MMT4: Multi Modality To Text Transfer Transformer

Amir Tavanaei		Karim Bouyarmane
atavanae@amazon.com	bouykari@amazon.com
Amazon	Amazon
Seattle, WA, USA	Seattle, WA, USA
Iman Keivanloo		Ismail Tutar
imankei@amazon.com	ismailt@amazon.com
Amazon	Amazon
Seattle, WA, USA	Seattle, WA, USA

ABSTRACT
Recent studies have demonstrated the ability of auto-regressive and seq-to-seq generative models to reach state-of-the-art performance on various Natural Language Understanding (NLU) and Natural Language Processing (NLP) tasks. They operate by framing all the tasks in a single formulation: text auto-completion or text-to-text encoding-decoding. These models can be trained on the products corpus in order to understand the information in the e-commerce products listings. In this paper, we present a new generative model to involve different modalities (e.g. text and vision). The proposed model is an encoder-decoder model with the T5 (Text To Text Transfer Transformer) foundation in which the non-text components are fused to the text tokens. Specific relative positional and token type embeddings are used in the encoder part, while the decoder generates new text corresponding to diverse tasks. Hence, we name the proposed model MMT4: Multi Modality To Text Transfer Transformer. The experiments are done over our proprietary e-commerce catalog involving image and text, with the rationale that the image of a product provides more information about the product. One of the main advantages of this model is to generate product attributes (product specifications) that can be either solely inferred from the text or the image, or both. In the experiments, we pre-train and fine-tune MMT4 to solve a number of downstream tasks: attribute generation, image-text matching (ITM), and title (product name) generation from product’s image (captioning). The experimental results show up to 35% accuracy improvement in comparison with the fine-tuned T5 in the attribute generation task. Product title generation also shows more than 3% higher Rouge-1 recall than the fine-tuned state-of-the-art captioning model. Although we fine-tuned our model on less than 2M samples in a generative mode, its performance is only 2% area under the precision-recall curve lower than the state-of-the-art ITM model.

CCS CONCEPTS
• Applied computing → Electronic commerce; • Information systems → Information systems applications.

KEYWORDS
Multimodal transformers, encoder-decoder, natural language generation

ACM Reference Format:
ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION
Following the success of transformers in encoding the natural language data and text representation [5, 13, 19, 32], recent research focused on using the pre-trained text encoders in downstream tasks such as sentence encoding [26], classification, semantic analytics [23, 26], question answering [28], and named entity recognition [30]. For each task, separate fine-tuning and model customization are required to provide task-specific models. Additionally, in classification tasks, the number of classes is pre-defined and the model should be re-trained after adding a new label to the task. To address these concerns, sequence-to-sequence transformers [15, 18, 25, 33] provide a unique framework to support multi-task learning and additional class labels by generating text using their decoder component. T5 (Text To Text Transfer Transformer) [25] is one of the popular generative models that offers a unified architecture across multiple tasks and has shown great performances in different applications [2, 7, 10, 21].

Encoder-based transformers have been customized to encode other modalities and use the self-attention mechanism [32] in multimodal frameworks. There are a series of multimodal transformers in the literature which perform early/late fusion in vision and language modeling such as ViLBERT [20], ViLT [12], MMBT [11], and ALBEF [16] or two-tower vision and language models such as CLIP [24], and ALIGN [9]. Data2Vec [1] is the most recent transformer that encodes text, image, and speech. It is pre-trained using mask prediction and latent target representation using teacher signal to learn individual representations of the modalities and it does not perform multimodal training. Although these multimodal networks performed well in learning from joint or individual modalities, they inherit the same limitations in multi-task learning and

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SIG-KDD, Workshop on Content Understanding and Generation for E-commerce, August 14–18, 2022, Washington, DC
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06 ... $15.00
https://doi.org/XXXXXXX.XXXXXXX
We name our model Multi Modality To Text Transfer Transformer (MMT4). The proposed approach in MMT4, on the other hand, does not need self-supervised learning approaches and pre-trains the model using self-supervised learning approaches for visual embedding preparation. This network limits the inputs to only text and image. The token type (segment) embedding (SE) in this architecture gets a unique token type id (in this figure 0 and 1) corresponding to diverse tasks specified by keywords (prompts).

