EAVE: Efficient Product Attribute Value Extraction via Lightweight Sparse-layer Interaction

Li Yang[1]*, Qifan Wang[2]*, Jianfeng Chi[2], Jiahao Liu[3], Jingang Wang[3], Fuli Feng[4], Zenglin Xu[5,6], Yi Fang[7], Lifu Huang[8], Dongfang Liu[9]

[1]Google Research, [2]Meta AI, [3]Meituan, [4]University of Science and Technology of China, [5]Fudan University, [6]Shanghai Academy of AI for Science, [7]Santa Clara University, [8]UC Davis, [9]Rochester Institute of Technology,

lyliyang@google.com, wqfcr@meta.com

Abstract

Product attribute value extraction involves identifying the specific values associated with various attributes from a product profile. While existing methods often prioritize the development of effective models to improve extraction performance, there has been limited emphasis on extraction efficiency. However, in real-world scenarios, products are typically associated with multiple attributes, necessitating multiple extractions to obtain all corresponding values. In this work, we propose an Efficient product Attribute Value Extraction (EAVE) approach via lightweight sparse-layer interaction. Specifically, we employ a heavy encoder to separately encode the product context and attribute. The resulting non-interacting heavy representations of the context can be cached and reused for all attributes. Additionally, we introduce a light encoder to jointly encode the context and the attribute, facilitating lightweight interactions between them. To enrich the interaction within the lightweight encoder, we design a sparse-layer interaction module to fuse the non-interacting heavy representation into the lightweight encoder. Comprehensive evaluation on two benchmarks demonstrate that our method achieves significant efficiency gains with neutral or marginal loss in performance when the context is long and number of attributes is large. Our code is available here.

1 Introduction

Product attributes serve as crucial features, carrying valuable information about a product. They constitute a fundamental aspect of e-commerce platforms, offering guidance to customers for product comparisons and purchase decisions. Additionally, product attributes play a vital role in various applications for merchants, including product recommendations (Truong et al., 2022), search (Nguyen et al., 2020; Lu et al., 2021), and question answering systems (Zhang et al., 2020; Rozen et al., 2021; Huang et al., 2022). Attribute value extraction has attracted a lot of attention from both academia and industry, with a plethora of research (Zheng et al., 2018; Xu et al., 2019; Dong et al., 2020; Yan et al., 2021; Shinzato et al., 2022a; Yao et al., 2023; Chen et al., 2023) being proposed to tackle this problem.

Figure 1: An example of product associated with multiple attributes and their corresponding values extracted from the product context.

With the advancements of Transformer models (Vaswani et al., 2017; Devlin et al., 2019), attribute value extraction approaches based on Transformers (Yang et al., 2022; Wang et al., 2023) have achieved state-of-the-art performance. These methods concatenate the product attribute and context into a single text sequence and jointly encode it through the self-attention mechanism, effectively capturing the comprehensive interaction between the attribute and context. Despite their superior performance, they entail dense computation for extracting each attribute value individually. However, in real-world applications, efficient attribute value extraction is crucial for two main reasons. First, there are millions of new products being generated by the merchants everyday. Each product is typically associated with multiple attributes that describe its characteristics from various perspectives. For instance, as illustrated in Figure 1, the 'Shoes' has numerous attributes, such as 'Brand', 'Pattern', 'Type', etc., necessitating multiple inferences and resulting in substantial computation costs. Second,

*Equal contribution

Findings of the Association for Computational Linguistics: EMNLP 2024, pages 1491–1505
November 12-16, 2024 ©2024 Association for Computational Linguistics

attribute values are dynamic, undergoing changes such as updates to the product 'Price' by the merchant. As a result, re-extraction is required whenever the product profile is updated. Hence, efficient extraction poses a significant research problem.

Several recent efficient sentence pair models (Ni et al., 2022; Li et al., 2022; Yang et al., 2023b) can be applied to attribute value extraction. These models initially employ a dual encoder architecture to separately encode the product attribute and context. Subsequently, they utilize a late interaction layer to combine the attribute and context representations. This approach allows the product context to be cached and utilized for all attributes associated with the product, a concept explored in the context of attribute value extraction as well (Xu et al., 2019). However, these late interaction techniques often yield less effective extraction results due to their neglect of the interaction during the attribute and context encoding. The interaction between the two encoders is particularly crucial in attribute value extraction, especially when the context length is short (additional discussion is presented in the experiments). In such cases, dense interactions become essential to thoroughly capture the connection between the attribute and context.

To address these challenges, in this paper, we propose a novel Efficient product Attribute Value Extraction (EAVE) approach via lightweight sparse-layer interaction. In particular, we employ a heavy encoder to separately encode the product context and attribute. The resulting non-interacting heavy representations of the context can be cached and reused for all attributes. Moreover, we introduce a light encoder to jointly encode the context and the attribute, enabling lightweight interactions between them. Furthermore, we design a sparse-layer interaction module to fuse the non-interacting heavy representation into the lightweight encoder, which further enriches the interactions between the context and the context. The evaluations on two product benchmarks demonstrate that our approach achieves similar performance to the state-of-the-art models while being much efficient. We summarize the main contributions as follows:

- We propose an efficient attribute value extraction method by introducing a heavy and a light encoder to learn effective product attribute-context representations. The heavy representations can be pre-computed and reused.
- We develop a sparse-layer interaction mechanism to fuse the non-interacting and interacting representations from the heavy and light encoders respectively to improve the model effectiveness with small computation cost.
- We conduct extensive experiments and demonstrate the effectiveness of the proposed approach over several state-of-the-art baselines.

2 Related Work

Attribute Value Extraction Early attribute value extraction methods (Carmel et al., 2018; Zhao et al., 2019), including rule-based extraction (Vandic et al., 2012; Gopalakrishnan et al., 2012) and named entity recognition (NER)-based approaches (Brooke et al., 2016; Chen et al., 2019), suffer from limited coverage and closed-world assumptions. Various neural network models (Huang et al., 2015; Zheng et al., 2018; Yan et al., 2021) have also been introduced, formulating the extraction task as a sequential tagging problem. Recently, AVEQA (Wang et al., 2020) and MAVEQA (Yang et al., 2022) leverage the BERT architecture (Devlin et al., 2019) by reformulating the problem as a question-answering task, establishing the state-of-the-art in attribute value extraction. Meanwhile, OA-Mine (Zhang et al., 2022) and MixPAVE (Yang et al., 2023a) focus on zero-shot and few-shot attribute value extraction. Notably, several multi-modal works (Tan and Bansal, 2019; Zhu et al., 2020; Lin et al., 2021) explore product visual features to enhance attribute value extraction. SMARTAVE (Wang et al., 2023) designs a structured multi-modal Transformer to better encode the correlation among different product modalities. Despite achieving state-of-the-art performance, these Transformer-based methods require significant computational resources for extracting values for billions of attributes, which can be resource-intensive in many real-world scenarios.

More recently, generation-based approaches (Shinzato et al., 2023) have been proposed, including those leveraging the LLMs (Brinkmann et al., 2023; Baumann et al., 2024; Fang et al., 2024), to directly decode the attribute and value pairs together, eliminating the requirement of the product taxonomy. However, generation-based models usually fail to generate a complete set of attribute-value pairs. Moreover, their performances are suboptimal compared to the extraction-based models. More discussions are presented in the experiments and appendix A.4.

