A Unified Generative Approach to Product Attribute-Value Identification

Keiji Shinzato  
Rakuten Institute of Technology,  
Rakuten Group, Inc.  
keiji.shinzato@rakuten.com

Naoki Yoshinaga  
Institute of Industrial Science,  
The University of Tokyo  
ynaga@iis.u-tokyo.ac.jp

Yandi Xia  
Rakuten Institute of Technology,  
Rakuten Group, Inc.  
yandi.xia@rakuten.com

Wei-Te Chen  
Rakuten Institute of Technology,  
Rakuten Group, Inc.  
weite.chen@rakuten.com

Abstract

Product attribute-value identification (PAVI) has been studied to link products on e-commerce sites with their attribute values (e.g., {⟨Material, Cotton⟩}) using product text as clues. Technical demands from real-world e-commerce platforms require PAVI methods to handle unseen values, multi-attribute values, and canonicalized values, which are only partly addressed in existing extraction- and classification-based approaches. Motivated by this, we explore a generative approach to the PAVI task. We finetune a pre-trained generative model, T5, to decode a set of attribute-value pairs as a target sequence from the given product text. Since the attribute-value pairs are unordered set elements, how to linearize them will matter; we, thus, explore methods of composing an attribute-value pair and ordering the pairs for the task. Experimental results confirm that our generation-based approach outperforms the existing extraction- and classification-based methods on large-scale real-world datasets meant for those methods.

1 Introduction

Since organized product data play a crucial role in serving better product search and recommendation to customers, product attribute value identification (PAVI) has been a core task in the e-commerce industry. For attributes pre-defined by e-commerce sites, the task aims to link values of those attributes to products using product titles and descriptions as clues (Figure 1). For example, from the title “D&G Cotton piqué polo shirt Designed and manufactured in Italy,” models are required to return a set of possible attribute-value pairs, namely {⟨Brand, Dolce & Gabbana⟩, ⟨Material, Cotton⟩, ⟨Country of origin, Italy⟩, ⟨Country of design, Italy⟩}.

In the literature, PAVI has been addressed basically by extraction from the product text by using named entity recognition (Probst et al., 2007; Wong et al., 2008; Putthividhya and Hu, 2011; Bing et al., 2012; Shinzato and Sekine, 2013; More, 2016; Zheng et al., 2018; Rezk et al., 2019; Karamanolakis et al., 2020; Zhang et al., 2020) or question answering (Xu et al., 2019; Wang et al., 2020; Shinzato et al., 2022; Yang et al., 2022). However, since PAVI requires canonicalized values rather than raw value strings in the product text, some researchers have started to solve PAVI as classification (Chen et al., 2022; Fuchs and Acriche, 2022).

To adopt PAVI models in real-world e-commerce platforms, there are the following challenges.

Unseen values. Since values can be entities such as brands, models need to identify values unseen in the training data (Zheng et al., 2018). Since the classification-based approach assumes a predefined set of target classes (attribute-value pairs), it cannot handle such unseen attribute-value pairs.

Multi-attribute values. When values can be associated with multiple attributes (e.g., Italy in Figure 1), models need to identify multiple attributes for a single value string in the text. To address this, the extraction-based approach must solve nested named entity recognition (Wang et al., 2020).
Table 1: Comparison of different PAVI approaches in terms of the challenges: handling unseen values, multi-attribute values, and canonicalized values.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Unseen</th>
<th>Multi</th>
<th>Canon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extraction</td>
<td>Support</td>
<td>Partially</td>
<td>Not</td>
</tr>
<tr>
<td>Classification</td>
<td>Not</td>
<td>Support</td>
<td>Support</td>
</tr>
<tr>
<td>Generation (ours)</td>
<td>Support</td>
<td>Support</td>
<td>Support</td>
</tr>
</tbody>
</table>

**Canonicalized values.** E-commerce vendors need attribute values in the canonical form (e.g., *Dolce & Gabbana* for *D&G*) in actual services such as faceted product search (Chen et al., 2022). The extraction-based approach needs a further step to canonicalize extracted raw value strings (Putthividhya and Hu, 2011; Zhang et al., 2021).

