PAM: Understanding Product Images in Cross Product Category Attribute Extraction

Rongmei Lin[1*], Xiang He[2], Jie Feng[2], Nasser Zalmout[2], Yan Liang[2], Li Xiong[1], Xin Luna Dong[2]
[1]Emory University [2]Amazon
[1]{rlin32,lxiong}@emory.edu [2]{xianghe,jiefeng,nzalmout,ynliang,lunadong}@amazon.com

ABSTRACT

Understanding product attributes plays an important role in improving online shopping experience for customers and serves as an integral part for constructing a product knowledge graph. Most existing methods focus on attribute extraction from text description or utilize visual information from product images such as shape and color. Compared to the inputs considered in prior works, a product image in fact contains more information, represented by a rich mixture of words and visual clues with a layout carefully designed to impress customers. This work proposes a more inclusive framework that fully utilizes these different modalities for attribute extraction. Inspired by recent works in visual question answering, we use a transformer based sequence to sequence model to fuse representations of product text, Optical Character Recognition (OCR) tokens and visual objects detected in the product image. The framework is further extended with the capability to extract attribute value across multiple product categories with a single model, by training the decoder to predict both product category and attribute value and conditioning its output on product category. The model provides a unified attribute extraction solution desirable at an e-commerce platform that offers numerous product categories with a diverse body of product attributes. We evaluated the model on two product attributes, one with many possible values and one with a small set of possible values, over 14 product categories and found the model could achieve 15% gain on the Recall and 10% gain on the F1 score compared to existing methods using text-only features.

ACM Reference Format:
Rongmei Lin[1*], Xiang He[2], Jie Feng[2], Nasser Zalmout[2], Yan Liang[2], Li Xiong[1], Xin Luna Dong[2]. 2021. PAM: Understanding Product Images in Cross Product Category Attribute Extraction. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3447548.3467164

(a) Additional modalities provide critical information

(b) Different modalities cross-validates retrieved information

Figure 1: Compared to existing attribute value extraction works which focused on text-only input features, our approach is able to take extra multi-modal information (Visual Features and OCR Tokens) from product image as input.

1 INTRODUCTION

Product attributes, such as brand, color and size, are critical product features that customers typically use to differentiate one product from another. Detailed and accurate attribute values can make it easier for customers to find the products that suit their needs and give them a better online shopping experience. It may also increase the customer base and revenue for e-commerce platforms. Therefore, accurate product attributes are essential for e-commerce applications such as product search and recommendation. Due to the manual and tedious nature of entering product information [6], the product attributes acquired from a massive number of retailers and manufacturers on e-commerce platforms are usually incomplete, noisy and prone to errors. To address this challenge, there has been a surge of interest in automatically extracting attribute values from readily available product profiles [6, 12, 22, 23, 25, 26]. Most of the existing works rely only on the textural cues obtained from the text descriptions in the product profiles, which is often far from sufficient to capture the target attributes.

In this work, we propose to leverage the product images in the attribute value extraction task. Besides the commonly used textural features from product descriptions, our approach simultaneously utilizes the generic visual features and the textual content hidden in the images, which are extracted through Optical Character Recognition (OCR). We studied 30 popular product attributes applicable

[*]Work performed during internship at Amazon.

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08...$15.00
https://doi.org/10.1145/3447548.3467164

to different product categories, including electronics, home innovation, clothes, shoes, grocery, and health. We observed that over 20% of the attribute values that are missing from the corresponding Amazon web pages can only be identified from the product images. We illustrate this with an intuitive example in Figure 1. In general, we identified two main areas where the product images can be particularly useful:

- **Additional information:** Important cues are sometimes absent in the product text descriptions. In Figure 1(a), the brand of the product is *"Lava"*, which is prominently displayed in the product image but not mentioned in the product title. In this case, OCR could perfectly recover the missing brand from the product image.
- **Cross validation:** The product title is likely to contain multiple possible values for one target attribute and the product image could help in disambiguating the correct value. In Figure 1(b), for the attribute *"Item Form"*, the product title contains the word *"Stick"*, *"Cream"* and *"Powder"*. However, both the product shape and the word *"Stick"* in the image strongly indicate the correct value should be *"Stick"*.

Despite the potential, leveraging product images for attribute value extraction remains a difficult problem and faces three main challenges:

- **C1: Cross-modality connections:** There are many intrinsic associations between product titles and images. An effective model needs to seamlessly and effectively make use of information from three modalities, including product images, texts in the images, and texts from product profiles.
- **C2: Domain-specific expressions:** The texts in the product images and product profiles are usually packed with certain phrases that are unique to a specific retail domain. For example, *"cleaning ripples"* in the category of toilet paper is a special wavy pattern to help with cleaning. *"free and clear"* in the category of detergent means that it is scent-free. In general, language models or word embeddings pre-trained on public corpora are unable to accurately capture and ground these domain-specific phrases.
- **C3: Multiple categories:** For the same attribute, there may be little overlap between the attribute values for different product categories. For example, the vocabulary for the attribute *"size"* in T-shirts (*i.e.*, small, median, large, x-large) is completely different from baby diapers (*i.e.*, newborn, 1, 2, 3, etc.). Therefore, a model trained on one product category may generalize poorly to other categories.