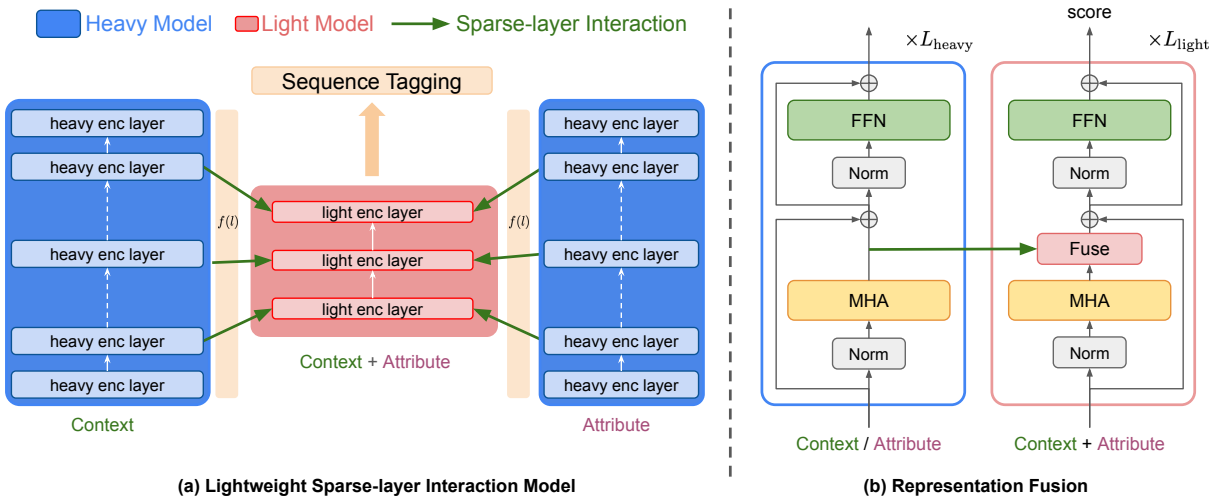

Figure 2: Overview of our EAVE model. (a) There are three key components: 1) Heavy model is used to encode the context and attribute, learning their non-interacting representations. 2) Light model generates the interacting representations of the concatenated context and attribute. 3) Sparse-layer interaction fuses the two representations from heavy and light encoders through sparse layer mapping (detailed in (b)).

Efficient Text Pair Encoders Efficient text pair encoders (Reimers and Gurevych, 2019; Chen et al., 2020; Cao et al., 2020; Humeau et al., 2020; Sun et al., 2023) can be employed in attribute value extraction by treating the product attribute and context as two text sequences. Most of these methods utilize a dual-encoder structure to individually encode the two text pieces, followed by a late interaction layer to combine the representations. In comparison with cross-attention models, they are more efficient as the text pair representations are computed independently without considering their interaction, allowing for caching and reuse. However, they often perform less optimally than cross-attention models. Several recent research (Ni et al., 2022; Li et al., 2022; Yang et al., 2023b) has focused on enhancing the performance of dual-encoder models in text pair modeling tasks by incorporating text pair interactions while preserving the efficiency of dual encoders. While these methods generally improve efficiency, the interactions between dual encoders are not fully explored. Moreover, given the typically short lengths of attributes and context, interactions between text pairs become exceedingly important and thus require careful modeling.

3 Method

3.1 Approach Overview

The objective of attribute value extraction is to extract the corresponding value for each attribute from the product context, i.e., product title, description, etc. For instance, in Figure 1, the associated value for attribute 'Brand' is 'Nature Breeze'. Previous state-of-the-art methods often concatenate the context and the attribute into a single sequence, and feeds into a Transformer encoder. Given that the context is usually much longer than the attribute, the input sequences for different attributes only vary by a small portion. However, each of these similar inputs undergoes a forward pass through the same Transformer model, a process we consider inefficient and believe can be optimized.

The overall model architecture of EAVE is shown in Figure 2. Essentially, our model contains three major components: (1) A heavy Transformer encoder to learn representations for context and attribute independently, which is referred to non-interacting representations since there is no interaction between context and attribute. These representations can then be pre-computed and cashed after the training. (2) A light Transformer encoder to compute representations for context and attribute jointly, allowing full attention between them. We referred these to interacting representations. (3) A sparse-layer interaction module to fuse the non-interacting and interacting representations and enhance the final representation. A sequence tagging module is employed on the final representation for the attribute value extraction.

3.2 Heavy Encoder

The inefficiency in cross-attention extraction models (Wang et al., 2023; Yang et al., 2023a) stems from the interdependence of computations for con-

text and attribute representations, coupled with their interactions within a single model. Given that the context sequence is typically much longer than the attribute, the entire representation undergoes certain perturbation due to the influence of attribute tokens. However, the representation computation, including the interaction between context and attribute, is dominated by the context part. Consequently, we propose isolating these interactions into a lightweight model, allowing the other representations to be independently computed through a heavy encoder. This approach enables the precomputation and caching of the context representation, which can be used for multiple attribute extractions. Formally, the computation of each heavy layer's self-attention is as follows:

$$\begin{aligned} y_l'^p &= \text{SelfAttn}(\text{Norm}(x_{l-1}^p)), \;\; y_l^p = y_l'^p + x_{l-1}^p \\ x_l'^p &= \text{MLP}(\text{Norm}(y_l^p)), \;\; x_l^p = x_l'^p + y_l^p \end{aligned} \tag{1}$$

where $p \in \{c, a\}$ is the superscript for context or attribute. x_l^p is the input heavy encoder representation for layer l. $y_l'^p$ is the self-attention output representation in layer l, which will be extracted and fused into the light Transformer encoder later. In this way, both the non-interaction representations for context ($y_l'^c$) and attribute ($y_l'^a$) can be pre-computed and cached. We initialize the heavy encoder from a pretrained T5 (Raffel et al., 2020).

3.3 Light Encoder

The heavy encoder learns the context representation independently, allowing its computation to be cashed and reused for various attribute value extraction. Previous works (Wang et al., 2020; Zhang et al., 2022) have demonstrated that the interaction between the context and attribute is critical for improving the effectiveness of the extraction. Therefore, in this work, we first introduce a light Transformer encoder to facilitate the lightweight interaction, and then fuse the representations from the heavy encoder into the light interacting representations through sparse-layer interaction (in Section 3.4). Specifically, the light encoder has a similar architecture as the heavy encoder, but with a much smaller number of parameters in terms of hidden size, number of attention heads and number of layers. The input to the light encoder is the concatenation of the context and attribute tokens. We initialize the light encoder from a smaller pretrained T5 (Raffel et al., 2020) checkpoint comparing with heavy encoder.

3.4 Sparse-layer Interaction

In the attribute value extraction task, the interactions between the product context and attribute are crucial, especially when the context length is short. While the interactions captured in the light Transformer encoder are efficient, they may not be as comprehensive for effective extraction when compared to methods using a heavy encoder with full attention. To enhance effectiveness, we introduce a sparse-layer interaction approach to merge the non-interacting representations from the heavy model into the interacting light encoder representations. This enables the transfer of valuable information encoded in the dense representations, facilitating interaction with the lightweight representation. The overall extraction performance is improved with only a marginal increase in computation cost.