Motivated by the shortcomings of the existing approaches to PAVI (Table 1), we propose to cast PAVI as sequence-to-set generation, which can handle all the challenges by using canonicalized attribute-value pairs for training (Figure 1). We expect that 1) generation can decode unseen values by considering corresponding values in the input, 2) generation can decode the same string in the input multiple times as values for different attributes, and 3) generation can learn how to canonicalize raw strings in input. We finetune the pre-trained generative model T5 (Raffel et al., 2020) to autoregressively decode a set of attribute-value pairs from the given text. As discussed in (Vinyals et al., 2016; Yang et al., 2018; Madaan et al., 2022), the output order will matter to decode sets as a sequence. We therefore explore methods of composing an attribute-value pair and ordering the pairs for the task.

We evaluate our generative framework on two real-world datasets, MAVE (Yang et al., 2022) and our in-house product data. The experimental results demonstrate that our generation-based approach outperforms extraction- and classification-based methods on their target datasets.

Our contribution is as follows.

- We have solved the product attribute-value identification task as a sequence-to-set generation for the first time.
- We revealed the effective order of attribute-value pairs for the T5 model among various ordering schemes (Table 2).
- We provided the first comprehensive comparison among extraction-, classification-, and generation-based models on two real-world PAVI datasets, and empirically confirmed that the generation-based models outperformed the others (Table 6) while addressing all challenges in PAVI (Tables 9, 11 and 12).

2 Related Work

**Product Attribute-Value Extraction** Traditionally, a myriad of previous studies formulated PAVI as named entity recognition (NER) (Probst et al., 2007; Wong et al., 2008; Putthividhya and Hu, 2011; Bing et al., 2012; Shinzato and Sekine, 2013; More, 2016; Zheng et al., 2018; Rezk et al., 2019; Karamanolakis et al., 2020; Zhang et al., 2020). However, since the number of attributes in real-world e-commerce sites can exceed ten thousand (Xu et al., 2019), the NER-based models suffer from the data sparseness problem, which makes the models perform poorly. While the extraction-based approach can identify unseen values in the training data, it cannot canonicalize values by itself and is difficult to handle overlapping values, although nested NER (surveyed in Wang et al. (2022)) can remedy the latter issue.

To mitigate the data sparseness problem, some studies leveraged QA models for the PAVI task (Xu et al., 2019; Wang et al., 2020; Yang et al., 2022; Shinzato et al., 2022), by assuming the target attribute for extraction as additional input. These QA-based approaches take an attribute as *query* and product text as *context*, and extract attribute values from the context as *answer* for the query. Similar to the traditional NER-based models, these extractive QA-based models do not work for canonicalized values. To improve the ability to find unseen values, Roy et al. (2021) generated a value for the given product text and attribute. However, we need to apply these QA-based models to the same context with each of thousands of attributes, unless comprehensive attribute taxonomy is designed to narrow down possible attributes; such taxonomy is not always available and is often imperfect, as investigated by Mao et al. (2020) for Amazon.com.

**Product Attribute-Value Identification as Classification** Chen et al. (2022) solved PAVI as multi-label classification (MLC), assuming attribute-value pairs as target labels. One of the problems in this approach is that the distribution between positive and negative labels is heavily skewed because the number of possible attribute values per product is much smaller than the total number of attribute values. To alleviate the imbalanced label prob-
problem, they introduced a method called label masking to reduce the number of negative labels using an attribute taxonomy designed by the e-commerce platform. To mitigate the extreme multi-class classification, Fuchs and Acriche (2022) decomposed the target label, namely attribute-value pair, into two atomic labels, attribute and value, to perform a hierarchical classification. Although these classification-based approaches support canonicalized values and multi-attribute values, they cannot handle unseen values.

In this study, we adopt a generative approach to return a set of attribute-value pairs from given product data, and empirically compare it with the above two approaches. Our approach can be applied to the task settings adopted by the QA-based models, by simply feeding one (or more) target attributes as additional input (e.g., title [SEP] description [SEP] attributes) to decode their values in order.