Existing solutions for multi-modal information extraction [3, 13, 17, 19, 20] fall short in the e-commerce domain, as it cannot address challenges **C2** and **C3**. On the other hand, text extraction solutions that manage to extract attribute values across multiple product categories [12] are text focused, and the techniques cannot easily be transferred to image information extraction. A comparison between these models are summarized in Table 1. In this paper, we address the central question: *how can we perform multi-modal product attribute extraction across various product categories?*

Inspired by recent progress on text visual question answering (*Text VQA)* [10, 19], we address challenge **C1** with a multi-modal sequence-to-sequence generation model. Our input comes from three folds: *1)* the images, containing both OCR results to capture texts in the image, along with visual features generated through Faster RCNN [18], *2)* the textual product profile, and *3)* pre-defined possible values for the target attribute. The sequence-to-sequence generator allows the extraction output to expand beyond substrings mentioned in product profiles or images. We adopt a transformer-based model in order to incorporate cross-modal attention between texts and images. The pre-defined vocabulary, which can be retrieved from the training data, is used to address challenge **C2**, by exposing the model to domain-specific target values.

Table 1: Comparison between Different Methods

Methods	C1			C2	C3
	Text	Image	OCR	Domain	Category
BAN[13]	√	√			
LXMERT[20]	√	√			
LoRRA[19]	√	√	√		
OpenTag[26]	√			√	
TXtract[12]	√			√	√
Ours	√	√	√	√	√

We further extend our model to a cross-category extraction model to address challenge **C3**. Towards this end, we utilize two different techniques: First, instead of using a fixed pre-defined vocabulary, the model dynamically switches between different category-specific vocabularies based on the category information of the input product. Second, we predict the product category as part of the decoding process, in a multi-task training setup. Instead of the common practice of introducing an auxiliary task as an additional training objective, we require our model to decode the product category as a part of the output. This implicit multi-task training fits naturally with our sequence-to-sequence model architecture.

We summarize the contributions of this work as follows:

- We propose a transformer-based sequence-to-sequence model to extract product attributes jointly from textual product profile, visual information, and texts in product images. To the best of our knowledge, this is the *first* work for multi-modal product attribute extraction.
- We extend our basic solution to a cross-product-category extraction model by *1)* equipping the model with an external dynamic vocabulary conditioned on product category and *2)* multi-task training incorporated with our sequence-to-sequence model.
- We conduct extensive experiments to evaluate our solution on a dataset collected from a public e-commerce website across multiple product categories. Our approach consistently outperforms state-of-the-art solutions by 15% on recall and 10% on F1 metric.

The remainder of the paper is organized as follows: We first review related work about attribute value extraction and visual question answering in Section 2. The problem formulation is given in Section 3. In Section 4 we describe our model in the case of single product category. In Section 5, we elaborate the necessary components to make the model aware of different product categories. Finally, we show the experimental results in section 6 and section 7 concludes the paper.

2 RELATED WORK

2.1 Value Extraction for Given Attributes

There are many works on extracting product attributes from product textual description. The open-tag model [26] used a Bi-directional LSTM model followed by conditional random field (CRF). Recent works focus on training a single model to perform value extraction for multiple product categories or multiple attribute types. Reference [23] and [22] proposed to include the attribute type in the model inputs so that one model could extract value for different attribute types based on this input. Reference [22] also introduced a distilled masked language model loss to help the model generalize better for attribute types or attribute values not seen in the training data. Reference [12] provided a solution for cross product category attribute value extraction. An auxiliary task of predicting product category is used to improve the intermediate representation. Product category is also represented by its Poincaré embedding and used as inputs to compute attention in later layers.

Most of the works mentioned above formulate the attribute exaction task as a sequence tagging problem, in which the output is the start and end position of the attribute value in input text. This work formulates the task as a sequence generation problem. The benefit is that the model can produce the correct attribute value even if it is phrased differently in the input texts.

Utilizing image to extract product attributes has received attention recently in, for example, [27] where a gated attention layer is introduced to combine information from product image and text. This work is different from [27] in that we utilize texts in product images which is shown to be useful in product attributes extraction task.

2.2 Visual Question Answering

Our task is related to the visual question answering (VQA) task in that both tasks study interaction between text and image modality. Early works in VQA focus on the design of the attention mechanism to merge information from image and text modality, such as the bilinear attention in [13]. The importance of words in the image to the VQA task was first recognized in [19] which proposed a new benchmark TextVQA dataset. Hu, et. al. [10] proposed to use transformer [21] to express a more general form of attention between image, image objects, image texts and questions. Recently [24] introduced pre-training tasks to this model architecture that boosted the state of the art of TextVQA benchmark significantly.

Our work uses a multi-modality transformer architecture similar to the one proposed in [10]. This work is different from [10] in that we address the challenges in product attribute value extraction which do not exist in the text VQA task studied in [10]. We achieve this by enabling the model to query a dynamic external vocabulary which is conditioned on product category and also introduce multi-task training so the model is aware of product category.

3 PROBLEM DEFINITION

A product profile displayed on an e-commerce website usually looks like Figure 2. A navigation bar (top in Figure 2) describes the category that the product belongs to. Left side is product image

Figure 2: Example of product profiles.

and right side are product texts, including a title and several bullet points.

We consider products in a set of N product categories $C = \{c_1, \cdots, c_N\}$; a category can be coffee, skincare, as shown in Table 2. We formally define the problem as follows.

Problem definition: We take as input a target attribute *attr* and a product with the following information:

(1) a phrase describing the product category;
(2) the text in the product profile (*i.e.,* title, bullet points), denoted by a sequence of M words $T = \{w_1^{text}, \cdots, w_M^{text}\}$;
(3) the product image [1].

Our goal is to predict the value for the target attribute *attr*.