Concretely, since the heavy encoder has a large number of layers, we first identify a sparse set of layers in the heavy encoder, through a layer mapping function f, to be fused into the light encoder. In this paper, we choose an even distribution layer mapping, i.e. every other $L_{\text{heavy}}/L_{\text{light}}$ heavy encoder layer will be mapped to a light encoder layer (more ablation studies on different layer mapping schemes are presented in Section 6.5). For each layer l in the light encoder, we then fuse representations from the heavy encoder $y_{f(l)}'^c$ and $y_{f(l)}'^a$ into the light encoder at the same location where the heavy encoder representations are extracted:

$$\begin{aligned} y_l' &= \text{SelfAttn}(\text{Norm}(x_{l-1})) \\ y_l &= \text{Fuse}(y_l', y_{f(l)}'^c, y_{f(l)}'^a) + x_{l-1} \\ x_l' &= \text{MLP}(\text{Norm}(y_l')) \\ x_l &= x_l' + y_l \end{aligned} \tag{2}$$

where x_l is the input light encoder representation for layer l. y_l' is the self-attention output representation in layer l, which will be fused with the heavy encoder representations $y_{f(l)}'^c$ and $y_{f(l)}'^a$ before skip connection. Note that we always fuse light and heavy representations extracted from the same location in Transformer encoder to ensure efficient fusion. As shown in Figure 2(b), we select the location to be immediately after the self-attention layer and before its skip connection (different locations for representation extraction and fusion are provided in Section 6.6). The heavy encoder representations and light encoder representations are fused with a linear interpolation for all layers.

$$\begin{aligned} &\text{Fuse}(y_l', y_{f(l)}'^c, y_{f(l)}'^q) \\ &= (1-\alpha) * y_l' + \alpha * W_{\text{adp}} \text{Concat}(y_{f(l)}'^c, y_{f(l)}'^q) \end{aligned} \tag{3}$$

where we first concatenate $y'^{c}_{f(l)}$ and $y'^{a}_{f(l)}$ along the sequence dimension, followed by an linear adaptor, W_{adp}, to project the hidden size into the same size as the light encoder, then linearly fuse with light representations y'_l. We also experiment with other fusion functions and provide discussion in Section 6.1. The final fused output of the light Transformer encoder is fed into the sequence tagging module for extraction.

3.5 Discussion

In real-world attribute value extraction, we only need to invoke the heavy encoder on each product context and attribute once to pre-compute the non-interacting representations. For each attribute, the light encoder and the sparse-layer interaction are used to compute interacting representations for the context-attribute pair. The total computation cost for each product with N attributes can be computed as $C^c_h + N \cdot (C^a_h + C_l + C_{sli})$, while previous methods would require $N \cdot C^{c+a}_h$, where C^c_h, C^a_h and C^{c+a}_h are the computation cost of the heavy encoder on the context, attribute and their concatenation, respectively. Typically C^a_h is much smaller than C^c_h, since attribute is much shorter than context. C_l is the cost of the light encoder on concatenated context and attribute. C_{sli} is the cost of the fusion module. It can be seen that our approach is much efficient. It is worth mentioning that if ignoring the C^c_h and C^a_h (since they can be cashed), our method becomes even more efficient compared to those cross-attention heavy encoder methods (Wang et al., 2023; Yang et al., 2023a).

4 Experiments

4.1 Datasets

MAVE[1] (Yang et al., 2022) is a large and diverse dataset for product attribute extraction study, which contains 3 million attribute value annotations across 1257 fine-grained categories created from 2.2 million cleaned Amazon product profiles (Ni et al., 2019). We use product title, description and metadata as context. We randomly split product ids into train and eval sets by 9:1.

AE-110K[2] (Xu et al., 2019) is collected from AliExpress Sports & Entertainment category, which contains over 110K data examples with more than 2.7K unique attributes and 10K unique values. The context length in MAVE is much longer then that in AE-110K. More details are in the Appendix.

[1] https://github.com/google-research-datasets/MAVE

[2] https://raw.githubusercontent.com/lanmanok/ACL19_Scaling_Up_Open_Tagging/master/publish_data.txt

4.2 Baselines and Settings

Our model is compared with seven state-of-the-art baselines, including three Transformer-based extraction models, AVEQA (Wang et al., 2020), MAVEQA (Yang et al., 2022), and SMARTAVE (Wang et al., 2023), one generation-based model (Shinzato et al., 2023) and three efficient text pair methods, DiPair (Chen et al., 2020), VIRT (Li et al., 2022) and MixEncoder (Yang et al., 2023b).

For all models in this paper, we use a learning rate of 1e-5, batch size 128, and trained up to 200k steps on 16 Cloud TPU V5 devices through data parallelism. Adam (Kingma and Ba, 2014) optimizer is used during training. For MAVE, we truncate the context sequence length to 512, and set attribute sequence length to 32. A T5-large model is used as the heavy encoder. For AE-110K, we pad or truncate the context sequence length to 128, and pad or truncate the attribute sequence length to 16. A T5-base model is used as the heavy encoder. For SMARTAVE, we use the SMARTAVE-text since we only focus on text based extraction. For generation-based method, we use beam search with width 4 and report the 'common first' result (which is the best). For the efficient text pair methods, a same sequence tagging module is applied to conduct the final extraction. More details on the hyper-parameters are presented in Appendix A.2.

5 Main Results

We present precision, recall, F1, and GFLOPS per example metrics for selected attributes, along with overall results on both MAVE and AE-110K in Table 1. Several key observations can be derived from these results. **First**, in comparison to state-of-the-art Transformer-based attribute value extraction methods, our EAVE approach achieves a comparable overall performance, with only a 0.2% and 0.9% decrease in F1 on MAVE and AE-110K, respectively. Notably, it significantly improves efficiency by a factor of 10, showcasing the effectiveness of our approach in efficient attribute value extraction. **Second**, although EAVE entails slightly higher computational costs compared to other efficient text pair methods, it substantially enhances extraction performance. For instance, the overall F1 score of EAVE increases by 2.34% and 5.41%

MAVE	Season			Department			Resolution			Compatibility			All			GFLOPS
Models	P(%)	R(%)	F_1(%)	P(%)	R(%)	F_1(%)	P(%)	R(%)	F_1(%)	P(%)	R(%)	F_1(%)	P(%)	R(%)	F_1(%)	
DiPair (Chen et al., 2020)	88.35	88.98	88.53	90.78	92.09	91.43	93.16	94.34	93.72	95.78	96.56	96.21	92.84	94.47	93.61	35.18
VIRT (Li et al., 2022)	89.76	90.81	90.34	93.48	94.82	93.78	94.31	95.72	95.47	97.18	98.12	97.56	94.77	96.39	95.52	41.67
MixEncoder (Yang et al., 2023b)	89.88	90.92	90.47	93.18	94.90	93.52	95.14	96.17	95.62	97.75	98.29	98.03	94.92	96.76	95.64	33.93
Generation (Shinzato et al., 2023)	89.51	87.18	88.33	91.38	89.47	90.41	94.86	91.69	93.25	96.25	95.85	96.05	94.31	92.48	93.52	87.57
AVEQA (Wang et al., 2020)	91.63	93.82	92.71	96.51	97.03	96.77	98.20	98.79	98.49	99.83	99.92	99.88	97.96	98.44	98.20	402.47
MAVEQA (Yang et al., 2022)	91.15	94.13	92.62	96.39	97.35	96.87	98.07	98.81	98.44	99.56	99.81	99.68	97.85	98.57	98.21	446.24
SMARTAVE (Wang et al., 2023)	91.17	94.16	92.64	96.47	97.38	96.92	98.15	98.78	98.46	99.62	99.83	99.72	97.88	98.60	98.24	465.71
EAVE	92.35	94.94	93.63	97.05	97.84	97.44	97.78	98.25	98.01	99.83	99.67	99.75	97.94	98.03	97.98	42.46