3 Proposed Method

As mentioned above, previous studies formalize PAWI as either sequence tagging or multi-label classification problems. These approaches do not address all the challenges derived from real-world e-commerce sites at the same time (Table 1).

We thus propose a unified generative framework that formalizes PAWI as a sequence-to-set problem. Let us denote \( x = \{x_1, x_2, \ldots, x_n\} \) as product data (title and description) where \( n \) is the number of tokens in \( x \). Given product data \( x \), the model is trained to return a set of attribute-value pairs \( y = \{\langle a_1, v_1 \rangle, \langle a_2, v_2 \rangle, \ldots, \langle a_k, v_k \rangle\} \) for \( x \), where \( k \) is the number of attribute-value pairs associated with the product; \( a_i = \{a_1, a_2, \ldots, a_{m_i}\} \) and \( v_i = \{v_1, v_2, \ldots, v_{l_i}\} \) are corresponding attribute and value.\(^1\) \( m_i \) and \( l_i \) are the numbers of tokens in \( a_i \) and \( v_i \), respectively.

As the backbone of our approach, we employ T5 (Raffel et al., 2020), a pre-trained generative model based on Transformer (Vaswani et al., 2017) that maps an input sequence to an output sequence.

The key issue in formulating the PAWI task as sequence-to-sequence generation is how to linearize a set of attribute-value pairs into a sequence. Firstly, we should consider how to associate attributes and their corresponding values in the output sequence. Secondly, the autoregressive generation decodes output tokens (here, attributes and values) one by one conditioned on the previous labels. Thus, if specific (or informative) tokens are first decoded, it will make it easy to decode the remaining tokens. However, due to the exposure bias, decoding specific (namely, infrequent) tokens are more likely to fail. To address the challenge, we decompose the issue on linearization into two subproblems on how to compose an attribute-value pair and how to order attribute-value pairs. In what follows, we will describe these subproblems.

3.1 Composition of Attribute-Value Pair

We consider the following ways to compose an attribute-value pair.\(^2\) In both ways, attributes and values are separated by a special token [SEP].

Attribute-then-value, \( \langle A, V \rangle \) Attribute is placed, and then its value (e.g., Color [SEP] White). In general, the vocabulary size of attributes is much smaller than that of values. Thus, models will be easier to decode attributes than values.

Value-then-attribute, \( \langle V, A \rangle \) Value is placed, and then its attribute (e.g., White [SEP] Color). This will be effective when the target values appear as raw strings in the given text and are easier to decode than attributes.

3.2 Ordering of Attribute-Value Pairs

In this work, we design three different types of the attribute-value pair ordering (Table 2). We use a special token [SEP] as a separator between pairs.

---

\(^1\)When there are more than one value for the same attribute (e.g., size), we decompose a pair of attribute and \( n (> 1) \) multiple values into \( n \) attribute-value pairs (Table 4).

\(^2\)We have also attempted to generate all attributes prior to values (namely, \( a_1 [SEP] \ldots a_k [SEP] v_1 [SEP] \ldots v_k \)) or vice versa; this unpaired generation slightly underperformed the paired generation used here.

<table>
<thead>
<tr>
<th>Ordering</th>
<th>Attribute-value pairs placed in the target sequence</th>
</tr>
</thead>
</table>

Table 2: Example of attribute-value pair ordering with the attribute-then-value composition. We assume that the frequency of the pairs is \( \langle \text{Color, White} \rangle > \langle \text{Color, Red} \rangle > \langle \text{Material, Nylon} \rangle \).
Rare-first Specific attribute values (e.g., brands) can help models decode other attribute values. For example, since Levi’s has many products made of denim, it is easy to decode the material if Levi’s is decoded in advance. Meanwhile, since there are many brands that have products made of denim, decoding denim as a material in advance is useless to decode the brands. To capture this inter-value dependency, we assume a correlation between the frequency and specificity of attribute-value pairs, and place attribute-value pairs to the target sequence in rare-first ordering of attribute-value pair frequency calculated from the training data. The attribute-value pairs with the same ranking will be placed randomly for this and following ordering.

