^{l-1}_{v_i}||h^{l-1}_{e_i})))}{\sum_{e\in \mathcal{E}_{v_i}} exp(LeakyReLU(w_1^T \cdot (h^{l-1}_{v_i}||h^{l-1}_{e})))} \tag{11}$$

$$h^l_{v_i} = ||_{n=1}^{N} \sigma(\sum_{e\in \mathcal{E}_{v_i}} \alpha^{v_i}_{e} \cdot h^{l-1}_{e}) \tag{12}$$

where $\alpha^{v_i}_{e_i}$ is the weight factor of hyperedge e_i to node v_i, $\mathcal{E}_{v_i}$ is the connected hyperedge set of node v_i. w_2^T is a trainable attention parameter and $h^l_{v_i}$ is the lth layer of node representation, which includes the information from the hyperedge $\mathcal{E}$.

Fusion Module. Instead of directly adding a readout layer and a linear prediction layer after obtaining the L layers node representations [71], we argue that different types of hyperedges from $\mathcal{T}_e$ have different importance to the final node representations. Thus, we propose fusion modules to fuse node representations learned from different hypergraphs constructed in Sec. 3.3.1. The updated node representations for product node $\hat{h_{v_{p_i}}}$ and aspect node $\hat{h_{v_{a_i}}}$ are:

$$\hat{h_{v_{p_i}}} = \alpha \cdot h^{\mathcal{E}^{\mathcal{P}}}_{v_{p_i}} + \beta \cdot h^{\mathcal{E}^{C}}_{v_{p_i}} + (1-\alpha-\beta)(\gamma \cdot h^{\mathcal{E}^{\mathcal{V}}}_{v_{p_i}} + (1-\gamma) \cdot h^{\mathcal{E}^{\mathcal{B}}}_{v_{p_i}}) \tag{13}$$

$$\hat{h_{v_{a_i}}} = \delta \cdot h^{\mathcal{E}^{\mathcal{P}}}_{v_{a_i}} + (1-\delta) \cdot h^{\mathcal{E}^{C}}_{v_{a_i}} \tag{14}$$

where $h^{\mathcal{E}^{\mathcal{P}}}_{v_{p_i}}, \cdot h^{\mathcal{E}^{C}}_{v_{p_i}}, h^{\mathcal{E}^{\mathcal{V}}}_{v_{p_i}}, h^{\mathcal{E}^{\mathcal{B}}}_{v_{p_i}}$ are product node representations and $h^{\mathcal{E}^{\mathcal{P}}}_{v_{a_i}}, h^{\mathcal{E}^{C}}_{v_{a_i}}$ are aspect node representations from different hyperedges in Equ. 3, respectively. α, β, γ, and δ are weights learnt from the validation sets. They are different for different categories of the dataset. These weights are also explored and studied in Sec. 4.2.4. After the above fusion steps, the node embeddings contain the features from neighbors defined by different hyperedges $\mathcal{T}_e$, which can well capture the high-order relations communicated among different types of nodes and hyperedges.

3.3.3 Inductive Link Prediction. After heterogeneous hypergraph relation learning, each node includes the higher-order features related to user behavior and product inventory information. Then, all the nodes go through L GNN layers to compute the final node representations. After generating the final embeddings of $\tilde{h_{v_p}}$ and $\tilde{h_{v_a}}$, the likelihood of the link between product p and aspect a is measured by the cosine similarity to decide the possibility $\hat{R}_{ij}$ of whether product p_i will have the aspect a_j:

$$f_{score}((\tilde{h_{v_p}})_i, (\tilde{h_{v_a}})_j) = \frac{(\tilde{h_{v_p}})_i \cdot (\tilde{h_{v_a}})_j}{\left\|(\tilde{h_{v_p}})_i\right\| \left\|(\tilde{h_{v_a}})_j\right\|} \tag{15}$$

We use the negative sampling strategy introduced in Sec. 3.2 to train HyperPAVE and employ a binary cross entropy loss to optimize our model:

$$\mathcal{L} = \sum_{p_i\in P, a_i\in A} R_{ij} log\hat{R}_{ij} + (1-R_{ij})(1-log\hat{R}_{ij}) \tag{16}$$

Note that HyperPAVE follows the multi-label zero-shot settings in Sec. 3.2 to eliminate the mandatory access of testing node features during training, making the model access the inductive inference ability. For unseen attribute values (aspects) and products, we can directly feed their corresponding contextual node embeddings by fine-tuned BERT encoder to HyperPAVE instead of representing

Table 2: Experimental Results F1 / mAP (%) of multi-label zero-shot learning over ten categories on MAVE. The results are reported as mean ± standard deviation over ten times of experiments. The best results are in bold.

	Arts	Books	Cellphones	Giftcards	Grocery
BERT-MLC [6]	24.11±0.09 / 10.31±0.16	36.72±0.08 / 27.17±0.37	22.92±0.25 / 28.67±0.37	36.54±0.07 / 41.15±0.08	19.74±0.24 / 12.07±0.09
Bart [32]	27.88±0.36 / 23.16±0.46	38.82±0.44 / 44.90±0.20	32.71±0.35 / 24.54±0.37	15.73±0.19 / 8.75±0.36	10.80±0.18 / 6.95±0.12
$T5_{small}$ [50]	30.85±0.31 / 23.16±0.17	36.17±0.45 / 42.60±0.13	30.95±0.31 / 24.27±0.30	10.14±0.26 / 8.08±0.23	23.53±0.27 / 17.32±0.25
HGCN [51]	16.87±0.10 / 25.30±0.33	39.39±0.18 / 37.40±0.12	17.23±.24 / 14.67±0.31	30.92±0.07 / 45.42±0.06	25.60±0.13 / 39.77± 0.18
HAN [63]	14.26±0.10 / 26.42±0.25	43.73±0.16 / 49.48±0.16	22.49±0.20 / 33.69±0.17	42.47±0.31 / 54.05±0.13	17.23±0.20 / 34.67±0.24
HGT [27]	30.81±0.13 / 38.53±0.16	48.06±0.11 / 41.67±0.17	14.53±0.16 / 23.73±0.20	42.30±0.40 / 42.39±0.19	27.30±0.11 / 40.76±0.24
HGNN+ [16]	27.90±0.28 / 36.91±0.13	46.79±0.20 / **58.33±0.15**	32.10±0.17 / **36.40±0.26**	37.18±0.07 / 57.20±0.04	32.40±0.14 / 38.60±0.15
HyperGCN [72]	20.20±0.17 / 38.45±0.21	48.97±0.13 / 45.18±0.16	20.90±0.25 / 26.00±0.40	**52.74±0.19** / 45.97±0.09	**35.90±0.22** / 42.20±0.21
HyperPAVE	**43.33±0.22 / 40.99±0.18**	**49.75±0.18** / 56.45±0.11	**39.01±0.16** / 35.81±0.18	52.34±0.22 / **65.03±0.13**	33.43±0.28 / **42.71±0.30**
	Industrial	Pet	Software	Tools	Videogames
BERT-MLC [6]	10.94±0.19 / 6.69±0.16	18.14±0.55 / 12.08±0.16	27.76±0.09 / 25.37±0.09	20.43±0.26 / 18.41±0.17	11.86±0.31 / 9.66±0.35
BART [32]	10.78±0.32 / 7.84±0.32	12.50±0.25 / 10.42±0.67	22.50±0.03 / 20.00±0.02	11.11±0.16 / 6.25±0.09	23.57±0.32 / 20.02±0.25
$T5_{small}$ [50]	15.81±0.47 / 15.35±0.16	25.28±0.20 / 25.72±0.26	26.19±0.42 / 24.60±0.31	**37.78±0.26** / 22.46±0.52	14.41±0.15 / 9.90±0.27
HGCN [51]	10.67±0.24 / 14.60±0.14	17.62±0.15 / 24.63±0.24	19.29±0.22 / 30.97±0.15	18.07±0.20 / 39.32±0.18	8.78±0.40 / 13.61±0.25
HAN [63]	15.35±0.20 / 30.45±0.50	16.82±0.13 / 23.33±0.25	28.24±0.31 / 29.03±0.14	19.78±0.03 / 41.40±0.14	9.68±0.16 / 16.29±0.21
HGT [27]	21.09±0.13 / 23.20±0.16	18.02±0.13 / 23.66±0.20	30.15±0.20 / 27.16±0.08	13.61±0.18 / 35.23±0.22	14.75±0.05 / 19.97±0.11
HGNN+ [16]	25.90±0.26 / 28.60±0.12	27.60±0.14 / 35.58±.16	39.90±0.26 / 28.76±.16	31.00±0.15 / 42.20±0.23	10.35±0.11 / 17.21±0.08
HyperGCN [72]	**29.20±0.13** / 33.20±.11	22.20±0.12 / 31.37±0.14	42.10±0.31 / 38.70±0.13	31.10±0.18 / 44.05±0.19	10.90±0.13 / 15.30±0.10
HyperPAVE	27.70±0.10 / **33.29±0.17**	**28.45±0.13 / 38.46±0.20**	**47.62±0.21 / 51.64±0.10**	34.00±0.28 / **47.83±0.29**	**25.31±0.19 / 21.19±0.17**