AE-110K	Brand Name			Material			Pattern Type			All			GFLOPS
Model	P(%)	R(%)	F_1(%)	P(%)	R(%)	F_1(%)	P(%)	R(%)	F_1(%)	P(%)	R(%)	F_1(%)	
DiPair (Chen et al., 2020)	87.64	89.87	89.25	76.31	80.29	78.62	78.72	82.58	80.43	79.74	70.69	76.82	3.75
VIRT (Li et al., 2022)	90.56	91.90	91.33	79.29	81.75	80.28	81.07	84.83	82.81	77.86	80.27	78.95	4.92
MixEncoder (Yang et al., 2023b)	90.68	92.55	91.41	80.40	81.97	81.24	81.86	84.98	83.15	78.53	80.74	79.21	3.42
Generation (Shinzato et al., 2023)	90.18	88.65	89.41	79.83	78.75	79.29	80.26	81.47	80.86	79.62	75.54	77.53	6.68
AVEQA (Wang et al., 2020)	96.36	98.57	97.46	84.01	88.89	86.38	87.42	90.41	88.89	85.01	86.09	85.54	16.11
MAVEQA (Yang et al., 2022)	96.21	98.52	97.39	83.96	88.65	86.21	87.55	90.57	89.03	84.96	86.05	85.51	18.54
SMARTAVE (Wang et al., 2023)	96.32	98.59	97.43	84.01	88.65	86.27	87.48	90.49	88.96	85.12	86.07	85.49	19.75
EAVE	96.90	97.13	97.02	53.27	53.95	53.61	87.21	91.10	89.11	85.01	84.24	84.62	5.20

Table 1: Performance comparison of selected attributes on both MAVE and AE-110K datasets. For all efficient methods, the GFLOPS do not include the pre-computation time. The standard deviation of our model on the F_1 metric is 0.13, indicating the statistical significant of the results.

on MAVE and AE-110K, respectively, compared to MixEncoder. We hypothesize that our specially designed lightweight model and sparse-layer interaction are more adept at capturing interactions compared to the late interaction layer used in previous methods. **Third**, another interesting observation is the lower overall performance on AE-110K compared to MAVE. This discrepancy is attributed to the product context in AE-110K being solely derived from the title, which is very short. The context alone may not provide sufficient useful contextual information for the attribute value extraction task. Instead, the context-attribute interaction is deemed more crucial than the context itself, aligning with our expectations. **Fourth**, the generation-based model does not perform well compared to the extraction-based models, especially on AE-110K. Our hypothese is that for the attribute value extraction task, in most cases, the value is from a text span in the product context and thus extractive models are more effective compared to the generative models. The observation is consistent with the findings in (Brinkmann et al., 2023) that a smaller LLM like Beluga-7B utilizing in-context learning fails to outperform a fine-tuned extraction-based model (i.e., AVEQA) with a significantly smaller size. More LLM-based results are reported in Appendix A.4.

6 Analysis and Discussion

6.1 Impact of the Sparse-layer Interaction

The sparse-layer interaction is a key component in our approach. To assess the impact of this module, we conducted an experiment by varying the value

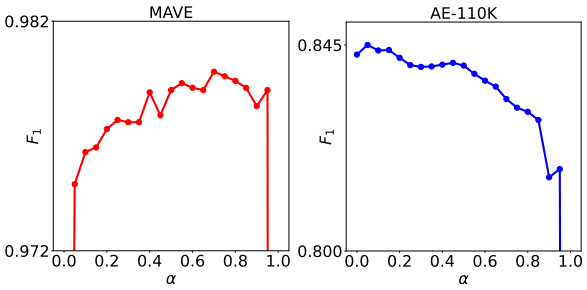

Figure 3: Impact of the sparse-layer interaction by varying the fusion weight α on both dataset.

of α from 0 to 1. The evaluation results of EAVE for different values of α are illustrated in Figure 3. It can be seen that when completely removing the heavy encoder representations by setting α to 0, and the model degrades to a single-light encoder baseline, which does not perform well compared to the heavy Transformer model. Conversely, when α is set to 1, the light encoder's self-attention outputs are entirely dominated by the heavy representations, resulting in the absence of interaction between context and attribute, which unsurprisingly yields suboptimal results. We also observe from the results that different datasets might have different optimal values of α for merging the non-interacting and interacting representations.

6.2 Different Fusion Methods

The fusion function in the sparse-layer interaction is an important factor that would impact the model performance. There are various fusion methods that can be used. In this study, we experiment with three representation fusion algorithms. The simplest one is using a linear interpolation with a fixed

	MAVE			AE-110K		
Fusion method	P (%)	R (%)	F_1 (%)	P (%)	R (%)	F_1 (%)
Fixed α	97.94	98.03	97.98	85.01	84.24	84.62
Learned α	97.02	97.08	97.05	84.31	84.24	84.27
Cross-attn	95.85	96.76	96.30	80.81	76.31	78.50

Table 2: Ablation study of representation fusion methods. For interpolation with fixed α, we use $\alpha = 0.7$ for MAVE and $\alpha = 0.05$ for AE-110K.

weight α for all layers, which is the one described and used in our main experiments. The second algorithm is the learned $alpha$. Specifically, instead of using a fixed α for all layers, we use a learnable zero-initialized α for each layer. We expect the light encoder first learns context-attribute interaction without the influence from non-interacting heavy representations, then as α updated to a non-zero value, the light encoder will gradually adapt to non-interacting heavy representations. The third fusion choice is the cross-attention fusion by first concatenating and project non-interacting heavy representations, and then updating the interacting light representations via cross attending to the projected non-interacting heavy representations, followed by a skip connection as follows:

$$\begin{aligned} &\text{Fuse}(y'_l, y'^{c}_{f(l)}, y'^{q}_{f(l)}) \\ &= \text{CrossAttn}(y'_l, W_{\text{adp}}\text{Concat}(y'^{c}_{f(l)}, y'^{q}_{f(l)}) + y'_l \end{aligned} \tag{4}$$

In contrast to linear interpolation combination, in this setup, a token's light representation not only sees heavy representation of the token itself, but also sees heavy representations of other tokens.

The results on MAVE and AE-110K for different fusion methods are shown in Table 2. It can be seen that our method achieves the best results by grid searching to select the best fixed α. It is also clear that while the strategy of learnable α achieves similar or slight worse performance, the cross-attention method performs not as good as other methods.