Common-first When the model autoregressively decodes outputs, intermediate errors affect future decoding. Thus, it is important to decode from confident attribute-value pairs. Since models will be easier to decode attribute-value pairs that have more training examples, we place attribute-value pairs to the target sequence in the common-first ordering of attribute-value pair frequency. This approach is adopted by Yang et al. (2018) in solving multi-label document classification as generation.

Random To see whether the orders matter, we randomly sort attribute-value pairs in the target sequence; more precisely, we collect, uniquely, and shuffle attribute-value pairs taken from all training examples, and sort the pairs in each example according to the obtained order of the pairs. If this random ordering shows inferior performance against the above orderings, we can conclude output orders matter in this task.

4 Experiments

We evaluate our generative approach to PAVI using two real-world datasets. In the literature, different types of approaches are rarely compared due to the proprietary nature of codes and datasets in this task. We thus compare our generation-based model with extraction- and classification-based models, all of which are based on public pre-trained models, using not only in-house but also public datasets.

4.1 Datasets

We used MAVE (Yang et al., 2022)\(^3\) and our in-house product data for experiments. The MAVE dataset is designed to evaluate the extraction-based PAVI models, while the in-house dataset is designed to evaluate classification-based models (Table 3).

MAVE dataset compiles the product data taken from Amazon Review Data (Ni et al., 2019). The dataset contains various kinds of products such as shoes, clothing, watches, books, and home decor decals. Each example consists of product titles and descriptions, attribute, value, and span of the attribute value. To construct such tuples, Yang et al. (2022) trained five AVEQA models (Wang et al., 2020) using a large amount of silver data where attribute values were annotated using manually tailored extraction rules. Then, they applied the trained models to the Amazon Review Data in order to detect spans of values corresponding to attributes given to the models. To produce attribute value spans with high precision, they chose only attribute values that all five models extracted.

\(^3\)https://github.com/google-research-datasets/MAVE

### Table 3: Detailed statistics of the datasets. We used the T5 tokenizer to tokenize examples, attributes and values.

<table>
<thead>
<tr>
<th></th>
<th>MAVE</th>
<th>In-House Product Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Train</td>
<td>Dev.</td>
</tr>
<tr>
<td>The number of examples</td>
<td>640,000</td>
<td>100,000</td>
</tr>
<tr>
<td>without values</td>
<td>150,412</td>
<td>23,220</td>
</tr>
<tr>
<td>The number of distinct attributes</td>
<td>693</td>
<td>660</td>
</tr>
<tr>
<td>The number of distinct attribute values</td>
<td>54,200</td>
<td>21,734</td>
</tr>
<tr>
<td>The number of distinct attribute-value pairs</td>
<td>63,715</td>
<td>25,675</td>
</tr>
<tr>
<td>The number of attribute-value pairs</td>
<td>1,594,855</td>
<td>249,543</td>
</tr>
<tr>
<td>with unseen values</td>
<td>0</td>
<td>4,667</td>
</tr>
<tr>
<td>whose values appear as raw strings in the product text</td>
<td>1,594,855</td>
<td>249,543</td>
</tr>
<tr>
<td>The average number of subwords per example (input)</td>
<td>253.73</td>
<td>253.73</td>
</tr>
<tr>
<td>The average number of subwords per example (output)</td>
<td>10.35</td>
<td>10.39</td>
</tr>
<tr>
<td>The average number of attributes per example</td>
<td>1.64</td>
<td>1.64</td>
</tr>
<tr>
<td>The average number of values per example</td>
<td>2.25</td>
<td>2.25</td>
</tr>
<tr>
<td>The average number of subwords per attribute</td>
<td>2.84</td>
<td>2.82</td>
</tr>
<tr>
<td>The average number of subwords per value</td>
<td>4.15</td>
<td>3.46</td>
</tr>
</tbody>
</table>
4.2 Models

We compare the following models:

**BERT-NER:** extraction-based model. On the top of BERT, we place a classification layer that uses the outputs from the last layer of BERT as feature representations of each subword. Each subword is classified into one of the labels. We employ BILOU chunking scheme (Sekine et al., 1998; Ratinov and Roth, 2009); the total number of labels is $N \times 4 + 1$, where $N$ is the number of distinct attributes in the training data. We have used BERT as the backbone here because the common extraction-based baseline (Zheng et al., 2018) uses classic BiLSTM-CRF as the backbone (Huang et al., 2015) and BERT-based models outperform in QA-based models (Wang et al., 2020); BERT-NER can be a stronger and easily replicable baseline.