Figure 2 displayed a few such attribute values for a sunscreen product. Specifically, considering the target attribute *"Item Form"* shown in Figure 2, our objective is to extract the attribute value *"Spray"*. If the target attribute is *"Brand"*, the objective is to extract the attribute value *'Alba Botanica'*.

We next describe the model used to solve this problem. We first describe the model for the scenario of single product category (Section 4) and then describe its extension to the scenario of multiple categories (Section 5). The model and its extension are illustrated in Figure 3.

4 ATTRIBUTE VALUE EXTRACTION FOR A SINGLE PRODUCT CATEGORY

4.1 Overall Architecture: Sequence to Sequence generation model

As shown in Figure 3, the overall model architecture is a sequence-to-sequence generation model. The encoder and decoder are implemented with one transformer, denoted as "Multi-Model Transformer" in Figure 3, where attention masks are used to separate the encoder and decoder computations from each other internally.

[1] we use the first image shown on the website. For certain attributes such as nutrition information, later images such as nutrition label are used instead.

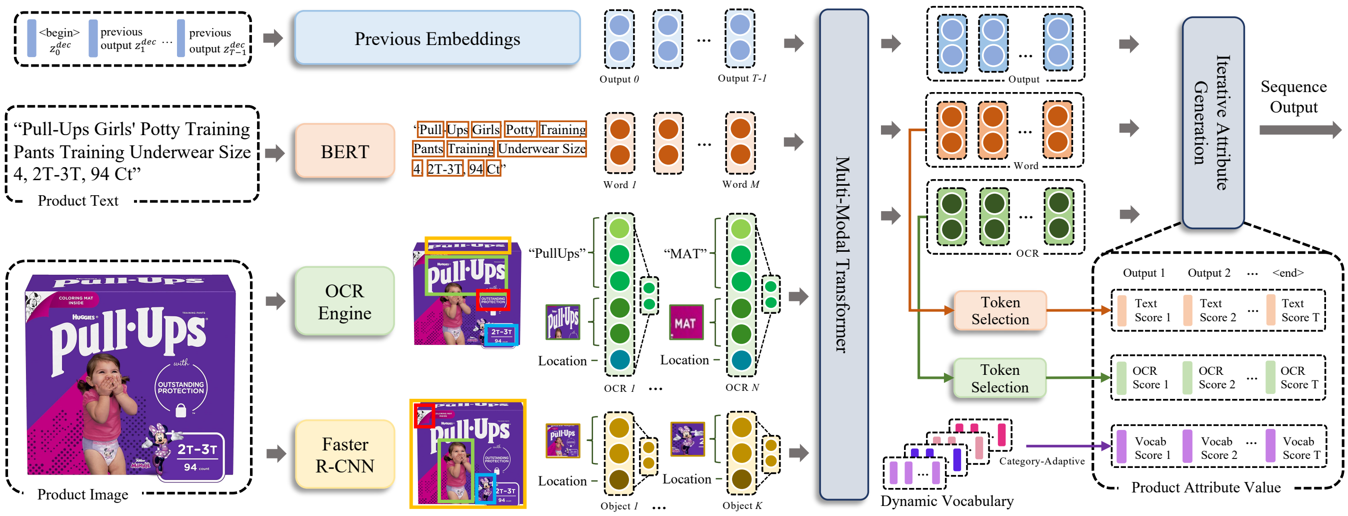

Figure 3: Overview of the proposed framework. 1) Input modalities: Product title and product image. 2) Token selection: Tokens could be selected from product title, OCR tokens identified in product image and a dynamic vocabulary conditioned on the product category. We consider edit distance between candidates and existing attribute values while selecting the next token. 3) Target sequence: We ask the decoder to first decode product category, and then decode attribute value.

The input from the different modalities and the previously decoded tokens are each converted into vector representations, the details can be found in Section 4.3. These representations are transformed into vectors of the same dimension, then concatenated as a single sequence of embeddings that is fed to the transformer. Therefore, an input position in each input modality is free to attend to other positions within the same modality, or positions from a different modality. The decoder operates recursively and outputs a vector representation z_t^{dec} at the tth step. The decoder output is based on the intermediate representations of the different encoder layers, along with the embeddings of the previously decoded tokens z_{t-1}^{dec} at step $0 \cdots t-1$, denoted by "Previous Embeddings" in Figure 3. A token selection module then chooses the token to output at step t based on z_t^{dec} from a candidate token set (Section 4.2).

4.2 Decoder Output Vocabulary and Token Selection

The traditional sequence-to-sequence generation model is known to suffer from text degeneration [9], in which the decoder outputs repetitive word sequences that are not well formed linguistically. To fix this problem, the output of the decoder in our model is constrained to a set of candidate words. At the tth decoding step, the decoder can return a token from the product text profile, the OCR engine, or from an external pre-defined vocabulary. The external vocabulary is composed of the words from the set of target attribute values, obtained from the training dataset. It is useful in the cases where *1)* the true attribute value is not mentioned in the product profile or image, *2)* the words in the image are not properly captured by the OCR algorithm, or *3)* the true attribute value is implied by the different inputs (from the product profile and images), but not explicitly mentioned. An example for the third case is predicting the target age of a "hair clips for young girl" product, where the target values include "kid", "teenage", or "adult" only.

The output token at the tth decoding step is selected based on z_t^{dec}, the output vector representation from the decoder at the tth step. z_t^{dec} is compared against the representation of every token in the candidate set, and every token is assigned a score using a cosine-similarity formula. The three rows in the rectangle "Product Attribute Value" on the right bottom corner in Figure 3 illustrate the tokens with the highest score within each of the three types of candidate tokens at T time steps. At any given time step, the token with the highest score among all candidate tokens is chosen and output by the decoder.