6.3 Impact of Heavy and Light Encoders

There are two separate encoders for learning the non-interacting representations and interacting representations. It is useful to understand how they would affect the model performance. To investigate this, we conduct experiments by modifying their parameter learning rates. In particular, we use a fixed learning rate LR_{light} for the light encoder but varying the heavy encoder learning rate as $\text{LR}_{\text{heavy}} = \beta\text{LR}_{\text{light}}$, where $\beta \geq 0$ is the learning rate ratio. Figure 4 shows the performance on the MAVE and the AE-110K datasets as a function of β. It can be seen that, the performance on MAVE

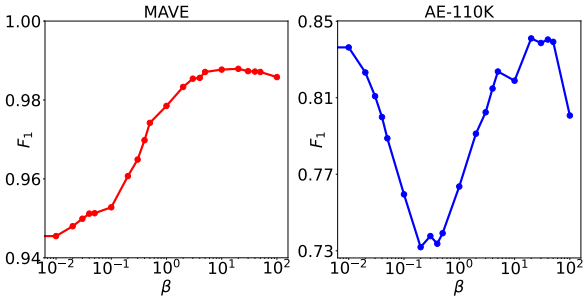

Figure 4: Impact of the heavy-light learning rate ratio β on the MAVE and AE-110K dataset. For MAVE, α is fixed to 0.7. For AE-110K, α is fixed to 0.05.

gets better as β increases from 0, suggesting the importance of updating the heavy encoder for non-interacting representations. This further validates our hypothesis that long context alone contains useful information for the attribute value extraction task. On the other hand, for AE-110K, the performance decreases as the value of β goes up, and increases again as β pass around 1.0, indicating that heavy encoder itself does not provide much useful information when context is short. In this case, interaction between context and attribute are more important.

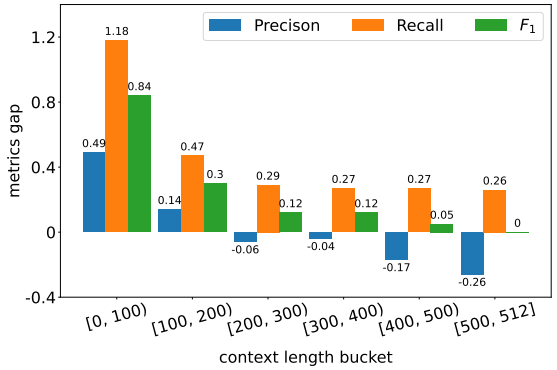

Figure 5: Precision, recall, and F1 **gaps** from the EAVE to the Transformer baseline on the MAVE dataset. We use a fixed $\alpha = 0.7$ and $\beta = 1.0$.

6.4 Results on Different Context Length

To understand how our approach behaves on different context lengths, we conduct an experiment on MAVE and report the evaluation metrics on different groups with various context sequence lengths from $\{0-100, 100-200, 200-300, 300-400, 400-500, 500-512\}$. The results are shown in Figure 5. It can be seen from the results that with the increasing of the context length, the performance gaps between our model and the Transformer baseline decreases, indicating the effectiveness of our approach in dealing with products with long context sequence.

	MAVE			AE-110K		
Layer mapping	P (%)	R (%)	F_1 (%)	P (%)	R (%)	F_1 (%)
[0, 3, 6, 9, 12, 15, 18, 21]	97.75	97.78	97.76	**85.03**	84.21	**84.62**
[1, 4, 7, 10, 13, 16, 19, 22]	97.86	97.87	97.86	84.89	84.05	84.47
[2, 5, 8, 11, 14, 17, 20, 23]	97.89	97.91	97.90	84.76	84.13	84.44
[23, 23, 23, 23, 23, 23, 23, 23]	**98.05**	**98.09**	**98.07**	83.92	84.29	84.10
[16, 17, 18, 19, 20, 21, 22, 23]	98.05	97.98	98.02	84.36	**84.36**	84.36
[0, 1, 2, 3, 4, 5, 6, 7]	96.37	96.70	96.54	85.01	84.14	84.57

Table 3: Ablation study of layer mapping on the MAVE and AE-110K datasets. Both are using EAVE L-S. For MAVE, we use $\alpha = 0.7$ and $\beta = 1.0$. For AE-110K, we use $\alpha = 0.05$ and $\beta = 0.0$.

6.5 Impact of Layer Mapping

Layer mapping is employed to select the set of sparse layers in the heavy encoder to interact with, representing a key factor in sparse-layer interaction. To assess the impact of different layer mapping strategies, we conducted a series of experiments on both datasets using T5-large with 24 layers as the heavy encoder and T5-small with 8 layers as the light encoder. Various layer mapping schemes were explored on both datasets, and their effects were evaluated. These schemes include: 1) Even distribution layer, which maps every other 3 heavy encoder layers to a light encoder layer, starting with heavy encoder layers 0, 1, and 2. 2) Only using the last heavy encoder layer. 3) Using the last 8 heavy encoder layers. 4) Using the first 8 heavy encoder layers. The values of α and β were set to 0.7 and 1.0 on MAVE, and 0.05 and 0.0 on AE-110K.

Results are reported in Table 3. It can be observed that using heavy representations from the last layer achieves the best results on MAVE, while for AE-110K, the even distribution strategy generally performs better. The rationale behind this distinction is that, for MAVE, the heavy encoder is updated to provide improved task-specific representations in the last layer. In contrast, for AE-110K, the heavy encoder is frozen, and multiple layers contribute more diverse information.

6.6 Representation Fusion Location

Within a Transformer encoder layer, the heavy and light representations can be extracted and fused at locations other than immediately after self-attention. Table 4 presents a comparison of performance for six different fusion locations, including 1) Immediately before the self-attention layer and after its pre-normalization, 2) Immediately after the self-attention layer, 3) After the skip connection of the self-attention layer, 4) Immediately before the MLP layer after its pre-normalization, 5) Immediately after the MLP layer, and 6) After the skip connection of the MLP layer. As observed, for MAVE with $\beta = 1.0$ (corresponding to fine-tuning the heavy encoder), the optimal fusion location is immediately before the self-attention layer and after its pre-normalization. For AE-110K with $\beta = 0.0$ (corresponding to freezing the heavy encoder), the best fusion location is immediately before the MLP layer and after its pre-normalization.

	MAVE			AE-110K		
Fusion location	P (%)	R (%)	F_1 (%)	P (%)	R (%)	F_1 (%)
before attn	**98.14**	**98.17**	**98.16**	84.11	84.34	84.23
after attn	97.95	98.01	97.98	84.72	84.08	84.40
after attn & skip	97.79	98.05	97.92	85.04	84.12	84.58
before mlp	97.80	97.91	97.85	**85.08**	**84.39**	**84.74**
after mlp	97.68	97.76	97.72	84.83	84.12	84.48
after mlp & skip	97.71	97.94	97.83	84.44	84.06	84.25

Table 4: Ablation study on representation fusion location on both datasets. For MAVE, we use $\alpha = 0.7$ and $\beta = 1.0$. For AE-110K, we use $\alpha = 0.05$ and $\beta = 0.0$.

6.7 Heavy Encoder Size

In this section, we ablate the heavy encoder size in our EAVE approach. The results on MAVE and AE-110K are shown in Table 5. As can be seen, for the MAVE dataset, as the heavy encoder size increases, the performance increases, while for AE-110K, as the heavy encoder size increases, the performance does not continuously increase. These results support our hypothesis that the context alone provides more useful information for the MAVE dataset comparing with the AE-110K dataset.