To annotate entities in text, we referred the beginning and ending positions in tuples for MAVE, and performed a dictionary matching for our in-house dataset. If annotations are overlapped, we keep the longest token length value, and drop all other overlapping values. For multi-attribute values, we adopt the most frequent attribute-value pair.

**BERT-MLC:** classification-based model. We put a classification layer on the top of BERT, and feed the embeddings of the CLS token to the classification layer as a representation of given text (Chen et al., 2022). The model predicts all possible attribute values from the representation through the classification layer. The total number of labels is the number of attribute values in the training data.

**BERT-MLC w/ Tax:** the current state-of-the-art classification-based model that can be comparable with the other methods. We added to BERT-MLC the...
label masking (Chen et al., 2022), which leverages the skewed distributions of attributes in training and testing, using an attribute taxonomy defined for our in-house data. Although this is the state-of-the-art classification-based method, it requires the attribute taxonomy as extra supervision. Since the MAVE dataset does not provide the attribute taxonomy, we train and evaluate this model only on our in-house dataset.

**T5:** generation-based model of ours. We finetune T5 on the training data obtained by each element in \{Attribute-then-value, Value-then-attribute\} × \{Random, Rare-first, Common-first\}. For random ordering, we create three training data with different random seeds, next train a model on each training data, and then chose the model that achieves the best micro F1 on the development set.

### 4.3 Implementations

We implemented all models in PyTorch.\(^5\) We used t5-base\(^6\) and sonoisa/t5-base-japanese\(^7\) in Transformers (Wolf et al., 2020), both of which have 220M parameters, as the pre-trained T5 models for MAVE and our in-house data, respectively. For training and testing, we used the default hyperparameters provided with each model. We ran teacher forcing in training, and performed beam search of size four in testing. For BERT-based models, we used bert-base-cased\(^8\) for MAVE, and cl-tohoku/bert-base-japanese\(^9\) for our in-house dataset, both of which have 110M parameters.\(^10\) We set 0.1 of a dropout rate to a classification layer.

We use Adam (Kingma and Ba, 2015) optimizer with learning rates shown in Table 14 in Appendix. We trained the models up to 10 epochs with a batch size of 32 and chose the models that perform the best micro F1 on the development set.

**Computing Infrastructure** We used NVIDIA DGX A100 GPU on a Linux (Ubuntu) server with a AMD EPYC 7742 CPU at 2.25 GHz with 2 TB main memory for performing the experiments. Table 5 shows GPU hours taken for the experiments.

### 4.4 Evaluation Measure

Following the literature (Xu et al., 2019; Wang et al., 2020; Yang et al., 2022; Shinzato et al., 2022; Chen et al., 2022), we used micro and macro precision (P), recall (R), and F1 as metrics. We compute macro performance in attribute-basis. Since the goal of PAVI is not to detect spans of values in text but to assign attribute-value pairs to products, we pick one attribute-value pair from multiple identical attribute-value pairs in MAVE (e.g., ⟨Type, jersey⟩ in Table 4). Note that we do not need this unification process for our in-house dataset because it provides unique attribute-value pairs.

Since attribute values in the MAVE dataset are based on outputs from QA-based models (Wang et al., 2020) and those in our in-house data are assigned voluntarily by sellers on our marketplace, both datasets may contain some missing values. To reduce the impact of those missing attribute-value pairs, we discard predicted attribute-value pairs if there are no ground truth labels for the attributes.