product and aspect nodes with one-hot vectors. Then, we only conduct message-passing and compute the probability of connections between the product node and the aspect node. Hence, we can handle the newly added products and attribute values in an inductive way instead of retraining the model.

4 EXPERIMENTS

4.1 Experimental Setup

4.1.1 Dataset. We evaluate our model [1] over ten different categories (Arts, Books, Cellphones, etc) of a public dataset MAVE [74], which is a large e-Commerce dataset derived from Amazon Review Dataset [46]. To simulate the zero-shot situation, we reconstruct the dataset into multi-label zero-shot learning settings followed by Sec. 3.2, where there is no overlap of products and attribute values between the training set and validation/testing set. Note that each time we train the model, the dataset will be randomly re-splitted for the zero-shot setting, so we report the whole data statistics in Table 1. A sample of data statistics for training, validation, and testing sets for each category is shown in Appendix 6.1.

4.1.2 Evaluation Metrics. Following other AVE tasks in the multi-label zero-shot setting [58], we choose to report macro-F1 and mAP (mean Average Precision) compared with classification and generation-based models in the main results as F1 score is the balance of both precision and recall. In Sec. 4.2.2 ablation study, we also report AUC (Area Under Curve), Hits@K, NDCG@K (Normalized Discounted Cumulative Gain), and MRR (Mean Reciprocal Ran), which are widely used metrics in graph-based recommendation tasks [15, 26, 35]. We also report training time to evaluate the efficiency in Sec. 4.2.3 efficiency study.

4.1.3 Baselines. We compare our proposed model HyeprPAVE with the following baselines in the zero-shot setting:

- Classification-based Models: Original classification-based models do not have any zero-shot abilities. We follow the baseline **BERT-MLC** in [58], then we add synthetic data for unseen classes (attribute values) following [9]. In this way, the zero-shot learning problem is translated into a supervised learning problem.
- Generation-based Models: Following generative models in zero-shot AVE task [58], we implement and fine-tune two text-to-text transformer-based encoder decoder architecture models: **BART** [32] and **$T5_{small}$** [50], to generate unseen attribute values directly.
- Graph-based Models [2]: As inductive graph can predict unseen nodes (zero-shot learning), we compare HyperPAVE with three heterogeneous graph neural networks: **HGCN** [51], **HAN** [63], **HGT** [27], and two representative hypergraph networks: **hyperGCN** [72], **HGNN+** [16].

4.2 Results and Discussions

4.2.1 Main Results. From the results shown in Table 2 and data statistics shown in Table 1, we observe that:

(1) The classification-based model generally performs worst among all models. BERT-MLC, which uses synthetic data for zero-shot prediction, only has competitive performance to generation-based models when the class number (#A) is small. We conjecture that as the number of classes grows, BERT-MLC needs to make distinctions among more classes, making it harder to find clearer decision boundaries. The average micro F1 of BART and $T5_{small}$ across all ten categories is worse than $T5_{base}$ in [58]. This is because $T5_{base}$ is pre-trained over 220 million parameters whereas $T5_{small}$ has only 60 million parameters. Generation-based models perform much better than classification-based models in most cases. BART and $T5_{small}$ show different performances over different categories.

[1]The code is available: https://github.com/gjiaying/HyperPAVE.

[2]Implemented on DHG: https://deephypergraph.com/

Table 3: Ablation study over HyperPAVE components in the zero-shot setting across three categories on MAVE dataset.

	F1	mAP	AUC	MRR	NDCG	Hits@5	Hits@10	Hits@100
	Books							
nodeID	11.54 ± 1.59	28.52 ± 1.29	95.31 ± 1.15	6.64 ± 1.07	48.41 ± 0.97	35.26 ± 0.63	53.85 ± 0.62	99.42 ± 0.05
BERT	23.87 ± 1.29	38.63 ± 0.57	97.07 ± 0.60	11.39 ± 0.83	57.31 ± 0.65	47.05 ± 0.42	63.59 ± 0.40	100.00 ± 0.00
BERT (Fine-tuned)	28.28 ± 0.81	40.32 ± 0.59	97.87 ± 0.21	14.44 ± 0.40	58.89 ± 0.41	50.90 ± 0.82	78.33 ± 0.38	100.00± 0.00
Hyper (Product)	30.44 ± 0.25	40.65 ± 0.41	98.03 ± 0.19	14.23 ± 0.19	59.49 ± 0.30	49.36 ± 0.14	80.51 ± 0.17	100.00± 0.00
Hyper (Behavior)	34.46 ± 0.29	35.93 ± 0.49	**98.40 ± 0.20**	19.37 ± 0.27	54.23 ± 0.31	63.67 ± 0.40	93.67 ± 0.24	100.00± 0.00
HyperPAVE	**49.75 ± 0.18**	**56.45 ± 0.11**	96.47 ± 0.02	**32.99 ± 0.14**	**69.35 ± 0.18**	**85.27 ± 0.12**	**94.04 ± 0.08**	**100.00 ± 0.00**
	Giftcards							
nodeID	6.67 ± 0.21	22.35 ± 0.20	41.94 ± 0.29	18.18 ± 0.30	42.92 ± 0.15	25.00 ± 0.00	97.50 ± 0.08	100.00± 0.00
BERT	26.41 ± 0.17	44.79 ± 0.14	71.94 ± 0.05	24.15 ± 0.18	62.47 ± 0.13	75.00 ± 0.00	100.00± 0.00	100.00± 0.00
BERT (Fine-tuned)	34.43 ± 0.17	41.67 ± 0.15	71.53 ± 0.16	23.17 ± 0.30	59.57 ± 0.11	67.50 ± 0.12	100.00± 0.00	100.00± 0.00
Hyper (Product)	39.77 ± 0.12	45.74 ± 0.10	84.55 ± 0.10	35.65 ± 0.17	61.83 ± 0.08	100.00± 0.00	100.00± 0.00	100.00± 0.00
Hyper (Behavior)	45.43± 0.15	60.50 ± 0.13	77.92 ± 0.12	29.13 ± 0.15	73.02 ± 0.07	71.00 ± 0.12	100.00± 0.00	100.00± 0.00
HyperPAVE	**52.34 ± 0.22**	**65.03 ± 0.13**	**90.08 ± 0.05**	**44.56 ± 0.16**	**75.07 ± 0.11**	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**
	Pets							
nodeID	6.95 ± 0.82	13.46 ± 0.92	98.51 ± 0.27	7.47 ± 0.19	42.17 ± 0.62	30.33 ± 0.46	50.00 ± 0.13	96.15 ± 0.10
BERT	9.93 ± 0.30	19.93 ± 0.51	99.73 ± 0.20	6.71 ± 0.27	41.16 ± 0.68	31.67± 0.54	65.00 ± 0.50	100.00± 0.00
BERT (Fine-tuned)	12.12 ± 0.29	19.99 ± 0.74	99.39 ± 0.11	10.79 ± 0.37	40.52 ± 0.61	25.00± 0.29	56.67 ± 0.14	100.00± 0.00
Hyper (Product)	17.58 ± 0.26	37.40 ± 0.32	99.03 ± 0.09	16.45 ± 0.19	45.67 ± 0.21	41.67 ± 0.17	**71.67 ± 0.15**	98.33 ± 0.12
Hyper (Behavior)	18.66 ± 0.14	24.71 ± 0.14	99.16 ± 0.10	19.01 ± 0.28	42.38 ± 0.35	36.07 ± 0.22	65.00 ± 0.09	100.00± 0.00
HyperPAVE	**28.45 ± 0.13**	**38.46 ± 0.20**	**99.82 ± 0.06**	**29.92 ± 0.13**	**61.55 ± 0.20**	**56.67 ± 0.09**	67.77 ± 0.03	**100.00± 0.00**