	MAVE			AE-110K		
Model size	P (%)	R (%)	F_1 (%)	P (%)	R (%)	F_1 (%)
B-S	96.83	97.15	96.99	84.94	84.02	84.48
L-S	97.94	98.03	97.98	84.89	84.11	84.50
XL-S	98.25	98.55	98.40	84.75	84.12	84.43

Table 5: Ablation study on the heavy encoder size on both datasets. For MAVE, we use $\alpha = 0.7$ and $\beta = 1.0$. For AE-110K, we use $\alpha = 0.05$ and $\beta = 0.0$.

7 Conclusions

Efficient product attribute value extraction is an important research problem in many real-world applications. In this work, we proposed an efficient attribute value extraction method with lightweight sparse-layer interaction. Specifically, we decouple the computations of context-attribute non-interacting representations and interacting representations, using a heavy and light Transformer encoders respectively. Additionally, these representations are fused together through the sparse-layer interaction. We conducted benchmarks and systematic ablation studies on two open-sourced prod-

uct attribute value extraction datasets. The results demonstrate that our method achieves much fast inference speed while maintaining the performance to the large encoder.

Limitations

There are two limitations of our current EAVE approach. First, although the lightweight sparse-layer interaction method is tested in product attribute value extraction task, we believe this method can be generalized to any other tasks that are able to be converted to a context-query format, such as question answering, text pair classification, etc. As long as the context is long and number of queries per context is large, our method will offer dramatic efficiency gain. We plan to explore a generalization approach of our model. Second, we only investigated the lightweight sparse-layer interaction method under sequence labeling setup. However, recent works on Large language Models (LLM) have shown the success of decoder models. In the future, we plan to systematically investigate our method under Transformer decoder setup.

References

Nick Baumann, Alexander Brinkmann, and Christian Bizer. 2024. Using llms for the extraction and normalization of product attribute values. *CoRR*, abs/2403.02130.

Alexander Brinkmann, Roee Shraga, and Christian Bizer. 2023. Product attribute value extraction using large language models. *CoRR*, abs/2310.12537.

Julian Brooke, Adam Hammond, and Timothy Baldwin. 2016. Bootstrapped text-level named entity recognition for literature. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*. The Association for Computer Linguistics.

Qingqing Cao, Harsh Trivedi, Aruna Balasubramanian, and Niranjan Balasubramanian. 2020. Deformer: Decomposing pre-trained transformers for faster question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 4487–4497. Association for Computational Linguistics.

David Carmel, Liane Lewin-Eytan, and Yoelle Maarek. 2018. Product question answering using customer generated content - research challenges. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, page 1349–1350, New York, NY, USA. Association for Computing Machinery.

Jiecao Chen, Liu Yang, Karthik Raman, Michael Bendersky, Jung-Jung Yeh, Yun Zhou, Marc Najork, Danyang Cai, and Ehsan Emadzadeh. 2020. Dipair: Fast and accurate distillation for trillion-scaletext matching and pair modeling. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 2925–2937. Association for Computational Linguistics.

Ke Chen, Lei Feng, Qingkuang Chen, Gang Chen, and Lidan Shou. 2019. Exact: Attributed entity extraction by annotating texts. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, page 1349–1352, New York, NY, USA. Association for Computing Machinery.

Wei-Te Chen, Keiji Shinzato, Naoki Yoshinaga, and Yandi Xia. 2023. Does named entity recognition truly not scale up to real-world product attribute extraction? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: EMNLP 2023 - Industry Track, Singapore, December 6-10, 2023*, pages 152–159. Association for Computational Linguistics.

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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Xin Luna Dong, Xiang He, Andrey Kan, Xian Li, Yan Liang, Jun Ma, Yifan Ethan Xu, Chenwei Zhang, Tong Zhao, Gabriel Blanco Saldana, Saurabh Deshpande, Alexandre Michetti Manduca, Jay Ren, Surender Pal Singh, Fan Xiao, Haw-Shiuan Chang, Giannis Karamanolakis, Yuning Mao, Yaqing Wang, Christos Faloutsos, Andrew McCallum, and Jiawei Han. 2020. Autoknow: Self-driving knowledge collection for products of thousands of types. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 2724–2734. ACM.

Chenhao Fang, Xiaohan Li, Zezhong Fan, Jianpeng Xu, Kaushiki Nag, Evren Körpeoglu, Sushant Kumar, and Kannan Achan. 2024. Llm-ensemble: Optimal large language model ensemble method for e-commerce product attribute value extraction. *CoRR*, abs/2403.00863.

Vishrawas Gopalakrishnan, Suresh Parthasarathy Iyengar, Amit Madaan, Rajeev Rastogi, and Srinivasan H. Sengamedu. 2012. Matching product titles using web-based enrichment. In *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*, pages 605–614. ACM.

Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. 2015. Deep learning with limited numerical precision. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1737–1746. JMLR.org.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Guanhuan Huang, Xiaojun Quan, and Qifan Wang. 2022. Autoregressive entity generation for end-to-end task-oriented dialog. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 323–332, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2020. Poly-encoders: Architectures and pre-training strategies for fast and accurate multi-sentence scoring. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Dan Li, Yang Yang, Hongyin Tang, Jiahao Liu, Qifan Wang, Jingang Wang, Tong Xu, Wei Wu, and Enhong Chen. 2022. VIRT: improving representation-based text matching via virtual interaction. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 914–925. Association for Computational Linguistics.

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 *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, pages 3262–3270. ACM.

Hanqing Lu, Youna Hu, Tong Zhao, Tony Wu, Yiwei Song, and Bing Yin. 2021. Graph-based multilingual product retrieval in e-commerce search. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Papers, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 146–153. Association for Computational Linguistics.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 14014–14024.

Thanh V. Nguyen, Nikhil Rao, and Karthik Subbian. 2020. Learning robust models for e-commerce product search. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 6861–6869. Association for Computational Linguistics.

Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B. Hall, Daniel Cer, and Yinfei Yang. 2022. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. In *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 1864–1874. Association for Computational Linguistics.

Jianmo Ni, Jiacheng Li, and Julian J. 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 2019, Hong Kong, China, November 3-7, 2019*, pages 188–197. Association for Computational Linguistics.

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. *Journal of Machine Learning Research*, 21(140):1–67.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. 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 2019, Hong Kong, China, November 3-7, 2019*, pages 3980–3990. Association for Computational Linguistics.

Ohad Rozen, David Carmel, Avihai Mejer, Vitaly Mirkis, and Yftah Ziser. 2021. Answering product-questions by utilizing questions from other contextually similar products. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 242–253. Association for Computational Linguistics.

Keiji Shinzato, Naoki Yoshinaga, Yandi Xia, and Wei-Te Chen. 2022a. 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), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 227–234. Association for Computational Linguistics.

Keiji Shinzato, Naoki Yoshinaga, Yandi Xia, and Wei-Te Chen. 2022b. 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), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 227–234. Association for Computational Linguistics.

Keiji Shinzato, Naoki Yoshinaga, Yandi Xia, and Wei-Te Chen. 2023. A unified generative approach to product attribute-value identification. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 6599–6612. Association for Computational Linguistics.