In the MAVE dataset, there are attributes whose values do not appear in the text (negative). For the ground truth with such no attribute values, models can predict no values (NN), or incorrect values (FP\(_n\)) while for the ground truth with concrete attribute values, the model can predict no values (FN), correct values (TP), or incorrect values (FP\(_p\)). Based on those types of predicted values, P and R

<table>
<thead>
<tr>
<th></th>
<th>MAVE</th>
<th>In-House Product Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BERT-NER</td>
<td>BERT-MLC</td>
</tr>
<tr>
<td>Training (10 epochs)</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>Inference (the dev set)</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Inference (the test set)</td>
<td>1.6</td>
<td>1.6</td>
</tr>
<tr>
<td>Total</td>
<td>31.6</td>
<td>31.6</td>
</tr>
</tbody>
</table>

We finetune and evaluate *-first models for each composition (2 × 2 models). For the random ordering, we train three models for each composition (3 × 2 models), and check the performance on the development set to choose the model with the best micro F1 for testing.

\(^5\)https://pytorch.org/  
\(^6\)https://huggingface.co/t5-base  
\(^7\)https://huggingface.co/sonoisa/t5-base-japanese  
\(^8\)https://huggingface.co/bert-base-cased  
\(^9\)https://huggingface.co/cl-tohoku/bert-base-japanese  
\(^10\)Training with BERT\(_\text{large}\) (330M parameters) did not work for BERT-MLC on either dataset; see Table 13 in Appendix.
are computed as follows:

\[ P = \frac{|TP|}{|TP| + |FP_p| + |FP_n|}, \quad R = \frac{|TP|}{|TP| + |FN|}. \]

\( F_1 \) is computed as \( 2 \times P \times R / (P + R) \). Note that since there are no attributes with no values in our in-house dataset, the value of \( |FP_n| \) is always 0.

### 4.5 Results

Table 6 shows the performance of each model on MAVE and our in-house datasets. Our generation-based models with *-first ordering mostly outperformed the extraction- and classification-based baselines in terms of \( F_1 \). The differences between the best models and the baselines were significant \((p < 0.0005)\) under approximate randomized test (Noreen, 1989). The higher recall of our generation-based models suggests the impact of capturing inter-value dependencies (§ 3.2).

The impact of the composition of attribute-value pairs depends on whether the output values are canonicalized. On the MAVE dataset, the models with the value-then-attribute composition outperformed those with attribute-then-value composition in terms of macro \( F_1 \). This is because all output values appear in the MAVE dataset. Thus, to the models, it is easier to generate values than attributes. Meanwhile, the advantage of value-then-attribute composition is smaller on our in-house dataset since there is no guarantee that the target values appear in the text as raw strings.

The impact of the ordering of attribute-value pairs depends on the number of attribute-value pairs per example. On the in-house dataset, the models with rare-first ordering consistently outperformed those with common-first ordering in terms of \( F_1 \). This result implies that decoding specific attribute-value pairs in advance is more helpful to generate general attribute-value pairs on the in-house dataset. Meanwhile, there is no clear difference between the models with *-first orderings on the MAVE dataset, since the number of attribute-value pairs per example is small.

These results confirm that the generative approach learns to flexibly perform canonicalization if it is required in the training data. Meanwhile, the performance of extraction- and classification-based approaches depends on whether the attribute-value pairs are canonicalized or not.

### Quantitative comparison of each approach

To see the detailed behaviors of individual approaches, we categorized the attributes in the MAVE and our in-house datasets according to the number of training examples and the number of distinct values per attribute. We divide the attributes into four accord-
4.6 Analysis

From the better macro $F_1$ of T5 with *-first ordering than with random ordering, we confirmed that our generation-based models successfully capture inter-value dependencies to decode attribute-value pairs. In what follows, we perform further analysis to see if the generative approach addresses the three challenges; namely, unseen, multi-attribute (or nested), and canonicalized values (Table 1).

Can generative models identify unseen values?

To see how effective our generative models are for unseen attribute values, we compare its performance with BERT-NER on attribute-value pairs in the test data that do not appear in the training data (13,578 and 491 unseen values exist in the MAVE and in-house datasets, respectively).