They can achieve similar performance with HyperPAVE when the dataset size is large enough. (2) Combining inductive graph-based models with LLM encoders can perform zero-shot prediction and achieve competitive performance with generative models. This inspires us that instead of fine-tuning the popular generative models [53, 54, 58, 78] to extract attribute values, the inductive graph for link prediction can also be explored for zero-shot prediction. Besides, using the attention mechanism shows better performance than using fixed and uniform weights for aggregation. This is probably because assigning different weights to neighboring nodes can capture varying levels of influence. (3) Compared with graph-based baselines, adding complex structured data to capture higher-order relationships demonstrates significant performance improvement over all ten categories. This is probably because hyperedges can model relationships that go beyond pairwise connections, resulting in more semantic node representations. Besides, our proposed model HyperPAVE achieves the best performance among all models in most categories, indicating that our proposed hypergraph construction from both user behavior data and product inventory data is important and worth recording and exploring. The effectiveness of different hyperedges is studied in Sec. 4.2.2.

4.2.2 Ablation Study. To evaluate the performance of each component in HyperPAVE, we conduct an ablation study over three categories in the zero-shot setting. Table 3 shows the performance of each component in HyperPAVE. More results are shown in Table 7 in the Appendix. We have the following observations based on Table 3: (1) Adding node features can significantly improve the performance. We perform a model 'nodeID', which doesn't use any pre-trained encoder for providing node features. The model 'nodeID' uses a simple embedding-lookup encoder, mapping each node to a unique low-dimensional vector. We can observe that among all models, 'nodeID' shows the worst performance. After adding node features, such as BERT or fine-tuned BERT, the performance increases significantly. We think that this is because, for link prediction in the zero-shot setting, pre-trained embeddings provide richer and more semantically meaningful representations for node features in graphs than simple one-hot encoding. (2) Fine-tuning the pre-trained encoders for node features results in a big performance improvement when the dataset (graph) is large enough. This is reasonable because a larger dataset (graph with more nodes) provides more diverse and representative data, enabling better generalization for unseen nodes in the zero-shot setting. However, as shown in Sec. 4.2.3, fine-tuning the pre-trained encoder may result in more time for model training. A balance of model performance and efficiency needs to be considered for different tasks/situations. (3) We explore the importance of different hypergraphs. We find out that adding different hyperedges built from user behavior data or product inventory data results in a significant performance improvement. We conjecture that this is because different hyperedges capture more complex higher-order information than the original binary-relation graph. For example, hyperedge 'P-P_{also_view}' built from user behavior data includes information on products with potential similar attributes because users may probably view similar products at the same time for their needs. Hyperedge 'C-P-A', built from product inventory data, aggregates all products and aspects in the same sub-category. Attribute values such as 'Chew Type: Bones' may only happen in a sub-category of 'Dog Treats' instead of 'Cat Food'. By using hyperedges, more complex relations can be included in the representation for each single node.

4.2.3 Efficiency Study. Table 4 presents the GPU computational cost and model parameter comparison between classification-based (BERT-MLC), generation-based (BART/$T5_{small}$) and graph-based (nodeID/HyperPAVE) models on Arts category of MAVE. Different categories (different sizes of graphs) may result in a slight difference.

Table 4: Comparison of computational efficiency. The batch size is set to 4.

Model	Memory Consumption	Model Parameters
Classification-based	5037MB	110M
Generation-based	8305MB / 5831MB	140M / 60M
Graph-based (ours)	1405MB / 1915 MB	5M / 115M

From the reported results, we can find that compared with classification or generation-based models, our proposed graph-based model HyperPAVE, has a significant computational advantage in terms of memory consumption. The main reason is that the zero-shot ability of generative LLMs is based on their extensive pretraining and understanding of the diverse data. When fine-tuning these LLMs, large quantities of model parameters need to be updated, resulting in a huge GPU memory consumption cost. However, the zero-shot ability of HyperPAVE results from the inductive inference that can generalize to unseen product and aspect nodes without retraining the whole model. The inductive HyperPAVE divides the hypergraphs into batches and only consumes per-batch memory when training. Note that for the classification model BERT-MLC, preprocessing steps for generating synthetic data from generation models are required to predict unseen aspects. We have not counted the computational cost for these preprocessing steps.

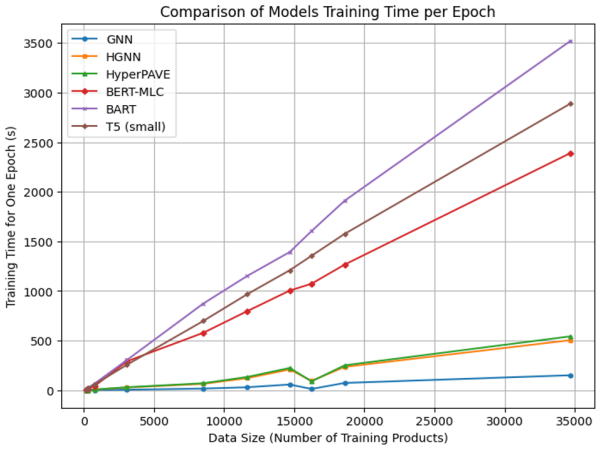

Figure 3: Time Efficiency Performance (GPU Time of Model Learning in Seconds for One Training Epoch).

To evaluate the computation time of our graph-based model HyperPAVE and other classification-based and generation-based models, we record the model training time for one epoch in seconds across the ten categories on MAVE as shown in Figure 3. All models use the same input max_length and batch size for training. From Figure 3, we observe that graph-based models show better model training efficiency. Compared with other graph-based models (i.e. GNN, HGNN), HyperPAVE can achieve the best prediction performance as shown in Table 2 with only sacrificing a little more time for training as shown in Figure 3. The slopes of BART, T5, and BERT-MLE are much larger than graph-based models, indicating that much more time is needed for training or fine-tuning with the increase in dataset size when updating the model parameters. More details are shown in Table 6 in Appendix 6.3.