The scoring formulas depend on the source of the candidate token set, and are given by (1), (2) and (3) for tokens from external vocabulary, OCR, and text respectively.

$$y_{t,l}^{voc} = (w_l^{voc})^T z_t^{dec} + b_l^{voc} \tag{1}$$

$$y_{t,n}^{ocr} = (W^{ocr} z_n^{ocr} + b^{ocr})^T (W^{dec} z_t^{dec} + b^{dec}) \tag{2}$$

$$y_{t,m}^{text} = (W^{text} z_m^{text} + b^{text})^T (W^{dec} z_t^{dec} + b^{dec}) \tag{3}$$

$y_{t,l}^{voc}$, $y_{t,n}^{ocr}$ and $y_{t,m}^{text}$ are the score for the lth word in the vocabulary of frequent answer words, the nth OCR word, and the mth text token respectively in the tth decoding step. The token in the external vocabulary is represented by w_l^{voc} which is a trainable vector. z_n^{ocr} and z_m^{text} are the output embeddings for the OCR token and the text token respectively computed by the encoder. b_l^{voc}, W^{ocr}, b^{ocr}, W^{text}, b^{text}, W^{dec}, b^{dec} represent the trainable parameters of the linear transforms that align representations of candidate token and z_t^{dec}.

4.3 Representation of Inputs

Previously decoded tokens are converted into vector representations and fed to the decoder. If the token is selected from product texts or OCR, it is represented by the output embedding from the encoder. If the token is selected from the external vocabulary, w_l^{voc} in (1) is used as its representation.

The product profile texts are fed into the first three layers of a BERT model. The outputs of the pre-processing steps are then converted into three embeddings with the same dimension, as in [10] (see Section 6.3 for implementation details).

The input image is pre-processed with object detection and OCR recognition. For object detection, the product image is fed into the Faster R-CNN object detection model [18], which returns bounding boxes of detected objects, denoted as "Location" in Figure 3, and fixed length vector representation extracted through RoI-Pooling among each detected region.

The OCR engine provides the detected texts, along with their bounding boxes. We also extract visual features over each OCR token's region using the same Faster R-CNN detector. We experimented with two different OCR solutions: *1)* Public Amazon OCR API. [2] *2)* Mask TextSpotter [16] which is more capable of detecting texts that are organized in unconventional shapes (such as a circle).

5 PAM: PRODUCT-CATEGORY-AWARE MULTIMODAL ATTRIBUTE VALUE EXTRACTION MODEL

The model described in the previous section performs reasonably well if trained and inferenced on a single product category. However, if it is trained and inferenced on multiple product categories, its precision and recall cannot outperform existing baselines and is sometimes even inferior. After analyzing the errors, we found that this happens because the output of the sequence generation model is not conditioned on product category. In this section we describe a model design that is aware of the product category.

5.1 Dynamic Vocabulary

The vocabulary of target attribute values could contain very different words for different product categories. For example, it is unlikely for sunscreen brands to share common words with coffee brands except for words *"company"* or *"inc"*. Hence for each (product category, attribute type) pair, we pre-identify a specific vocabulary of frequent attribute value words, denoted by $V_{i,j}$ for the ith product category and jth attribute type. We also added the words that appear in the product category name to the vocabulary, we will clarify its role in Section 5.3. In addition, the word *"unknown"* is added to the vocabulary, so the model can output *"unknown"* when the input product profile does not contain a value for this attribute. The goal is to capture patterns for products where the attribute value is not usually conveyed. During the training process, the model will query the vocabulary $V_{i,j}$ according to the input product category i, which provides a more precise prior knowledge. Instead of training the representation from scratch as w_l^{voc} in (1), the vector representation for each word in $V_{i,j}$ is obtained with pre-trained fastText [4] embedding. This is because there are not enough training data in some product categories to compute a good word representation during the learning process.

5.2 Domain Specific Output Token Selection

The scoring function used for token selection, (2) and (3), are based on pre-trained embeddings. However, in our task the word could have domain-specific meaning. For example, *"all"* and *"off"* are considered stop words in English, but *"all"* is a popular brand of detergent and *"off"* is a popular brand of insect repellent. The lack of domain-specific embedding hurts the accuracy of the token selection module. We therefore utilize the edit distance between the candidates and existing attribute values as a supplemental feature. We denote the edit distance based similarity ratio [3] for specific word token w compared with the vocabulary $V_{i,j} = \{v_1, \cdots, v_L\}$ with L words as:

$$f^e(w) = \max_{l \in L} \ \text{similarity\_ratio}(w, v_l) \tag{4}$$

With the decoded embedding z_t^{dec} and the edit distance function f^e, we calculate the final score for candidates from the different modalities as follows, where (5), (6), and (7) are for tokens in the dynamic vocabulary $V_{i,j}$, OCR tokens, and tokens from product profile texts, respectively.

$$y_{t,l}^{voc} = (W^{voc} z_l^{voc} + b^{voc})^T (W^{dec} z_t^{dec} + b^{dec}) \tag{5}$$

$$y_{t,n}^{ocr} = (W^{ocr} z_n^{ocr} + b^{ocr})^T (W^{dec} z_t^{dec} + b^{dec}) + \lambda f^e(w_n^{ocr}) \tag{6}$$

$$y_{t,m}^{text} = (W^{text} z_m^{text} + b^{text})^T (W^{dec} z_t^{dec} + b^{dec}) + \lambda f^e(w_m^{text}) \tag{7}$$

If d denotes the dimension of the encoder's output embedding, then W^{text}, W^{ocr} and W^{dec} are $d \times d$ projection matrices, W^{voc} is a $d \times 300$ matrix (300 is the dimension of the fastText embeddings). z_m^{text}, z_n^{ocr} are the output embeddings for text tokens and OCR tokens computed by the encoder, respectively. z_l^{voc} is the fastText embedding of the frequent vocabulary words. b^{text}, b^{ocr}, b^{voc}, b^{dec} are all d-dimensional bias vectors and λ is the hyper-parameter to balance the score. Finally, the auto-regressive decoder will choose the candidate tokens with the highest score from the concatenated list $[y_{t,m}^{text},\ y_{t,n}^{ocr},\ y_{t,l}^{voc}]$ at each time step t.