Figure 1: T5 versus MMT4 facing complete and incomplete product titles to generate product attributes. The MMT4 uses image information to compensate information lack in the text. ‘?’ means that the model fails to predict correct response.

Figure 2: Title generation and image-title/text matching decision using MMT4.

In summary, our contributions in this work are to build a general sequence-to-sequence generative model architecture that can support different kinds of modalities, provide an efficient multitask, multimodal framework to improve the text-only generative models, and address a number of problems in e-commerce using the proposed model (MMT4).

2  MMT4 MODEL ARCHITECTURE

MMT4 is a customized and augmented version of T5 [25] where the input is not limited to text and can be chosen from different modalities. Fig. 3 shows the model architecture where the input is the combination of an input text and another source of input (e.g. corresponding image to the text). The “Modality Feature Representator” pre-processes the non-text input and extracts a sequence of feature vectors representing the new modality. For instance, the image input can be divided into image patches flattened to $D_p$-dimensional feature vectors, or a voice signal can be represented by a number of Mel Frequency Cepstral Coefficient (MFCC) [22] feature vectors. This component can also be a neural network architecture processing the raw input. For example, a vision transformer or convolutional neural network (CNN) can extract sequential feature vectors from images. The “Projection” component adjusts the modality feature vector dimension to the T5 hidden layer dimension (e.g. 768 for T5-base) using a linear layer followed by a normalization layer (LayerNorm). At this point, the new modality’s inputs are ready to be passed to the embedding and transformer layers next to the embedded text tokens. Eq. 1 shows the modality component ($P_i$) obtained by the modality (V) feature representator ($f_v$) that is projected by a linear NN ($W_{proj}$) followed by normalization, $y$.

$$P_i = f_v(V)_i , \quad I_i = y(W_{proj}P_i)$$ (1)

The token type (segment) embedding (SE) in this architecture separates different modalities and helps the model distinguish information flow from different input segments. Each modality in this architecture gets a unique token type id (in this figure 0 and 1) as shown in Eq 2. The embedded inputs for the text ($X^t$) and the other modality ($X^o$) are concatenated to prepare a single sequence.
where channel vectors passed to the decoder as key-value for the cross-attention. Image to squared patches (e.g. 16x16 or 32x32) as input feature. [6] trains a transformer encoder on ImageNet [4] by dividing the image component features, $P_i$, as documented in Fig. 4.1.

### Vision Transformer (ViT)
The Vision Transformer (ViT) model [6] trains a transformer encoder on ImageNet [4] by dividing the image into squared patches (e.g. 16x16 or 32x32) as input feature vectors. ViT has outperformed the CNN models in image classification tasks and has been widely used for visual and language transformers [12, 16, 24]. In this study, we use the ViT pre-trained on 224x224 images that are divided into 32x32 patches. This model represents an image by $m+1$ feature vectors where $m$ is the number of patches and $1$ is for the [CLS] token in this architecture as shown in Fig. 4.2.

#### Convolutional Neural Networks:
CNN as a well known NN architecture for image processing [8, 14] has shown great success in cooperating with the multi-modal transformers for encoding visual and natural language data [11, 16]. In this study, we use EfficientNet-B4 [31] as a high performance and efficient CNN model to extract visual features of images before fusing them to the text token features in the encoder. The visual features are acquired from the last convolutional layer of the EfficientNet-B4 in which each point across 1792 feature maps represent one image component. Fig. 4.3 depicts the image components extracted by the EfficientNet-B4. Given an image with the size of [380x380], the last convolutional layer’s output consists of 1792 feature maps with the size of [12x12]. Thus, this image feature representaor extracts 144 features vectors with the dimension of 1792.