4.2.4 Parameter Sensitivity Analysis. The key hyperparameters of HyperPAVE are the weights of the different hyperedges. Thus, we explore the importance of different types of hyperedges in the category of Giftcards as shown in Figure 4.

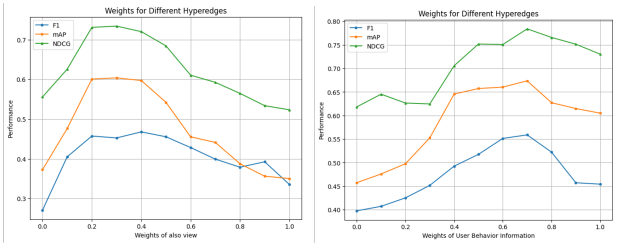

Figure 4: Effects on weights of different hyperedges on the category of giftcards.

The left figure explores the weights of 'P-$P_{\text{also view}}$' and 'P-$P_{\text{also buy}}$' hyperedges from user behavior information. The right figure explores the weights of user behavior hyperedges (P-P) and product inventory hyperedges ('P-A' and 'C-P-A'). From Figure 4, we observe that both 'P-$P_{\text{also view}}$' and 'P-$P_{\text{also buy}}$' contribute to the model's performance. The best weight for 'P-$P_{\text{also view}}$' falls in the [0.2, 0.5] interval, which means 'P-$P_{\text{also buy}}$' is slightly more important than 'P-$P_{\text{also view}}$'. This is probably because 'P-$P_{\text{also buy}}$' records users' history preference while 'P-$P_{\text{also view}}$' may include some noise such as accidental clicks. We can also observe from the right 4 that the best weight for user behavior data falls in the [0.6, 0.8] interval, indicating that user behavior is much more important than product inventory data. As shown in Table 1, the number of user behavior hyperedges is much smaller than the number of product inventory hyperedges ('P-A' and 'C-P-A'). But they show more importance in Figure 4, demonstrating that user behavior information is worth recording and exploring for extracting unseen attribute values for new products.

5 CONCLUSION AND FUTURE WORK

In this paper, we formulate the AVE task in a zero-shot learning scenario to identify unseen attribute values from new products with no corresponding labeled data available for training. We propose an inductive heterogeneous hypergraph (HyperPAVE) for multi-label zero-shot attribute value extraction. Specifically, the heterogeneous hypergraph captures the higher-order relationships among users and products, and the inductive mechanism infers the future connections between unseen nodes. Extensive experimental results on ten different categories across the public dataset MAVE demonstrate that our proposed model HyperPAVE outperforms other state-of-the-art classification-based and generation-based models. The ablation study validates the efficiency and effectiveness of different hypergraphs constructed from user behavior and product inventory data. We plan to explore the following directions in future work: (1) Including multimodal features (i.e. product images) as node attributes to capture more semantic information from the products. (2) Building dynamic graphs by including timestamps to make the product graph adapt to the developing market.

REFERENCES

[1] Chih-Yao Chen and Cheng-Te Li. 2021. ZS-BERT: Towards Zero-Shot Relation Extraction with Attribute Representation Learning. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Online, 3470–3479. https://doi.org/10.18653/v1/2021.naacl-main.272

[2] Jiaoyan Chen, Yuxia Geng, Zhuo Chen, Ian Horrocks, Jeff Z. Pan, and Huajun Chen. 2021. Knowledge-aware Zero-Shot Learning: Survey and Perspective. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, Zhi-Hua Zhou (Ed.). International Joint Conferences on Artificial Intelligence Organization, 4366–4373. https://doi.org/10.24963/ijcai.2021/597 Survey Track.

[3] Jiayi Chen and Aidong Zhang. 2020. Hgmf: heterogeneous graph-based fusion for multimodal data with incompleteness. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 1295–1305.

[4] Mengru Chen, Chao Huang, Lianghao Xia, Wei Wei, Yong Xu, and Ronghua Luo. 2023. Heterogeneous Graph Contrastive Learning for Recommendation. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining* (Singapore, Singapore) *(WSDM '23)*. Association for Computing Machinery, New York, NY, USA, 544–552. https://doi.org/10.1145/3539597.3570484

[5] Wei-Te Chen, Yandi Xia, and Keiji Shinzato. 2022. Extreme Multi-Label Classification with Label Masking for Product Attribute Value Extraction. In *Proceedings of The Fifth Workshop on e-Commerce and NLP (ECNLP 5)*. 134–140.

[6] Wei-Te Chen, Yandi Xia, and Keiji Shinzato. 2022. Extreme Multi-Label Classification with Label Masking for Product Attribute Value Extraction. In *Proceedings of the Fifth Workshop on e-Commerce and NLP (ECNLP 5)*. Association for Computational Linguistics, Dublin, Ireland, 134–140. https://doi.org/10.18653/v1/2022.ecnlp-1.16

[7] Ziyi Chen, Yutong Gao, Congyan Lang, Lili Wei, Yidong Li, Hongzhe Liu, and Fayao Liu. 2023. Integrating Topology beyond Descriptions for Zero-shot Learning. *Pattern Recognition* (2023), 109738.

[8] Dian Cheng, Jiawei Chen, Wenjun Peng, Wenqin Ye, Fuyu Lv, Tao Zhuang, Xiaoyi Zeng, and Xiangnan He. 2022. IHGNN: Interactive Hypergraph Neural Network for Personalized Product Search. In *Proceedings of the ACM Web Conference 2022*. 256–265.

[9] Yew Ken Chia, Lidong Bing, Soujanya Poria, and Luo Si. 2022. RelationPrompt: Leveraging Prompts to Generate Synthetic Data for Zero-Shot Relation Triplet Extraction. In *Findings of the Association for Computational Linguistics: ACL 2022*. Association for Computational Linguistics, Dublin, Ireland, 45–57. https://doi.org/10.18653/v1/2022.findings-acl.5

[10] Hejie Cui, Rongmei Lin, Nasser Zalmout, Chenwei Zhang, Jingbo Shang, Carl Yang, and Xian Li. 2023. PV2TEA: Patching Visual Modality to Textual-Established Information Extraction. arXiv:2306.01016 [cs.CL]

[11] Zhongfen Deng, Wei-Te Chen, Lei Chen, and S Yu Philip. 2022. AE-smnsMLC: Multi-Label Classification with Semantic Matching and Negative Label Sampling for Product Attribute Value Extraction. In *2022 IEEE International Conference on Big Data (Big Data)*. IEEE, 1816–1821.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[13] Kaize Ding, Jianling Wang, Jundong Li, Dingcheng Li, and Huan Liu. 2020. Be More with Less: Hypergraph Attention Networks for Inductive Text Classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 4927–4936. https://doi.org/10.18653/v1/2020.emnlp-main.399

[14] Haoyi Fan, Fengbin Zhang, Yuxuan Wei, Zuoyong Li, Changqing Zou, Yue Gao, and Qionghai Dai. 2021. Heterogeneous hypergraph variational autoencoder for link prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 8 (2021), 4125–4138.

[15] Ziwei Fan, Zhiwei Liu, Shelby Heinecke, Jianguo Zhang, Huan Wang, Caiming Xiong, and Philip S Yu. 2023. Zero-shot Item-based Recommendation via Multi-task Product Knowledge Graph Pre-Training. *arXiv preprint arXiv:2305.07633* (2023).

[16] Yue Gao, Yifan Feng, Shuyi Ji, and Rongrong Ji. 2023. HGNN+: General Hypergraph Neural Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 3 (2023), 3181–3199. https://doi.org/10.1109/TPAMI.2022.3182052

[17] Yuxia Geng, Jiaoyan Chen, Wen Zhang, Yajing Xu, Zhuo Chen, Jeff Z. Pan, Yufeng Huang, Feiyu Xiong, and Huajun Chen. 2022. Disentangled ontology embedding for zero-shot learning. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 443–453.