5.3 Multi-Task Training

We use the multi-task learning setup to incorporate the product categories in the overall model. We experiment with two methods of multi-task training.

5.3.1 Embed the Product Category in Target Sequence. The first multi-task training method is based on prefixing the target sequence the decoder is expected to generate during training with the product category name. For example, the target sequence of product shown in Figure 2 would be *"sunscreen spray"* for attribute *"Item Form"* and *"sunscreen alba botanica"* for attribute *"Brand"*. The category name prefix serves as an auxiliary task that encourages the model to learn the correlation between the product category and attribute value. At inference time, the decoder output at the tth step depends on its previous outputs. But since the ground truth value of the

[2] https://aws.amazon.com/rekognition

[3] The FuzzyWuzzy ratio implemented in https://github.com/seatgeek/fuzzywuzzy

Table 2: Dataset Statistics

Category	# Samples	# Attr1	# Attr2
cereal	3056	7	631
dishwasher detergent	741	8	114
face shaping makeup	6077	16	926
fish	2517	11	391
herb	6592	19	1220
honey	1526	20	472
insect repellent	994	20	373
jerky	3482	9	475
sauce	4218	10	878
skin cleaning agent	10904	22	3016
skin foundation concealer	8564	17	744
sugar	1438	10	347
sunscreen	5480	26	1295
tea	5719	14	1204

product category is known, there is no need to depend on product category estimated by the decoder. We simply replace it with the true product category value. We have seen empirically that this modification improves the precision of the model.

Let the target label during the tth decoding step be $[y_{t,m}^{text},\ y_{t,n}^{ocr},\ y_{t,l}^{voc}]$, which takes the value 1 if the token from text, OCR, or external vocabulary is the correct token and 0 otherwise. More than one token could have label 1 if they are identical but from different sources. We use the multi label cross entropy loss between the target label list $[y_{t,m}^{text},\ y_{t,n}^{ocr},\ y_{t,l}^{voc}]$ and the predicted score list $[\hat{y}_{t,m}^{text},\ \hat{y}_{t,n}^{ocr},\ \hat{y}_{t,l}^{voc}]$ given by (5)-(7). The loss function hence contains two term: the loss contributed by the category name prefix, and the loss contributed by the attribute value in the target sequence:

$$Loss = Loss_{\text{attribute value}} + \lambda_{cat} Loss_{\text{category name prefix}} \quad (8)$$

where λ_{cat} is the tunable hyper-parameter to balance between these two losses.

5.3.2 Auxiliary Task of Category Prediction. The target sequence method is by no means the only possible design of multi-task training. It is also possible to introduce a separate classifier $f^{cat}(z)$ to predict the product category. A specific classification token <CLS> is inserted as the first entity in the input to the encoder. After concatenating and fusing with the other multimodal contexts in the transformer encoding layer, the enriched representation z^{cls} corresponding to the classification token <CLS> will be passed to the feed-forward neural network $f^{cat}(z)$ to predict the product category.

$$f^{cat}(z) = softmax(W^{cat} z^{cls} + b^{cat}) \quad (9)$$

where W^{cat} and b^{cat} are trainable parameters.

The training of the end-to-end model is jointly supervised by the sequence generation task and product category prediction tasks as described in (10).

$$Loss = Loss_{\text{attribute value}} + \lambda_{cat} Loss_{cat} \quad (10)$$

where $Loss_{cat}$ is the loss for product category prediction task.

Table 3: Hyper-parameters Details

Hyper-parameters	Value
batch size	128
iterations	24,000
optimizer	Adam
base learning rate	$1e^{-4}$
learning rate decay	0.1
learning rate decay steps	15,000, 20,000
max decoding steps	10
transformer layers	4
transformer attention heads	12
embedding dimension	768
lambda for edit distance	0.05

6 EXPERIMENTS SETUP AND RESULTS

6.1 Dataset

We evaluate our approach on 61,308 samples that cover 14 product categories. For each product category, we randomly collect the product texts, attribute values and images from the `amazon.com` web pages. We split the dataset into 56,843 samples as training/validation set and 4,465 samples as held-out testing set. The attribute values shown on the web page are used as training label after basic pre processing, which handle the symbol and morphology issues. The attribute values labeled by annotators are used as benchmark testing label. Assuming the attribute type is applicable to the products, if the attribute value information can not be observed from the given product profile (text description, image, OCR tokens), we will assign *"unknown"* as the corresponding value. In terms of the target attribute, we focus on two different types of attribute in the experiments. One of the criteria to determine the attribute type is the size of the value space. We consider attribute 1 *"Item Form"* with around 20 values as an attribute with closed vocabulary, and attribute 2 *"Brand"* with more than 100 values as an attribute with open vocabulary. Table 2 summarizes the statistics of our dataset, where "# Samples" denotes the number of samples, "# Attr1" denotes the number of unique values for attribute 1 *"Item Form"* and "# Attr2" denotes the number of unique values for attribute 2 *"Brand"*.