### 4 EXPERIMENTS AND RESULTS
The modalities in the experiments are product’s title and image where the image modality is represented by the three approaches mentioned above. The training dataset includes 1.92M titles and their corresponding images and the validation dataset includes 7720 titles and corresponding images from different products than the training dataset. Pre-training and fine-tuning tasks are performed using the same training hyper-parameters with learning rate=2e-4 decayed linearly, training epochs=2, and batch size=288. This section describes the experiments and results of the pre-training and fine-tuning tasks.

#### 4.1 Pre-training
Pre-training of MMT4 involves self-supervised span prediction and masked language modeling (MLM). The span prediction replaces several, random token spans by special tokens and the MLM replaces random tokens by a mask token. TS tokenizer provides 100 extra special tokens names <extra-token-0> to <extra-token-99> where 99 of those are used for the span prediction task and <extra-token-99> is used as the [MASK] token in MLM. Fig. 5 shows an example of the pre-training data and expected generated texts. In the span prediction task, the noise density is 0.2 and the average span length is 2 tokens. The MLM mask density is 0.3. In both tasks, the cross entropy loss (ce) of the generated text and the target text is calculated and equally weighted for pre-training. Eq. 4 shows the calculation in the decoder layers. The "Linear Head" of the decoder layers including multi-head self-attention layers. The last hidden encoder blocks.

$$X_i^p = I_i + SE(0)$$

$$a_{ij} = \text{RPE}(\text{head}, L_i, L_j), \quad a_{ij} = \text{RPE}(\text{head}, L_i, L_j)$$

$$a_{ij} = \text{concat}(a_{ij}^p, a_{ij})$$

The embedded features are concatenated and fed to the encoder layers including multi-head self-attention layers. The last hidden state of the encoder is a sequence of $(L_v+L_d)D_h$-dimensional feature vectors passed to the decoder as key-value for the cross-attention calculation in the decoder layers. The “Linear Head” of the decoder section maps the decoder’s last hidden state to token IDs based on the casual language modeling masking. In this study, we only focus on the text generation, thus, the “Linear Head” is used in all the experiments. In the case that the other modality generation is expected, the “Linear Head” may or may not be used according to the modality type and the task.

### EXPERIMENTAL METHOD: IMAGE+TEXT
In this paper, we focus on image as the additional modality so that an encoder input involves a product’s title and its corresponding image. In this study we experiment three different image feature representation approaches as explained in the following paragraphs.

**Image Patching:** The image patching component divides the image to squared patches with the shape of $[\text{patch} \times \text{patch} \times \text{channel}]$ where $\text{channel} = 3$ (RGB channels). The flattened patches represent the image component features, $P_i$, as demonstrated in Fig. 4.1.

**Vision Transformer (ViT):** The Vision Transformer (ViT) model [6] trains a transformer encoder on ImageNet [4] by dividing the image to squared patches (e.g. 16x16 or 32x32) as input feature vectors. ViT has outperformed the CNN models in image classification tasks and has been widely used for visual and language transformers [12, 16, 24]. In this study, we use the ViT pre-trained on 224x224 images that are divided into 32x32 patches. This model represents an image by $m+1$ feature vectors where $m$ is the number of patches and $1$ is for the [CLS] token in this architecture as shown in Fig. 4.2.

**Convolutional Neural Networks:** CNN as a well known NN architecture for image processing [8, 14] has shown great success in cooperating with the multi-modal transformers for encoding visual and natural language data [11, 16]. In this study, we use EfficientNet-B4 [31] as a high performance and efficient CNN model to extract visual features of images before fusing them to the text token features in the encoder. The visual features are acquired from the last convolutional layer of the EfficientNet-B4 in which each point across 1792 feature maps represent one image component. Fig. 4.3 depicts the image components extracted by the EfficientNet-B4. Given an image with the size of [380x380], the last convolutional layer’s output consists of 1792 feature maps with the size of [12x12]. Thus, this image feature representaor extracts 144 features vectors with the dimension of 1792.