Rongyi Sun, Ziran Li, Yifeng Ding, Qifan Wang, Jingang Wang, Haitao Zheng, Wei Wu, and Yunsen Xian. 2023. Fusion or defusion? flexible vision-and-language pre-training. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5105–5119, Toronto, Canada. Association for Computational Linguistics.

Hao Tan and Mohit Bansal. 2019. LXMERT: learning cross-modality encoder representations from transformers. 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 2019, Hong Kong, China, November 3-7, 2019*, pages 5099–5110. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Quoc-Tuan Truong, Tong Zhao, Changhe Yuan, Jin Li, Jim Chan, Soo-Min Pantel, and Hady W. Lauw. 2022. Ampsum: Adaptive multiple-product summarization towards improving recommendation captions. In *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, pages 2978–2988. ACM.

Damir Vandic, Jan-Willem van Dam, and Flavius Frasincar. 2012. Faceted product search powered by the semantic web. *Decis. Support Syst.*, 53(3):425–437.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

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*, KDD '20, page 47–55, New York, NY, USA. Association for Computing Machinery.

Qifan Wang, Li Yang, Jingang Wang, Jitin Krishnan, Bo Dai, Sinong Wang, Zenglin Xu, Madian Khabsa, and Hao Ma. 2023. Smartave: Structured multimodal transformer for product attribute value extraction. In *In Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates.*, page 263–276. Association for Computational Linguistics.

Huimin Xu, Wenting Wang, Xin 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 Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 5214–5223. Association for Computational Linguistics.

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, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4694–4705. Association for Computational Linguistics.

Li Yang, Qifan Wang, Jingang Wang, Xiaojun Quan, Fuli Feng, Yu Chen, Madian Khabsa, Sinong Wang, Zenglin Xu, and Dongfang Liu. 2023a. Mixpave: Mix-prompt tuning for few-shot product attribute value extraction. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 9978–9991. Association for Computational Linguistics.

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 *WSDM '22: The Fifteenth ACM International Conference on Web Search and Data Mining, Virtual Event / Tempe, AZ, USA, February 21 - 25, 2022*, pages 1256–1265. ACM.

Yuanhang Yang, Shiyi Qi, Chuanyi Liu, Qifan Wang, Cuiyun Gao, and Zenglin Xu. 2023b. Once is enough: A light-weight cross-attention for fast sentence pair modeling. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 2800–2806. Association for Computational Linguistics.

Barry Menglong Yao, Yu Chen, Qifan Wang, Sijia Wang, Minqian Liu, Zhiyang Xu, Licheng Yu, and Lifu Huang. 2023. AMELI: enhancing multimodal entity linking with fine-grained attributes. *CoRR*, abs/2305.14725.

Wenxuan Zhang, Yang Deng, Jing Ma, and Wai Lam. 2020. Answerfact: Fact checking in product question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 2407–2417. Association for Computational Linguistics.

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 *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, pages 3153–3161. ACM.

Jie Zhao, Ziyu Guan, and Huan Sun. 2019. Riker: Mining rich keyword representations for interpretable product question answering. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 1389–1398, New York, NY, USA. Association for Computing Machinery.

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, KDD 2018, London, UK, August 19-23, 2018*, pages 1049–1058. ACM.

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 2020, Online, November 16-20, 2020*, pages 2129–2139. Association for Computational Linguistics.

A Appendix

A.1 Datasets

This section contains more details on the datasets. The train and eval splits on the selected attributes in Table 1 are shown in Table 6 for the MAVE dataset and Table 7 for the AE-110K dataset. We select those attributes to ensure a broad spectrum of number of examples per attribute.

Splits	MAVE				
	Season	**Department**	**Resolution**	**Compatibility**	**All**
Train	59199	20108	13792	6544	4011570
Eval	11431	3766	2492	1222	755968

Table 6: Statistics of the MAVE dataset.

Splits	AE-110K			
	Brand Name	**Material**	**Pattern Type**	**All**
Train	9098	3001	1496	88479
Eval	2316	810	348	21888

Table 7: Statistics of the AE-110K dataset.

A.1.1 MAVE

MAVE is a dataset with long and structured context. After tokenization, the average context sequence length is 281, but there is a long tail distribution such that 11.2% examples are with sequence length larger than 512. We truncate or pad the context length to 512 and the attribute length to 32. Each product belongs to one category, which contains multiple possible attributes. During real-world application inference, for each product, we need to perform attribute value extraction for all attributes of its category. The statistics of several selected categories are shown in Table 8. As can be seen, some categories have O(10) number of attributes and a large amount of examples. Our EAVE method will achieve dramatic efficiency gain on such categories, which is presented in Section A.5.

Categories	**Shoes**	**Mobile Phones**	**Televisions**	**Dresses**
Attributes	15	12	11	5
Train	599088	49543	22041	99276
Eval	114269	9671	3958	18803

Table 8: Statistics of the MAVE dataset for the selected categories.

A.1.2 AE-110K

AE-110K is a dataset with context being only the title, so its context is short and simple. After tokenization, the average context sequence length is 32, and there are less than 0.1% examples with sequence length larger than 64. We truncate or pad the context length to 64 and attribute length to 8. Different from the MAVE dataset, a product in AE-110K doesn't have a category label. But each product can contain multiple attributes.

Following the work in (Shinzato et al., 2022b), we manually fix several quality issues, including HTML entities, and extra white spaces in titles, attributes, and values, and the same attributes sometimes have different letter cases. We thus decoded HTML entities, converted trailing spaces into a single space, and removed white spaces at the beginning and ending. We also remove duplicated tuples.

Dropout Rate	0.1
Adam β_1	0.9
Adam β_2	0.99
Adam ϵ	1e-8
Activation Type	bfloat16

Table 9: Hyper-parameters and configs.

A.2 Training Details

For all models in this paper, we use a learning rate of 1e-5, batch size 128, and Adam (Kingma and Ba, 2014) optimizer for training. We train up to 200k steps on 16 V5 Cloud TPUs using data parallelism. For both datasets, we use the same SentencePiece (Kudo and Richardson, 2018) tokenizer as in the T5 paper (Raffel et al., 2020). We compute the GFlops numbers in Table 1 based on the sequence lengths mentioned in Section A.1. More details on the hyper-parameters and configs are in Table 9.

Note that the learning rate, batch size and training steps are the same for all baselines, while the other hyperparameters are set to the optimal values in the original papers. W use the same learning rate, batch size and training steps for two main reasons. First, we want to ensure all models consume the same amount of training data (by using the same batch size and training steps) in order to achieve a fair comparison on the efficiency. Second, we set 200k training steps to ensure all models are sufficiently converged. In fact, we've observed that most methods converge within 50k steps. We also tried different hyperparameters and found the performances are quite stable.

Method	MAVE		AE-110K	
	F_1 (%)	GFLOPS	F_1 (%)	GFLOPS
AVEQA (backbone)	98.20	402.47	85.54	16.11
Pruning (Michel et al., 2019)	94.36	246.62	80.17	8.85
Distillation (Hinton et al., 2015)	95.11	270.49	81.55	9.23
QAT (Gupta et al., 2015)	93.57	85.45	80.49	3.91
EAVE (ours)	97.98	42.46	84.62	5.20

Table 10: Comparison with traditional efficient methods, including Pruning (Michel et al., 2019), Distillation (Hinton et al., 2015) and Quantization (Gupta et al., 2015), on both datasets.