6.2 Evaluation Metrics

We use *Precision*, *Recall* and *F1* score as the evaluation metrics. We compute *Precision* (denoted as *P*) as percentage of "match" value generated by our framework; *Recall* (denoted as *R*) as percentage of ground truth value retrieved by our framework; *F1* score (denoted as $F1$) as harmonic mean of *Precision* and *Recall.* We determine whether the extraction result is a "match" using the exact match criteria, in which the full sequence of words are required to be correct.

6.3 Implementation Details

The multi-modal encoder / decoder module is a 4-layers, 12-heads transformer which is initialized from scratch. More hyper parameters are illustrated in Table 3. The dimension of the input representations are: *1)* 100 product title tokens, *2)* 10 image objects, *3)* 100 OCR tokens. These representations are computed as following:

6.3.1 *Embeddings for Text Modality.*

The product text tokens are projected into sequence of vectors as the word-level embeddings using the first three layers of a BERT model, whose parameters are fine-tuned in training.

6.3.2 *Embeddings for Image Modality.*

The external object detection component is the Faster R-CNN model [18], which has ResNet-101 [8] as backbone and is pre-trained on the Visual Genome dataset [14]. For each detected object, the Faster R-CNN model produces a fixed length vector which is a visual feature for the region of the object. In addition, the bounding box information of each object are also included as 4 coordinates. In order to combine and balance the energy of these two different types of features, projection fully-connected layer and the layer normalization are applied. During the training process, the final fully-connected layer of Faster R-CNN is fine-tuned.

6.3.3 *Embeddings for OCR Modality.*

The OCR modality conveys both textual features (*i.e.* characters) and visual features (*i.e.* color, font, appearance, location of the tokens). For textual features, the OCR texts are embedded using pre-trained fastText [4], which could handle the out of vocabulary (OOV) words and morphology of words properly. For robustness, additional character-level features are extracted by manually designed Pyramidal Histogram of Characters (PHOC) [1]. Similar to image embeddings, the pre-trained Faster R-CNN will represent the region of OCR tokens with its visual and coordinates features. Finally, linear projections are used to cast these features into the same length, which are then added together, passed through a layer normalization layer.

6.4 Baselines

To evaluate our proposed framework, we choose the following models as baselines: BiLSTM-CRF [11], OpenTag [26], BUTD [2] and M4C [10]. Our attribute value extraction task is highly related to the visual question answering tasks. Thus, among the four baselines, BiLSTM-CRF and OpenTag are attribute value extraction models, BUTD and M4C are visual question answering models. Some variants of our model and baselines are also included to make fair comparison on input sources. The details of baselines are listed below:

- BiLSTM-CRF [11]: the hidden states generated by the BiLSTM model are fed into the CRF as input features, the CRF will capture dependency between output tags. Text modality is used in this model.
- OpenTag [26]: on top of the BiLSTM-CRF, attention mechanism is introduced to highlight important information. Text modality is used in this model.
- BUTD [2]: Bottom-Up and Top-Down (BUTD) attention encodes question with GRU [5], then attends to object region of interest (ROI) features to predict answer. Text and Image modalities are used in this model.
- M4C [10]: cross-modality relationships are captured using multimodal transformer, the model will then generate the answer by iterative sequence decoding. Text, image and OCR modalities are used in this model. The answer could be selected from OCR tokens and frequent vocabulary.

Table 4: Comparison between the proposed framework PAM and different baselines

Attributes	Models	*P(%)*	*R(%)*	*F1(%)*
Item Form	BiLSTM-CRF	90.8	60.2	72.3
	OpenTag	95.5	59.8	73.5
	BUTD	83.3	53.7	65.3
	M4C	89.4	52.6	66.2
	M4C full	90.9	63.4	74.6
	PAM (ours) text-only	94.5	60.1	73.4
	PAM (ours)	91.3	75.3	82.5
Brand	BiLSTM-CRF	81.8	71.0	76.1
	OpenTag	82.3	72.9	77.3
	BUTD	79.7	62.6	70.1
	M4C	72.0	67.8	69.8
	M4C full	83.1	74.5	78.6
	PAM (ours) text-only	81.2	78.4	79.8
	PAM (ours)	86.6	83.5	85.1

- M4C full: to accommodate to the attribute value extraction task in e-commerce applications, extra input source of product title are added directly in the decoding process. Text, image and OCR modalities are used in this model. The answer could be selected from product title, OCR tokens and frequent vocabulary.
- PAM text-only: the text-only variant of our framework, image visual features and OCR tokens extracted from the product image are excluded from the input embeddings. Text modality is used in this model.

6.5 Results and Discussion

6.5.1 *Comparison between different model architectures.*

We conduct experiments on the dataset described in section 6.1. We first show the performance comparisons between our approach, baselines and some variants on two different attributes *"Item Form"* and *"Brand"* in Table 4. As can be seen from these comparison results, PAM could consistently outperform the other baseline methods on Recall and F1 score. For example, for the *"Item Form"* attribute, the Recall of PAM increases by 15% compared with the text-only variant of PAM and increases by 22% compared with the M4C model. For the *"Brand"* attribute, the Recall of PAM increases by 5% compared with the text-only variant of PAM and increases by 15% compared with the M4C model. Note that PAM could achieve higher score on all metrics compared with the M4C full variant (full access to all modalities in the decoding process), which demonstrate the effectiveness of our task-specific designs on the framework. There are two main reasons contribute to these improvements: *1)* PAM utilizes rich information from naturally fused text, image and OCR modalities that could significantly improve Recall. These three modalities could help each other by providing important cues while information might be missing in specific modality. *2)* PAM utilizes product category inputs in the decoding process. Attribute value is highly related to product category. By considering such crucial information, our model is able to learn enriched embeddings that could discriminate targeted attribute values from distracting values that belong to other product categories.