### EXPERIMENTAL METHOD: IMAGE+TEXT
In this paper, we focus on image as the additional modality so that an encoder input involves a product’s title and its corresponding image. In this study we experiment three different image feature representation approaches as explained in the following paragraphs.

**Image Patching:** The image patching component divides the image to squared patches with the shape of $[\text{patch} \times \text{patch} \times \text{channel}]$ where $\text{channel} = 3$ (RGB channels). The flattened patches represent the image component features, $P_i$, as demonstrated in Fig. 4.1.

**Vision Transformer (ViT):** The Vision Transformer (ViT) model [6] trains a transformer encoder on ImageNet [4] by dividing the image to squared patches (e.g. 16x16 or 32x32) as input feature vectors. ViT has outperformed the CNN models in image classification tasks and has been widely used for visual and language transformers [12, 16, 24]. In this study, we use the ViT pre-trained on 224x224 images that are divided into 32x32 patches. This model represents an image by $m+1$ feature vectors where $m$ is the number of patches and $1$ is for the [CLS] token in this architecture as shown in Fig. 4.2.

**Convolutional Neural Networks:** CNN as a well known NN architecture for image processing [8, 14] has shown great success in cooperating with the multi-modal transformers for encoding visual and natural language data [11, 16]. In this study, we use EfficientNet-B4 [31] as a high performance and efficient CNN model to extract visual features of images before fusing them to the text token features in the encoder. The visual features are acquired from the last convolutional layer of the EfficientNet-B4 in which each point across 1792 feature maps represent one image component. Fig. 4.3 depicts the image components extracted by the EfficientNet-B4. Given an image with the size of [380x380], the last convolutional layer’s output consists of 1792 feature maps with the size of [12x12]. Thus, this image feature representaor extracts 144 features vectors with the dimension of 1792.

### EXPERIMENTAL METHOD: IMAGE+TEXT
In this paper, we focus on image as the additional modality so that an encoder input involves a product’s title and its corresponding image. In this study we experiment three different image feature representation approaches as explained in the following paragraphs.

**Image Patching:** The image patching component divides the image to squared patches with the shape of $[\text{patch} \times \text{patch} \times \text{channel}]$ where $\text{channel} = 3$ (RGB channels). The flattened patches represent the image component features, $P_i$, as demonstrated in Fig. 4.1.

**Vision Transformer (ViT):** The Vision Transformer (ViT) model [6] trains a transformer encoder on ImageNet [4] by dividing the image to squared patches (e.g. 16x16 or 32x32) as input feature vectors. ViT has outperformed the CNN models in image classification tasks and has been widely used for visual and language transformers [12, 16, 24]. In this study, we use the ViT pre-trained on 224x224 images that are divided into 32x32 patches. This model represents an image by $m+1$ feature vectors where $m$ is the number of patches and $1$ is for the [CLS] token in this architecture as shown in Fig. 4.2.

**Convolutional Neural Networks:** CNN as a well known NN architecture for image processing [8, 14] has shown great success in cooperating with the multi-modal transformers for encoding visual and natural language data [11, 16]. In this study, we use EfficientNet-B4 [31] as a high performance and efficient CNN model to extract visual features of images before fusing them to the text token features in the encoder. The visual features are acquired from the last convolutional layer of the EfficientNet-B4 in which each point across 1792 feature maps represent one image component. Fig. 4.3 depicts the image components extracted by the EfficientNet-B4. Given an image with the size of [380x380], the last convolutional layer’s output consists of 1792 feature maps with the size of [12x12]. Thus, this image feature representaor extracts 144 features vectors with the dimension of 1792.
The accuracy of the number-of-items generation is greater than 99% because most of the values are 1. If we only take the number-of-items=1 into account, our model is 31% (12 out of 39) matched with the labels.

To evaluate the fine-tuned MMT4, the validation dataset including 7700 samples is used. As shown in Table 1, in average, using ViT as image representation slightly outperforms the other methods.