[18] Yuxia Geng, Jiaoyan Chen, Xiang Zhuang, Zhuo Chen, Jeff Z Pan, Juan Li, Zonggang Yuan, and Huajun Chen. 2023. Benchmarking knowledge-driven zero-shot learning. *Journal of Web Semantics* 75 (2023), 100757.

[19] Rayid Ghani, Katharina Probst, Yan Liu, Marko Krema, and Andrew Fano. 2006. Text Mining for Product Attribute Extraction. *SIGKDD Explor. Newsl.* 8, 1 (jun 2006), 41–48. https://doi.org/10.1145/1147234.1147241

[20] Pushpendu Ghosh, Nancy Wang, and Promod Yenigalla. 2023. D-Extract: Extracting Dimensional Attributes From Product Images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 3641–3649.

[21] Jiaying Gong, Wei-Te Chen, and Hoda Eldardiry. 2023. Knowledge-Enhanced Multi-Label Few-Shot Product Attribute-Value Extraction. *arXiv preprint arXiv:2308.08413* (2023).

[22] Jiaying Gong and Hoda Eldardiry. 2021. Zero-Shot Relation Classification from Side Information. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management* (Virtual Event, Queensland, Australia) *(CIKM '21)*. Association for Computing Machinery, New York, NY, USA, 576–585. https://doi.org/10.1145/3459637.3482403

[23] Jiaying Gong and Hoda Eldardiry. 2023. Prompt-based Zero-shot Relation Extraction with Semantic Knowledge Augmentation. arXiv:2112.04539 [cs.CL]

[24] Dagmar Gromann, Yuxia Geng, Jiaoyan Chen, Zhiquan Ye, Zonggang Yuan, Wei Zhang, and Huajun Chen. 2021. Explainable Zero-Shot Learning via Attentive Graph Convolutional Network and Knowledge Graphs. *Semant. Web* 12, 5 (jan 2021), 741–765. https://doi.org/10.3233/SW-210435

[25] Pietro Hiram Guzzi and Marinka Zitnik. 2022. Editorial deep learning and graph embeddings for network biology. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 19, 2 (2022), 653–654.

[26] Zhongxuan Han, Xiaolin Zheng, Chaochao Chen, Wenjie Cheng, and Yang Yao. 2023. Intra and Inter Domain HyperGraph Convolutional Network for Cross-Domain Recommendation. In *Proceedings of the ACM Web Conference 2023*. 449–459.

[27] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *Proceedings of the web conference 2020*. 2704–2710.

[28] Yen-Hao Huang, Yi-Hsin Chen, and Yi-Shin Chen. 2022. ConTextING: Granting Document-Wise Contextual Embeddings to Graph Neural Networks for Inductive Text Classification. In *Proceedings of the 29th International Conference on Computational Linguistics*. 1163–1168.

[29] Mayank Jain, Sourangshu Bhattacharya, Harshit Jain, Karimulla Shaik, and Muthusamy Chelliah. 2021. Learning Cross-Task Attribute - Attribute Similarity for Multi-task Attribute-Value Extraction. In *Proceedings of the 4th Workshop on e-Commerce and NLP*. Association for Computational Linguistics, Online, 79–87. https://doi.org/10.18653/v1/2021.ecnlp-1.10

[30] Bilal Khan, Jia Wu, Jian Yang, and Xiaoxiao Ma. 2023. Heterogeneous Hypergraph Neural Network for Social Recommendation using Attention Network. *ACM Transactions on Recommender Systems* (2023).

[31] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[32] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461* (2019).

[33] Mengran Li, Yong Zhang, Xiaoyong Li, Yuchen Zhang, and Baocai Yin. 2023. Hypergraph transformer neural networks. *ACM Transactions on Knowledge Discovery from Data* 17, 5 (2023), 1–22.

[34] Yongkang Li, Zipei Fan, Du Yin, Renhe Jiang, Jinliang Deng, and Xuan Song. 2022. HMGCL: Heterogeneous multigraph contrastive learning for LBSN friend recommendation. *World Wide Web* (2022), 1–24.

[35] Yongkang Li, Zipei Fan, Jixiao Zhang, Dengheng Shi, Tianqi Xu, Du Yin, Jinliang Deng, and Xuan Song. 2022. Heterogeneous Hypergraph Neural Network for Friend Recommendation with Human Mobility. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 4209–4213.

[36] Rongmei Lin, Xiang He, Jie Feng, Nasser Zalmout, Yan Liang, Li Xiong, and Xin Luna Dong. 2021. PAM: understanding product images in cross product category attribute extraction. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 3262–3270.

[37] Fangchao Liu, Hongyu Lin, Xianpei Han, Boxi Cao, and Le Sun. 2022. Pre-training to Match for Unified Low-shot Relation Extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Dublin, Ireland, 5785–5795. https://doi.org/10.18653/v1/2022.acl-long.397

[38] Jiongnan Liu, Zhicheng Dou, Guoyu Tang, and Sulong Xu. 2023. JDsearch: A Personalized Product Search Dataset with Real Queries and Full Interactions. In *Proceedings of the SIGIR 2023*. ACM. https://doi.org/10.1145/3539618.3591900

[39] Jingquan Liu, Xiaoyong Du, Yuanzhe Li, and Weidong Hu. 2022. Hypergraph Variational Autoencoder for Multimodal Semi-supervised Representation Learning. In *Artificial Neural Networks and Machine Learning–ICANN 2022: 31st International Conference on Artificial Neural Networks, Bristol, UK, September 6–9, 2022, Proceedings; Part IV*. Springer, 395–406.

[40] Jie Liu, Lingyun Song, Guangtao Wang, and Xuequn Shang. 2023. Meta-HGT: Metapath-aware HyperGraph Transformer for heterogeneous information network embedding. *Neural Networks* 157 (2023), 65–76.

[41] Mengyin Liu, Chao Zhu, Hongyu Gao, Weibo Gu, Hongfa Wang, Wei Liu, and Xu cheng Yin. 2023. Boosting Multi-Modal E-commerce Attribute Value Extraction via Unified Learning Scheme and Dynamic Range Minimization.