Table 9 shows the results. We can see that the T5 models outperform BERT-NER, especially in terms of macro $F_1$. Although the extraction-based approach can extract unseen values, the unified generative approach works better for extracting unseen values than the extraction-based approach.

Can generative models identify multi-attribute values?

Next, to see how effective our generative models are for identifying multi-attribute values, we compare its performance to the baselines on the in-house dataset. 'lo' and 'hi' are intervals for the number of training examples, (0, 268) and (268, $\infty$), respectively. T5 refers to the rare-first model with (V, A) composition, which achieves the best micro $F_1$.

Table 10 shows the results. We can see that the T5 models outperform all baselines in terms...
of macro $F_1$. Although the classification-based models can identify multi-attribute values, the generative models outperformed those models.

**Can generative models identify canonicalized values?** Lastly, to verify how effective our generative models are for identifying canonicalized values, we compare its performance with BERT-MLC (w/ TAX) on 207,997 attribute-value pairs whose values do not appear as raw strings in the corresponding product text in our in-house dataset.

Table 12 shows the results. The T5 models show comparable performance to and outperform the baselines in terms of micro and macro $F_1$, respectively. To see what types of canonicalization the T5 models need to perform when the canonicalized values do not appear in the text, we manually inspect attribute-value pairs whose values do not appear in text on the development set.

Table 10 exemplifies canonicalization that T5 models need to perform. From the table, we can see that the canonicalization included understanding structure in values (labels) (e.g., *iPhone* is a product of *Apple*), referring the world knowledge (*the coat has long sleeves*), recognizing paraphrases (*PU* is an abbreviation of *polyurethane*), and understanding product descriptions (*the card slot is on the left*). We conclude that our generative model addressed all the challenges in the PAVI task better than the other two approaches.

## 5 Conclusions

We have proposed a generative framework for product attribute-value identification (PAVI), which is a task to return a set of attribute-value pairs from product text on e-commerce sites. Our model can address the challenges of the PAVI task; unseen values, multi-attribute values, and canonicalized values. We finetune a pre-trained model T5 to autoregressively decode a set of attribute-value pairs from the given product text. To linearize the set of attribute-value pairs, we explored two types of attribute-value composition and three types of the orderings of the attribute-value pairs. Experimental results on two real-world datasets demonstrated that our generative approach outperformed the extraction- and classification-based baselines.

We plan to augment the ability to decode unseen values by using a pluggable copy mechanism (Liu et al., 2021). We will evaluate our model on another PAVI setting where the target attribute(s) are given.
6 Limitations

Since our generative approach to product attribute-value identification autoregressively decodes a set of attribute-value pairs as a sequence, the inference is slow (Table 5) and how to linearize the set of attribute-value pairs in the training data will affect the performance (Table 6). The best way of composing an attribute-value pair and ordering the pairs will depend on the characteristics of the datasets such as the existence of canonicalized values and the number of attribute-value pairs per example. Those who attempt to apply our method to their own datasets should keep this in mind.

Acknowledgements

This work (second author) was partially supported by JSPS KAKENHI Grant Number 21H03494. We thank the anonymous reviewers for their hard work.

References


Yuning Mao, Tong Zhao, Andrey Kan, Chenwei Zhang, Xin Luna Dong, Christos Faloutsos, and Jiawei Han. 2020. Octet: Online catalog taxonomy enrichment with self-supervision. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD ’20, page 2247–2257, New York, NY, USA. Association for Computing Machinery.

Ajinkya More. 2016. Attribute extraction from product titles in ecommerce. In KDD 2016 Workshop on Enterprise Intelligence, San Francisco, CA, USA.


Table 13: Performance of each model on two PAVI datasets. We used BERT\textsubscript{large} as a base model for extraction- and classification-based approaches. The best score is in bold face and the second best score is underlined. BERT-MLC w/ TAX uses the extra supervision (taxonomy) for label masking, and it reduces the size of labels relevant to inputs from 14,829 to 405 on average.