These attributes are the examples of attributes that may be discovered from the image if the attribute is not in the text input (title). However, the other modalities (e.g. image of the product) can solve this problem. Additionally, even if the textual data include required target tokens, additional modality can improve the model’s performance and correct the attribute values. To assess the impact of the image in information generation, we pre-process the dataset according to the task and remove the tokens of interest from the input data.

The attributes selected for the attribute generation task are color, brand, style, material, sleeve-type, number-of-items, and pattern. These attributes are the examples of attributes that may be discovered from the image if the attribute is not in the text input (title). The number of samples for each task may be different from other tasks because not all attributes are valid for all the products (for instance, shoes do not have sleeve-type).

The MMT4 is fine-tuned to generate the attributes mentioned above given the modified titles followed by task prompts (e.g. color:). To evaluate the fine-tuned MMT4, the validation dataset including 7700 samples is used. As shown in Table 1, in average, using ViT as image representation slightly outperforms the other methods.

### 4.2 Fine-tuning: Attribute Generation

The aim of MMT4 is mainly to improve the performance of downstream tasks and to provide a new network architecture to address questions that cannot be solved by text-only models. One of the main concerns in attribute generation/extraction from the product title (and description) is the lack of relevant information pinpointing the attributes. For instance, generating the “sleeve-type” attributes from a shirt’s title (and other descriptions) that does not reveal any information about the sleeve type of the shirt is almost impossible. However, the other modalities (e.g. image of the product) can solve this problem.

The MMT4 is fine-tuned to generate the attributes mentioned above given the modified titles followed by task prompts (e.g. color:). To evaluate the fine-tuned MMT4, the validation dataset including 7700 samples is used. As shown in Table 1, in average, using ViT as image representation slightly outperforms the other methods. The accuracy of the number-of-items generation is greater than 99% because most of the values are 1. If we only take the number-of-items=1 into account, our model is 31% (12 out of 39) matched with the labels.

### 4.3 Fine-tuning: Title Generation from Image

To evaluate MMT4 for title generation from image, we used 6819 product images from the validation dataset. The generated titles are

![Figure 6: Loss value trend during training. Left: all the steps. Right: steps 1000-12000.](image)

### Table 1: Attribute generation performance of MMT4 based on incomplete titles.

<table>
<thead>
<tr>
<th>Model</th>
<th>Color</th>
<th>Material</th>
<th>Pattern</th>
<th>Style</th>
<th>Brand</th>
<th>Sleeve</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMT4 (Image Patch)</td>
<td>78.00</td>
<td>71.54</td>
<td>63.37</td>
<td>47.78</td>
<td>30.32</td>
<td>81.94</td>
<td>94.89</td>
</tr>
<tr>
<td>MMT4 (EffNet)</td>
<td>74.75</td>
<td>72.61</td>
<td>63.38</td>
<td>51.45</td>
<td>29.59</td>
<td>85.17</td>
<td>95.53</td>
</tr>
<tr>
<td>MMT4 (ViT)</td>
<td>79.92</td>
<td>71.20</td>
<td>69.43</td>
<td>50.87</td>
<td>30.15</td>
<td>86.34</td>
<td>95.53</td>
</tr>
</tbody>
</table>

| Number of samples      | 3018  | 2337     | 314     | 519   | 6518  | 681    | 626   |

![Figure 7: MMT4’s impact in generating relevant attributes by relying on both image and title. This figure shows examples for color, pattern, and sleeve-type attribute generation.](image)
Table 2: Fine-tuned T5 versus fine-tuned MMT4 for attribute generation. First, all the titles are pre-processed (i.e. the attribute values in the titles are removed). Second, 20% of titles are pre-processed. Third, Original titles are used.