Table 5: Usefulness of Image, text, OCR inputs

Models	*P(%)*	*R(%)*	*F1(%)*
PAM *w/o* text	79.9	63.4	70.7
PAM *w/o* image	88.7	72.1	79.5
PAM *w/o* OCR	82.0	69.4	75.1
PAM	91.3	75.3	82.5

Table 6: Impact of different components in the model

Models	*P(%)*	*R(%)*	*F1(%)*
PAM *w/o* target sequence	88.5	72.9	80.0
PAM *w/o* dynamic vocabulary	89.1	69.5	78.1
PAM	91.3	75.3	82.5

6.5.2 Usefulness of image, text and OCR inputs. In order to quantify the impact of each modality, we further conduct ablation study on the input sources. We evaluate following variants of PAM:

- PAM *w/o* text is the variant that removes product texts modality from inputs.
- PAM *w/o* image is the variant where features of detected objects are removed from inputs.
- PAM *w/o* OCR is the variant that removes the OCR tokens from inputs.

From Table 5 we can see that all the metrics on the attribute *'Item Form'* degrade by removing any modality from the PAM framework, which demonstrates the necessity of combining all the modalities in our attribute extraction task. Closer inspection on the table shows that the text modality plays the most important role in this task. On the other hand, the image modality which represents the appearance of the product might be less effective compared to the other two modalities. The first possible reason is that the image could contain noisy information. In addition, similar shape of product might have different semantic meanings among various product categories. Finally, different attribute types also affect the performance, image modality could contributes more if the attribute type is related to color or obvious shape.

6.5.3 Impact of different components in the model. We also conduct experiments by removing each individual model design component from our framework to evaluate its effectiveness. The variants are listed below:

- PAM *w/o* target sequence is the variant that will generate attribute value without first generating the product category name.
- PAM *w/o* dynamic vocabulary is the variant that uses a large vocabulary of words shared by multiple categories instead of a dynamic vocabulary conditioned on product category.

Table 6 presents the extraction results on the *"Item Form"* attribute. Without the category specific vocabulary set, the model has to search on a much larger space for possible attribute values. The target sequence could enforce the category information via back-propagation loss. It is apparent from the results that these two category modules contribute to the final gains on Recall/F1 score.

Table 7: Comparison of multi-task training

Models	*P(%)*	*R(%)*	*F1(%)*
Auxiliary Task of Category Prediction	90.8	75.4	82.3
Embed Product Category in Target Sequence	91.3	75.3	82.5

Table 8: Impact of OCR component on model performance

OCR Detectors	average # OCR tokens extracted	*F1(%)*
Mask TextSpotter	13	71.1
Amazon Rekognition	42	80.3

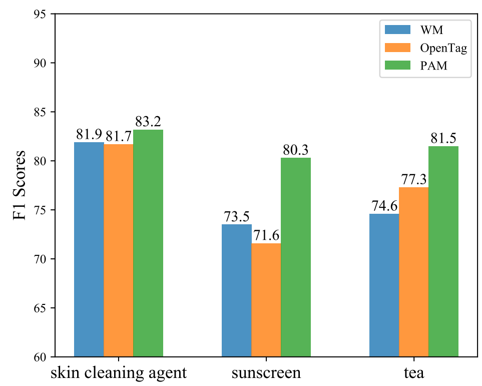

Figure 4: Comparison between different methods on a single product category.

6.5.4 Comparison of Different Multi-Task Training Methods. The performance of the two multi-task training methods in Section 5.3 are compared in Table 7 for the *'Item Form'* attribute. Overall, these two methods demonstrate similar performance.

6.5.5 Comparison with Baseline on a Single Product Category. Our approach is able to accommodate various product categories in one model. In order to verify such generalization ability on single category, we perform the category-level individual training tasks using the following baselines:

- OpenTag[26]: the setting is the same as described in 6.4, except the training and evaluation are performed on single product category.
- Word Matching (WM): this is a brute-force matching baseline. *1)* all the possible attribute values will be collected as attribute value dictionary ({*"value"* : *count*}), the count represents the popularity of corresponding attribute value; *2)* manually exclude some distracting words like "tea" from the dictionary of the "tea" product category; *3)* extract title tokens and OCR tokens for each sample; *4)* compare the extracted tokens with attribute value dictionary in a popularity ascending sequence; *5)* identify the attribute value if exact match is found in the attribute value dictionary. This baseline method does not require training, it is evaluated on the testing data directly.

For the purpose of performing category-level individual training, we choose three categories that contain enough number of samples: *skin cleaning agent*, *sunscreen* and *tea*. Figure 4 demonstrates the comparison of two baselines and our model on single category. Our method consistently improves the extraction metric. Although the WM baseline could also produce a good F1 score for *"skin cleaning agent"*, it requires manual efforts to create a good list to exclude words that are impossible to appear in attribute values, which is expensive to scale up to many product categories.