Methods	AE-110K			
	Precision	Recall	F1	GFLOPS
SMARTAVE (Wang et al., 2023)	85.12	86.07	85.49	19.75
LLaMa2 7B (Touvron et al., 2023)	83.65	84.77	84.15	6578
EAVE	85.01	84.24	84.62	5.20

Table 11: Comparison results with several LLMs on the AE-110K dataset.

Category	Model	P (%)	R (%)	F_1 (%)	GFLOPS
Shoes	T5-Small	97.81	98.23	98.02	42.46
	T5-Base	99.09	99.21	99.15	131.10
	T5-Large	99.48	99.51	99.50	402.47
	EAVE L-S	99.48	99.48	99.48	89.86
Mobile Phones	T5-Small	81.02	89.81	85.19	42.46
	T5-Base	90.86	95.57	93.16	131.10
	T5-Large	95.04	97.57	96.29	402.47
	EAVE L-S	96.08	96.95	96.52	96.22
Televisions	T5-Small	88.32	92.57	90.39	42.46
	T5-Base	95.81	97.06	96.43	131.10
	T5-Large	98.37	98.61	98.49	402.47
	EAVE L-S	98.69	98.53	98.61	99.11
Dresses	T5-Small	94.53	95.5	95.01	42.46
	T5-Base	97.75	97.98	97.87	131.10
	T5-Large	98.78	98.87	98.83	402.47
	EAVE L-S	98.85	98.79	98.82	140.72

Table 12: Performance and cost comparison on the MAVE dataset sliced by categories with multiple attributes.

A.3 Comparison with Traditional Efficient Techniques

Traditional efficiency optimization techniques such as quantization, distillation and pruning are orthogonal to our method, as they are not targeting the efficiency improvement under the scenario of a single long context with multiple short queries. In this study, we conduct comprehensive experiments with these standard methods using AVEQA as the backbone. Specifically, for pruning (Michel et al., 2019) technique, we prune 50% of the network. For vanilla distillation (Hinton et al., 2015), we distill to a 6-layer model with roughly 50% parameters. For quantization, we use Quantization-Aware Training (QAT) (Gupta et al., 2015) with 4-bit from the original 32-bit.

The results on both datasets are reported in Table 10. It can be seen that while the GFLOPS of these methods decrease, there is significant performance drop compared with the original backbone. Moreover, when dealing with multiple attributes for a single product, they still need to encode the long product context multiple times. On the other hand, our approach is able to achieve good efficiency with on par performance, indicating the effectiveness of our modeling.

A.4 Comparison with LLM

Large language models become the defacto in many NLP applications. Therefore, in this section, we conduct a comparison with LLaMa2 7B model (Touvron et al., 2023). For LLaMa2 7B, we set the batch size to 1, sequence length to 256, hidden dimension to 4096, number of layers to 32 and run it on 8 GPUs. For fair comparison between our method and LLM attribute value extraction, we extract different attributes separately by concatenating context and attribute as inputs to LLM. We also tried extracting attributes together in a single prompt, which leads to worse performance due to LLMs hallucination issue, and requires LLM with a larger finetuning set to achieve similar performance. The prompt we use is "Please extract the attribute value of attribute from context".

The evaluation results are reported in Table 11. As it can be seen in the table, LLaMa2 7B is able to achieve reasonable performance, while the inference cost is extremely expensive. Our hypothesis is that for the attribute value extraction task, in most cases, the value is from a text span in the product context and thus extractive models are more effective compared to the generative models. The observation is consistent with the findings in (Brinkmann et al., 2023) that a smaller LLM like Beluga-7B utilizing in-context learning fails to outperform a fine-tuned BERT-based sequence tagging model (i.e., AVEQA) with a significantly smaller size, as indicated in their Tables 9 and 13.

A.5 Results on Selected Categories

We present results on selected categories for the MAVE dataset in Table 12. We show results of our EAVE model and three T5 baseline models of different sizes: Small, Base, and Large. GFLOPS per example is computed under a real world scenario: for a given product with N attributes, we

only need to compute context heavy representations once, so the GFLOPS per product is computed by $C_h^c/N + C_h^a + C_l + C_{sli}$. For simplicity, we don't consider the cost saving from caching heavy attribute representations. As can be seen from Table 12, when N increases we can observe the following: comparing with T5-Large, EAVE's performance is close, while its cost is dramatically reduced; comparing with T5-Small, EAVE's performance is significantly better, while its cost is only marginally heavier. The effectiveness of our approach on efficient is validated from those results.

A.6 Practical Usage of Efficient Extraction

In this section, we'd like to provide more insights into the importance and the practical usage of efficient attribute value extraction methods.

First, it is a vast scale of real-world product system such as Amazon's or Google Shopping's online catalogs, numbering in the hundreds of millions to billions. It's important to note that each product is associated with multiple attributes, averaging more than 10 attributes per product. Consequently, any updates to the model or system necessitate a comprehensive re-extraction across all products, entailing billions of model inference calls (approximately 100 million products multiplied by an average of 10 attributes per product). To illustrate, we conducted a test involving 2 billion extractions using a 3-layer distilled AVEQA model (BERT-base) across 20,000 CPU machines, which took more than two weeks to complete.

Second, even if we were to limit the inference to only new and modified products, it would still require a significant time investment - approximately 7-10 hours per day. It's worth emphasizing that this scenario pertains to a 3-layer BERT-base model; the computational cost would escalate considerably for a full BERT model and LLMs.

In addition, attribute value extraction serves as a crucial component for generating features in retrieval or ranking systems, such as web search and advertisements. For these applications, extraction must encompass all products available on the web across multiple platforms, potentially numbering in the tens of billions. Therefore, we believe that efficient product attribute value extraction represents a significant challenge.

A.7 Extraction on Noisy Data

In real-world product attribute value extraction, the products usually contain noise and even contradictory information within input texts. In fact, during our experimentation on product web pages extraction, we have indeed encountered failure cases due to noisy product data. For instance, a product title might mention 'red' shoes while the product description describes 'blue' pairs. One straightforward approach to address such discrepancies is to implement post-processing filtering similar to the method mentioned above. Specifically, when the model extracts multiple spans or values for an attribute (e.g., 'red' and 'blue') that do not align, we can simply return an UNKNOWN extraction.

Noise Level (during inference)	p = 0	p = 0.1	p = 0.2	p = 0.4
trained on noisy data	96.96	97.10	96.99	97.25
trained on clean data	97.27	95.36	93.57	90.39

Table 13: Model performance with noisy data.

To further understand the behavior of our model on noisy product data, we conduct an experiment by introducing noise to the context of the MAVE dataset. Specifically, with a probability p, for product description, we append up to 5 random selected attribute values from within the product category. With the same probability, we add another paragraph containing up to 5 random selected attribute values from within the product category. We then randomly split the noise augmented dataset into the train and eval set, and trained for 100k steps. The results are shown in Table 13. It can be seen from the preliminary results that: 1) When training on noisy data, the performance of our model is relatively stable since the training also sees the noise while the label remains correct. 2) When training on clean data, the performance clearly drops, which is consistent with our expectation.