<table>
<thead>
<tr>
<th>Models</th>
<th>Micro</th>
<th>Macro</th>
<th>Micro</th>
<th>Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P (%)</td>
<td>R (%)</td>
<td>F1</td>
<td>P (%)</td>
</tr>
<tr>
<td>extraction-based (large)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BERT-NER</td>
<td>96.85</td>
<td>86.65</td>
<td>79.85</td>
<td>61.76</td>
</tr>
<tr>
<td>classification-based (large)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BERT-MLC</td>
<td>93.19</td>
<td>51.63</td>
<td>66.45</td>
<td>40.26</td>
</tr>
<tr>
<td>BERT-MLC w/ TAX</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>generation-based (ours)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T5 (A, V)</td>
<td>Rare-first</td>
<td>95.45</td>
<td>91.70</td>
<td>93.54</td>
</tr>
<tr>
<td></td>
<td>Common-first</td>
<td>95.29</td>
<td>92.16</td>
<td>93.70</td>
</tr>
<tr>
<td></td>
<td>Random</td>
<td>95.10</td>
<td>91.46</td>
<td>93.24</td>
</tr>
<tr>
<td></td>
<td>Rare-first</td>
<td>95.24</td>
<td>91.97</td>
<td>93.57</td>
</tr>
<tr>
<td></td>
<td>Common-first</td>
<td>94.62</td>
<td>92.85</td>
<td>93.73</td>
</tr>
<tr>
<td></td>
<td>Random</td>
<td>95.13</td>
<td>92.04</td>
<td>91.97</td>
</tr>
</tbody>
</table>

Table 14: Hyperparameters for training models.

A Final Hyperparameters Used for Each Model

Table 14 shows the hyperparameters we used for training models. Other than those, we follow the default hyperparameters of T5\textsuperscript{6,7} and BERT\textsuperscript{8,9} available from the HuggingFace models.

B Performance of Models Using BERT\textsubscript{large}

Table 13 shows the performance of models when we use BERT\textsubscript{large} as the base model for extraction- and classification-based approaches. We adopt bert-large-cased\textsuperscript{13} for MAVE and cl-tohoku/bert-large-japanese\textsuperscript{14} for our in-house data. From the table, we can see that training BERT-MLC did not work well on both datasets. Especially, we cannot compute the performance on our in-house data because the model did not predict any attribute-value pairs for all inputs. Although BERT\textsubscript{large} has a larger number of parameters (330M) than the T5 models (220M), BERT-NER based on BERT\textsubscript{large} still shows lower performance than our generative models on both datasets. This result means that our generative approach is more effective in the PAVI task than the extraction-based approaches based on BERT-NER. Meanwhile, BERT-MLC w/ TAX shows a slightly better micro F\textsubscript{1} score than ours. Given that it requires an attribute taxonomy as the extra supervision and exhibits low macro F\textsubscript{1}, the generative approach is sufficiently comparable to the classification-based approach.

\textsuperscript{13}https://huggingface.co/bert-large-cased
\textsuperscript{14}https://huggingface.co/cl-tohoku/bert-large-japanese
ACL 2023 Responsible NLP Checklist

A For every submission:

✓ A1. Did you describe the limitations of your work?
   Section 6.

✓ A2. Did you discuss any potential risks of your work?
   Section 6.

✓ A3. Do the abstract and introduction summarize the paper’s main claims?
   Abstract and Section 1.

✗ A4. Have you used AI writing assistants when working on this paper?
   Left blank.

B ✓ Did you use or create scientific artifacts?

Section 4

✓ B1. Did you cite the creators of artifacts you used?
   Section 4

✗ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
   The license or terms of the data we used is not described on the official page.

□ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
   Not applicable. Left blank.

□ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
   Not applicable. Left blank.

□ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
   Not applicable. Left blank.

✓ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
   Section 4.

C ✓ Did you run computational experiments?

Section 4

✓ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
   Section 4

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.
C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
   *Section 4 and Appendix*

C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
   *Section 4*

C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
   *Section 4*

D  ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?
   *Left blank.*

- □ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
  *No response.*

- □ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants’ demographic (e.g., country of residence)?
  *No response.*

- □ D3. Did you discuss whether and how consent was obtained from people whose data you’re using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
  *No response.*

- □ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
  *No response.*

- □ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
  *No response.*