arXiv:2207.07278 [cs.CV]
[42] Shilei Liu, Lin Li, Jun Song, Yonghua Yang, and Xiaoyi Zeng. 2023. Multimodal Pre-Training with Self-Distillation for Product Understanding in E-Commerce. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*. 1039–1047.
[43] Yuqing Ma, Shihao Bai, Shan An, Wei Liu, Aishan Liu, Xiantong Zhen, and Xianglong Liu. 2020. Transductive Relation-Propagation Network for Few-shot Learning.. In *IJCAI*, Vol. 20. 804–810.
[44] Marianna Milano, Giuseppe Agapito, and Mario Cannataro. 2022. Challenges and limitations of biological network analysis. *BioTech* 11, 3 (2022), 24.
[45] Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).
[46] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*. 188–197.
[47] Farhad Pourpanah, Moloud Abdar, Yuxuan Luo, Xinlei Zhou, Ran Wang, Chee Peng Lim, Xi-Zhao Wang, and Q. M. Jonathan Wu. 2023. A Review of Generalized Zero-Shot Learning Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 4 (2023), 4051–4070. https://doi.org/10.1109/TPAMI.2022.3191696
[48] Duangmanee (Pew) Putthividhya and Junling Hu. 2011. Bootstrapped Named Entity Recognition for Product Attribute Extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (Edinburgh, United Kingdom) *(EMNLP '11)*. Association for Computational Linguistics, USA, 1557–1567.
[49] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.
[50] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research* 21, 1 (2020), 5485–5551.
[51] Rahul Ragesh, Sundararajan Sellamanickam, Arun Iyer, Ramakrishna Bairi, and Vijay Lingam. 2021. Hetegcn: heterogeneous graph convolutional networks for text classification. In *Proceedings of the 14th ACM international conference on web search and data mining*. 860–868.
[52] Martin Rezk, Laura Alonso Alemany, Lasguido Nio, and Ted Zhang. 2019. Accurate Product Attribute Extraction on the Field. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. 1862–1873. https://doi.org/10.1109/ICDE.2019.00202
[53] Kalyani Roy, Pawan Goyal, and Manish Pandey. 2021. Attribute value generation from product title using language models. In *Proceedings of The 4th Workshop on e-Commerce and NLP*. 13–17.
[54] Kalyani Roy, Tapas Nayak, and Pawan Goyal. 2022. Exploring Generative Models for Joint Attribute Value Extraction from Product Titles. *arXiv preprint arXiv:2208.07130* (2022).
[55] Oscar Sainz, Oier Lopez de Lacalle, Gorka Labaka, Ander Barrena, and Eneko Agirre. 2021. Label Verbalization and Entailment for Effective Zero and Few-Shot Relation Extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 1199–1212. https://doi.org/10.18653/v1/2021.emnlp-main.92
[56] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE transactions on neural networks* 20, 1 (2008), 61–80.
[57] Keiji Shinzato, Naoki Yoshinaga, Yandi Xia, and Wei-Te Chen. 2022. Simple and Effective Knowledge-Driven Query Expansion for QA-Based Product Attribute Extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 227–234.
[58] Keiji Shinzato, Naoki Yoshinaga, Yandi Xia, and Wei-Te Chen. 2023. A Unified Generative Approach to Product Attribute-Value Identification. *arXiv preprint arXiv:2306.05605* (2023).
[59] Xiangguo Sun, Hongzhi Yin, Bo Liu, Hongxu Chen, Jiuxin Cao, Yingxia Shao, and Nguyen Quoc Viet Hung. 2021. Heterogeneous hypergraph embedding for graph classification. In *Proceedings of the 14th ACM international conference on web search and data mining*. 725–733.
[60] Kai Wang, Jianzhi Shao, Tao Zhang, Qijin Chen, and Chengfu Huo. 2023. MP-KGAC: Multimodal Product Attribute Completion in E-commerce. In *Companion Proceedings of the ACM Web Conference 2023*. 336–340.
[61] Qifan Wang, Li Yang, Bhargav Kanagal, Sumit Sanghai, D Sivakumar, Bin Shu, Zac Yu, and Jon Elsas. 2020. Learning to extract attribute value from product via question answering: A multi-task approach. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 47–55.
[62] Qifan Wang, Li Yang, Jingang Wang, Jitin Krishnan, Bo Dai, Sinong Wang, Zenglin Xu, Madian Khabsa, and Hao Ma. 2022. SMARTAVE: Structured Multimodal Transformer for Product Attribute Value Extraction. In *Findings of the Association for Computational Linguistics: EMNLP 2022*. 263–276.
[63] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *The world wide web conference*. 2022–2032.
[64] Xuemei Wei, Yezheng Liu, Jianshan Sun, Yuanchun Jiang, Qifeng Tang, and Kun Yuan. 2023. Dual subgraph-based graph neural network for friendship prediction in location-based social networks. *ACM Transactions on Knowledge Discovery from Data* 17, 3 (2023), 1–28.
[65] Zheng Wenping, Liu Meilin, and Liang Jiye. 2022. Hypergraphs: Concepts, Applications and Analysis. In *2022 IEEE 13th International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)*. 1–6. https://doi.org/10.1109/PAAP56126.2022.10010428
[66] Yuk Wah Wong, Dominic Widdows, Tom Lokovic, and Kamal Nigam. 2009. Scalable Attribute-Value Extraction from Semi-Structured Text. In *ICDM Workshop on Large-scale Data Mining: Theory and Applications*. http://www.computer.org/portal/web/csdl/doi/10.1109/ICDMW.2009.81
[67] Hanrui Wu, Yuguang Yan, and Michael Kwok-Po Ng. 2022. Hypergraph collaborative network on vertices and hyperedges. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 3 (2022), 3245–3258.
[68] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2020. Inductive representation learning on temporal graphs. *arXiv preprint arXiv:2002.07962* (2020).
[69] Huimin Xu, Wenting Wang, Xinnian Mao, Xinyu Jiang, and Man Lan. 2019. Scaling up open tagging from tens to thousands: Comprehension empowered attribute value extraction from product title. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 5214–5223.
[70] Liyan Xu, Chenwei Zhang, Xian Li, Jingbo Shang, and Jinho D Choi. 2023. Towards Open-World Product Attribute Mining: A Lightly-Supervised Approach. *arXiv preprint arXiv:2305.18350* (2023).
[71] Zixuan Xu, Penghui Wei, Shaoguo Liu, Weimin Zhang, Liang Wang, and Bo Zheng. 2023. Correlative Preference Transfer with Hierarchical Hypergraph Network for Multi-Domain Recommendation. In *Proceedings of the ACM Web Conference 2023*. 983–991.
[72] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. 2019. Hypergcn: A new method for training graph convolutional networks on hypergraphs. *Advances in neural information processing systems* 32 (2019).
[73] Jun Yan, Nasser Zalmout, Yan Liang, Christan Grant, Xiang Ren, and Xin Luna Dong. 2021. AdaTag: Multi-Attribute Value Extraction from Product Profiles with Adaptive Decoding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Online, 4694–4705. https://doi.org/10.18653/v1/2021.acl-long.362
[74] Li Yang, Qifan Wang, Zac Yu, Anand Kulkarni, Sumit Sanghai, Bin Shu, Jon Elsas, and Bhargav Kanagal. 2022. MAVE: A product dataset for multi-source attribute value extraction. In *Proceedings of the fifteenth ACM international conference on web search and data mining*. 1256–1265.
[75] Jiying Zhang, Yuzhao Chen, Xi Xiao, Runiu Lu, and Shu-Tao Xia. 2022. Learnable hypergraph laplacian for hypergraph learning. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 4503–4507.
[76] Liyan Zhang, Jingfeng Guo, Jiazheng Wang, Jing Wang, Shanshan Li, and Chunying Zhang. 2022. Hypergraph and uncertain hypergraph representation learning theory and methods. *Mathematics* 10, 11 (2022), 1921.
[77] Xinyang Zhang, Chenwei Zhang, Xian Li, Xin Luna Dong, Jingbo Shang, Christos Faloutsos, and Jiawei Han. 2022. OA-Mine: Open-World Attribute Mining for E-Commerce Products with Weak Supervision. In *Proceedings of the ACM Web Conference 2022* (Virtual Event, Lyon, France) *(WWW '22)*. Association for Computing Machinery, New York, NY, USA, 3153–3161. https://doi.org/10.1145/3485447.3512035
[78] Yupeng Zhang, Shensi Wang, Peiguang Li, Guanting Dong, Sirui Wang, Yunsen Xian, Zhoujun Li, and Hongzhi Zhang. 2023. Pay attention to implicit attribute values: a multi-modal generative framework for AVE task. In *Findings of the Association for Computational Linguistics: ACL 2023*. 13139–13151.
[79] Guineng Zheng, Subhabrata Mukherjee, Xin Luna Dong, and Feifei Li. 2018. OpenTag: Open Attribute Value Extraction from Product Profiles. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (London, United Kingdom) *(KDD '18)*. Association for Computing Machinery, New York, NY, USA, 1049–1058. https://doi.org/10.1145/3219819.3219839
[80] Tiangang Zhu, Yue Wang, Haoran Li, Youzheng Wu, Xiaodong He, and Bowen Zhou. 2020. Multimodal Joint Attribute Prediction and Value Extraction for E-commerce Product. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 2129–2139. https://doi.org/10.18653/v1/2020.emnlp-main.166

Table 5: Example of zero-shot dataset statistics in training, validation and testing sets, respectively.