Under the single category settings, we also implement experiments to evaluate the impact of the external OCR components on end-to-end performance. As introduced in Section 4, we use Amazon Rekognition and Mask TextSpotter to extract OCR tokens from the product image. $F1(\%)$ is the extraction performance on attribute *Item Form* and category *Sunscreen* of using corresponding detectors. It can be seen from Table 8 that Rekognition is more suitable for our task. This is because Mask TextSpotter is trained on a public dataset that is different from the product image dataset. Therefore, Rekognition on average identifies more OCR tokens and hence lead to better end-to-end $F1$ scores.

7 CONCLUSIONS

To sum up, we explored a multimodal learning task that involves textual, visual and image text collected from product profiles. We presented a unified framework for the multimodal attribute value extraction task in the e-commerce domain. Multimodal transformer based encoder and decoder are used in the framework. The model is trained to simultaneously predict product category and attribute value and its output vocabulary is conditioned on the product category as well, resulting in a model capable of extracting attributes across different product categories. Extensive experiments are implemented on a multi categories/multi attributes dataset collected from public web page. The experimental results demonstrate both the rich information contained within the image/OCR modality and the effectiveness of our product category aware multimodal framework.

For future works, pre-training task from [24] could be useful in our attribute value extraction scenario. It is also valuable to scale from 14 product categories to thousands of product categories and model the complex tree structure of product categories properly [12]. The dynamic selection of vocabulary in this framework could be incorporated into the training process as in the RAG architecture [15]. Finally, it is useful to design a model that extracts different attributes with one model in which case the attribute generation order could be part of the learning process too [7].

ACKNOWLEDGMENTS

Rongmei Lin and Li Xiong are partially supported by NSF under CNS-1952192, IIS-1838200.

REFERENCES

[1] Jon Almazán, Albert Gordo, Alicia Fornés, and Ernest Valveny. 2014. Word spotting and recognition with embedded attributes. *IEEE transactions on pattern analysis and machine intelligence* 36, 12 (2014), 2552–2566.

[2] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 6077–6086.

[3] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision.* 2425–2433.

[4] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146.

[5] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2015. Gated feedback recurrent neural networks. In *International conference on machine learning*. PMLR, 2067–2075.

[6] Xin Luna Dong, Xiang He, Andrey Kan, Xian Li, Yan Liang, Jun Ma, Yifan Ethan Xu, Chenwei Zhang, Tong Zhao, Gabriel Blanco Saldana, et al. 2020. AutoKnow: Self-Driving Knowledge Collection for Products of Thousands of Types. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2724–2734.

[7] Dmitrii Emelianenko, Elena Voita, and Pavel Serdyukov. 2019. Sequence Modeling with Unconstrained Generation Order. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc., 7700–7711. https://proceedings.neurips.cc/paper/2019/file/1558417b096b5d8e7cbe0183ea9cbf26-Paper.pdf

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 770–778.

[9] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The Curious Case of Neural Text Degeneration. In *International Conference on Learning Representations*.

[10] Ronghang Hu, Amanpreet Singh, Trevor Darrell, and Marcus Rohrbach. 2020. Iterative answer prediction with pointer-augmented multimodal transformers for textvqa. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9992–10002.

[11] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991* (2015).

[12] Giannis Karamanolakis, Jun Ma, and Xin Luna Dong. 2020. Txtract: Taxonomy-aware knowledge extraction for thousands of product categories. *arXiv preprint arXiv:2004.13852* (2020).

[13] Jin-Hwa Kim, Jaehyun Jun, and Byoung-Tak Zhang. 2018. Bilinear attention networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 1571–1581.

[14] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision* 123, 1 (2017), 32–73.

[15] Patrick Lewis, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. arXiv:2005.11401 [cs.CL]

[16] Minghui Liao, Guan Pang, Jing Huang, Tal Hassner, and Xiang Bai. 2020. Mask TextSpotter v3: Segmentation Proposal Network for Robust Scene Text Spotting. In *Proceedings of the European Conference on Computer Vision (ECCV).*

[17] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *arXiv preprint arXiv:1908.02265* (2019).

[18] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2016. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence* 39, 6 (2016), 1137–1149.

[19] Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. 2019. Towards vqa models that can read. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8317–8326.

[20] Hao Tan and Mohit Bansal. 2019. Lxmert: Learning cross-modality encoder representations from transformers. *arXiv preprint arXiv:1908.07490* (2019).

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* (2017).

[22] Qifan Wang, Li Yang, Bhargav Kanagal, Sumit Sanghai, D Sivakumar, Bin Shu, Zac Yu, and Jon Elsas. 2020. Learning to Extract Attribute Value from Product via Question Answering: A Multi-task Approach. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 47–55.

[23] Huimin Xu, Wenting Wang, Xinnian Mao, Xinyu Jiang, and Man Lan. 2019. Scaling up open tagging from tens to thousands: Comprehension empowered attribute value extraction from product title. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 5214–5223.

[24] Zhengyuan Yang, Yijuan Lu, Jianfeng Wang, Xi Yin, Dinei Florencio, Lijuan Wang, Cha Zhang, Lei Zhang, and Jiebo Luo. 2020. TAP: Text-Aware Pre-training for Text-VQA and Text-Caption. arXiv:2012.04638 [cs.CV]

[25] Pengchuan Zhang, Xiujun Li, Xiaowei Hu, Jianwei Yang, Lei Zhang, Lijuan Wang, Yejin Choi, and Jianfeng Gao. 2021. VinVL: Making Visual Representations Matter in Vision-Language Models. arXiv:2101.00529 [cs.CV]

[26] 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*. 1049–1058.

[27] 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. *arXiv preprint arXiv:2009.07162* (2020).