Category	Training			Validation			Testing			All
	#P	#A	#PA	#P	#A	#PA	#P	#A	#PA	Ave #A/P
Arts	10,250	1,796	8,400	3	6	6	15	23	30	2.48
Books	9,310	158	5,210	4	3	8	413	54	852	1.44
Cellphones	6,772	1,149	5,187	91	109	192	157	175	332	2.38
Giftcards	84	8	74	8	2	16	11	3	9	2.37
Grocery	15,834	3,945	13,933	8	16	16	18	33	36	2.56
Industrial	2,644	1,264	2,381	16	27	33	8	14	17	2.76
Pet	12,878	2,193	13,187	24	42	48	73	117	150	3.16
Software	187	87	152	2	4	4	8	14	16	2.11
Tools	30,236	6,210	29,759	14	24	28	58	97	120	2.92
Videogames	559	240	477	35	45	75	57	67	128	2.86

Table 6: Model Training Time in One Epoch (second).

Model	Giftcards	Software	Videogames	Industrial	Cellphones	Arts	Pet	Books	Grocery	Tools
BERT-MLC	2.37	15.48	42.12	291.60	578.78	797.11	1004.53	1073.65	1266.68	2391.12
BART	3.66	24.48	66.60	304.56	873.36	1152.00	1292.95	1604.52	1910.52	3521.88
$T5_{small}$	2.21	19.14	58.23	256.70	698.10	967.87	1209.27	1355.23	1576.67	2890.03
GNN	0.09	0.19	1.00	5.46	16.46	30.02	57.50	12.80	73.33	150.91
HGNN	0.72	1.60	6.28	27.59	64.24	122.06	209.28	94.52	235.20	504.61
HyperPAVE	0.90	1.66	6.71	30.06	70.18	133.43	224.40	89.22	251.14	543.07

6 APPENDIX

6.1 Dataset

Table 5 reports an example of dataset statistics in training, validation, and testing sets, where $\#P$, $\#A$, and $\#PA$ denotes the number of product nodes, the number of aspect nodes and the number of product to aspect edges, respectively. The last column Ave $\#A/p$ indicates the average number of attribute value pairs for each product. Because training, validation, and testing sets for the multi-label zero-shot setting are randomly generated for each run of the experiment, there exist different dataset statistics.

6.2 Parameter Settings

We randomly select unseen attribute value pairs with unseen products following the sampling rule in Sec. 3.2. For the hyperparameter and configuration of HyperPAVE, we implement HyperPAVE in PyTorch and optimize it with AdamW optimizer. We train HyperPAVE and all baselines on the training set and we use a validation set to select the optimal hyper-parameter settings, and finally report the performance on the test set. We follow the early stopping strategy when selecting the model for testing. For all methods, we run 10 times with different random seeds and report the average results with standard deviation. Our proposed model HyperPAVE achieves its best performance with the following setup. The nodes' features are initialized by a BERT encoder with a 768-dimension size. The max length for category, product, and attribute values are 32, 512, and 32, respectively. The initial learning rate is selected via grid search within the range of $\{5e-1, 5e-3, 5e-4, 5e-5\}$ with 1e-6 weight decay for minimizing the loss. The hidden sizes for convolution layers are 768 in both HyperConv and GraphConv. The activation function is ReLU. The dropout rate is 0.5 and the batch size is 4. We set the number of neighbors to 20 and the negative sampling rate is 2.0. For the fusion module, the weights of the product node embeddings from hyperedges of 'also buy', 'also view', 'products with all aspects', and 'category with all products and aspects' are dynamically changed for different categories. Experiments are conducted in Sec. 4.2.4 to explore the weights in these fusion modules.

6.3 Experiments

Due to limited space in the main context, we only demonstrate ablation study over three categories (Books, Giftcards, and Pets) in Table 3. Here in Table 7, we report the ablation study over the other seven categories on MAVE. We also demonstrate the model training time for one epoch across the ten categories on MAVE in Table 6. All models use the same input max_length as 512 and batch size as 4. For different graph-based models, they show similar efficiency performance. Thus, we only demonstrate two representative graph-based models (GNN and HGNN) for training efficiency comparison.

Table 7: Ablation study over HyperPAVE components in the zero-shot setting across seven categories on MAVE dataset.

	F1	mAP	AUC	MRR	NDCG	Hit@5	Hits@10	Hits@100
				Arts				
nodeID	1.35 ± 0.29	10.73 ± 0.52	92.03 ± 0.12	0.86 ± 0.03	21.92 ± 0.58	15.00 ± 1.23	28.33 ± 1.10	61.67 ± 0.64
BERT	8.23 ± 0.24	26.30 ± 0.14	92.69 ± 0.06	11.48 ± 0.19	40.27 ± 0.15	35.43 ± 0.27	52.86 ± 0.14	82.43 ± 0.10
BERT (Fine-tuned)	19.87 ± 0.15	34.77 ± 0.39	**99.84 ± 0.04**	13.16 ± 0.17	53.84 ± 0.25	75.00 ± 0.00	75.00 ± 0.00	100.00 ± 0.00
Hyper (Product)	30.93 ± 0.26	42.33 ± 0.27	99.06 ± 0.01	22.95 ± 0.27	57.84 ± 0.24	57.50 ± 0.31	75.00 ± 0.03	93.75 ± 0.04
Hyper (Behavior)	37.03 ± 0.67	42.39 ± 0.62	98.35 ± 0.02	28.51 ± 0.50	60.47 ± 0.46	56.67 ± 0.29	**83.33 ± 0.24**	100.00 ± 0.00
HyperPAVE	**43.33 ± 0.22**	**40.99 ± 0.18**	99.22 ± 0.01	**47.52 ± 0.30**	**64.87 ± 0.39**	**75.00 ± 0.00**	82.50 ± 0.12	**100.00 ± 0.00**
				Cellphones				
nodeID	19.75 ± 0.67	22.88 ± 0.20	97.72 ± 0.01	10.65 ± 0.45	38.33 ± 0.23	38.33 ± 0.50	57.22 ± 0.15	80.00 ± 0.11
BERT	24.21 ± 0.15	26.15 ± 0.16	97.60 ± 0.02	17.02 ± 0.47	40.97 ± 0.24	41.11 ± 0.05	70.05 ± 0.30	86.11 ± 0.36
BERT (Fine-tuned)	22.77 ± 0.16	26.80 ± 0.31	98.09 ± 0.04	16.50 ± 0.43	43.03 ± 0.37	50.00 ± 0.00	75.00 ± 0.00	92.50 ± 0.12
Hyper (Product)	32.27 ± 0.28	33.32 ± 0.09	98.94 ± 0.04	22.26 ± 0.30	**54.17 ± 0.25**	70.25 ± 0.27	**90.00 ± 0.00**	100.00 ± 0.00
Hyper (Behavior)	28.32 ± 0.38	33.88 ± 0.22	**99.63 ± 0.01**	22.57 ± 0.10	47.81 ± 0.26	52.50 ± 0.14	61.67 ± 0.04	97.50 ± 0.08
HyperPAVE	**39.91 ± 0.16**	**35.81 ± 0.18**	99.22 ± 0.02	**23.54 ± 0.20**	52.88 ± 0.18	**72.50 ± 0.08**	75.00 ± 0.00	**100.00 ± 0.00**
				Grocery				
nodeID	6.50 ± 0.49	23.31 ± 0.27	95.48 ± 0.04	15.33 ± 0.19	21.98 ± 0.38	22.50 ± 0.28	35.00 ± 0.31	65.00 ± 0.27
BERT	14.65 ± 0.40	22.85 ± 0.34	96.18 ± 0.08	15.80 ± 0.33	22.55 ± 0.42	30.10 ± 0.31	35.10 ± 0.17	75.00 ± 0.51
BERT (Fine-tuned)	19.42 ± 0.46	25.84 ± 0.18	99.20 ± 0.01	17.78 ± 0.20	27.93 ± 0.29	25.00 ± 0.10	35.50 ± 0.13	**87.50 ± 0.13**
Hyper (Product)	22.41 ± 0.62	32.41 ± 0.37	99.48 ± 0.02	18.82 ± 0.28	35.64 ± 0.40	33.33 ± 0.71	35.50 ± 0.21	66.67 ± 0.10
Hyper (Behavior)	29.20 ± 0.29	32.85 ± 0.49	98.34 ± 0.04	14.41 ± 0.16	37.66 ± 0.37	35.05 ± 0.16	50.00 ± 0.00	70.00 ± 0.11
HyperPAVE	**33.43 ± 0.28**	**42.71 ± 0.30**	**99.56 ± 0.00**	**22.52 ± 0.38**	**52.64 ± 0.36**	**50.00 ± 0.00**	**50.00 ± 0.00**	75.50 ± 0.50
				Industrial				
nodeID	10.40 ± 0.38	16.44 ± 0.22	93.16 ± 0.05	2.59 ± 0.17	30.07 ± 0.24	28.75 ± 0.49	35.00 ± 0.35	68.75 ± 0.27
BERT	1.48 ± 0.24	5.37 ± 0.16	89.75 ± 0.11	0.66 ± 0.10	13.58 ± 0.32	8.13 ± 0.31	11.87 ± 0.54	55.63 ± 0.71
BERT (Fine-tuned)	14.06 ± 0.11	18.82 ± 0.50	99.05 ± 0.01	4.99 ± 0.14	41.11 ± 0.50	25.00 ± 0.00	50.00 ± 0.04	100.00 ± 0.00
Hyper (Product)	19.78 ± 0.19	14.15 ± 0.17	94.34 ± 0.08	7.63 ± 0.16	26.68 ± 0.16	24.73 ± 0.29	37.50 ± 0.20	75.00 ± 0.40
Hyper (Behavior)	15.70 ± 0.31	31.42 ± 0.30	96.57 ± 0.04	7.19 ± 0.33	45.26 ± 0.22	41.25 ± 0.31	55.00 ± 0.35	87.50 ± 0.00
HyperPAVE	**27.70 ± 0.10**	**33.29 ± 0.17**	**99.71 ± 0.01**	**16.10 ± 0.08**	**54.08 ± 0.26**	**52.50 ± 0.18**	**80.00 ± 0.16**	**100.00 ± 0.00**
				Software				
nodeID	1.97 ± 0.22	18.11 ± 0.25	76.27 ± 0.18	4.39 ± 0.32	30.12 ± 0.53	23.75 ± 0.58	62.50 ± 0.05	100.00 ± 0.00
BERT	7.38 ± 0.14	14.10 ± 0.31	74.89 ± 0.14	6.38 ± 0.29	34.19 ± 0.44	26.70 ± 0.20	36.25 ± 0.11	100.00 ± 0.00
BERT (Fine-tuned)	11.78 ± 0.31	15.29 ± 0.50	76.70 ± 0.03	6.75 ± 0.46	36.52 ± 0.45	23.75 ± 0.23	37.50 ± 0.16	100.00 ± 0.00
Hyper (Product)	35.88 ± 0.37	40.72 ± 0.16	**84.40 ± 0.10**	21.25 ± 0.46	59.51 ± 0.18	46.25 ± 0.26	**63.75 ± 0.40**	100.00 ± 0.00
Hyper (Behavior)	12.22 ± 0.36	36.33 ± 0.48	81.25 ± 0.10	6.09 ± 0.27	34.19 ± 0.20	25.00 ± 0.31	38.75 ± 0.51	100.00 ± 0.00
HyperPAVE	**47.62 ± 0.21**	**51.64 ± 0.10**	77.80 ± 0.12	**26.66 ± 0.15**	**63.48 ± 0.10**	**61.25 ± 0.40**	62.50 ± 0.25	**100.00 ± 0.00**
				Tools				
nodeID	8.90 ± 0.37	17.00 ± 0.24	97.91 ± 0.10	2.36 ± 0.19	22.27 ± 0.37	50.00 ± 0.02	50.00 ± 0.00	50.00 ± 0.00
BERT	14.53 ± 0.17	18.51 ± 0.25	96.21 ± 0.09	6.51 ± 0.18	21.30 ± 0.48	48.50 ± 0.15	52.05 ± 0.33	80.00 ± 0.20
BERT (Fine-tuned)	21.33 ± 0.14	23.85 ± 0.36	99.19 ± 0.05	6.81 ± 0.31	26.88 ± 0.32	49.15 ± 0.26	55.70 ± 0.39	**87.07 ± 0.25**
Hyper (Product)	32.86 ± 0.24	29.20 ± 0.47	98.27 ± 0.06	12.26 ± 0.14	43.96 ± 0.26	49.53 ± 0.35	65.00 ± 0.24	83.87 ± 0.17
Hyper (Behavior)	31.43 ± 0.27	25.13 ± 0.25	**99.30 ± 0.07**	11.51 ± 0.17	28.11 ± 0.23	50.06 ± 0.25	58.20 ± 0.34	86.40 ± 0.18
HyperPAVE	**34.00 ± 0.28**	**47.83 ± 0.29**	98.00 ± 0.06	**12.93 ± 0.18**	**59.05 ± 0.27**	**52.00 ± 0.34**	**65.37 ± 0.25**	84.72 ± 0.20
				Videogames				
nodeID	3.25 ± 0.47	7.31 ± 0.38	79.00 ± 0.58	1.49 ± 0.22	17.27 ± 0.19	10.00 ± 1.21	20.00 ± 1.26	70.00 ± 0.62
BERT	6.67 ± 0.41	10.25 ± 0.27	85.83 ± 0.21	3.01 ± 0.52	**33.30 ± 0.35**	30.05 ± 0.26	43.50 ± 0.35	100.00 ± 0.00
BERT (Fine-tuned)	12.87 ± 0.21	11.44 ± 0.17	76.84 ± 0.15	4.21 ± 0.41	25.26 ± 0.29	15.71 ± 0.54	37.86 ± 0.30	73.70 ± 0.26
Hyper (Product)	20.00 ± 0.23	16.45 ± 0.19	91.51 ± 0.20	8.76 ± 0.39	28.61 ± 0.25	45.00 ± 0.16	50.00 ± 0.15	**100.00 ± 0.00**
Hyper (Behavior)	16.83 ± 0.26	12.38 ± 0.11	86.73 ± 0.36	7.33 ± 0.16	27.28 ± 0.17	15.00 ± 0.55	40.71 ± 0.38	80.71 ± 0.35
HyperPAVE	**25.31 ± 0.19**	**21.19 ± 0.17**	84.32 ± 0.05	**9.31 ± 0.30**	23.99 ± 0.16	**50.00 ± 0.50**	**50.00 ± 0.50**	85.71 ± 0.12