<table>
<thead>
<tr>
<th>Titles Pre-processed</th>
<th>Model</th>
<th>Attributes (Evaluation Accuracy)%</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Color</td>
<td>Sleeve_type</td>
</tr>
<tr>
<td>All of the titles</td>
<td>T5</td>
<td>39.5</td>
<td>78.9</td>
</tr>
<tr>
<td></td>
<td>MMT4</td>
<td>74.8</td>
<td>85.2</td>
</tr>
<tr>
<td>20% of the titles</td>
<td>T5</td>
<td>75.9</td>
<td>85.8</td>
</tr>
<tr>
<td></td>
<td>MMT4</td>
<td>87.3</td>
<td>88</td>
</tr>
<tr>
<td>No pre-processing</td>
<td>T5</td>
<td>84.2</td>
<td>87.7</td>
</tr>
<tr>
<td></td>
<td>MMT4</td>
<td>89.8</td>
<td>88.1</td>
</tr>
</tbody>
</table>

Figure 8: Material, style, and brand attribute generation examples in which MMT4 uses visual features to outperform T5. The visual features of the product image help MMT4 to pick the right value for the attributes.

compared with the reference titles using the Rouge-1 metrics [17]. Rouge-1 shows what percentage of unigrams are in both generated and reference texts. The MMT4 with EfficientNet and ViT image representors showed 18% Rouge-1 recall. It means, in average, 18% of the words in each title is generated by our model. We compared the generated titles by MMT4 with the generated titles by the state-of-the-art captioning model, X-VLM [34]. The zero-shot X-VLM reported 13.3% and the fine-tuned (using the same data as MMT4) X-VLM reported 14.9% Rouge-1 recall. Fig. 9 shows some examples of the reference and generated titles. As shown in this figure, the type of shirt (t-shirt or tank top), item type (curtain, shows, shirt, pants, ...), phone model (Samsung, iPhone), and other attributes such as color and pattern are well addressed in the generated titles.

4.4 Fine-tuning: Image-Text Matching

Another application of the multimodal generative model is classification in generative mode. That is, the output class name is generated instead of being selected from a list of classes. In this task, the input data includes the product image and title followed by “match” prompt. The output in this task is either “yes” or “no”. The performance of MMT4 for image-text matching of 6819 validation data is shown in Table 4. The confusion matrix, precision-recall curve, and ROC of this test are shown in Fig. 10. As shown in Table 4, CLIP outperforms MMT4. This is mostly because of different training process where CLIP is pre-trained on 400 million image-text pairs using metric learning while MMT4 is only trained on 1.9 M samples in a multi-task generative schema. Thus, only 2% PR-AUC performance drop is a green light showing that MMT4 can be applied to classification tasks in multi-task problems. Additionally, using task-specific discriminators for classification is limited to a pre-defined set of classes and cannot extract out-of-box information from the input whereas MMT4 can generate new words that well explain the input data from multiple modalities.

5 CONCLUSION

This paper introduces a new multimodal generative model named MMT4 to generate text given different modalities as input. Involving more than one modality in catalog data processing improves the downstream tasks performance by providing more information about the product. The experimental results of attribute generation showed that MMT4 outperforms the text-only model (T5) in attribute generation, given that the attribute values do not always exist in the text. Thus, the results of this application can be used for
Table 3: Examples of attribute values generated by MMT4 and T5. The “Labels” for each attribute shows the number of valid values in the ground truth for that specific attribute. Data size shows the number of products with the corresponding attribute value in the ground truth. [Brand names are hidden for the sake of privacy].

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Attribute Value</th>
<th>Labels</th>
<th>Data Size</th>
<th>T5 TP</th>
<th>MMT4 TP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleeve-Type</td>
<td>Long Sleeve</td>
<td>220</td>
<td>172</td>
<td>206</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Short Sleeve</td>
<td>5</td>
<td>344</td>
<td>320</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sleeveless</td>
<td>53</td>
<td>391</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td>Color</td>
<td>Red</td>
<td>187</td>
<td>27</td>
<td>159</td>
<td></td>
</tr>
<tr>
<td></td>
<td>White</td>
<td>514</td>
<td>210</td>
<td>369</td>
<td></td>
</tr>
<tr>
<td>Pattern</td>
<td>Striped</td>
<td>31</td>
<td>1</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Solid</td>
<td>73</td>
<td>74</td>
<td>63</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Print</td>
<td>88</td>
<td>82</td>
<td>81</td>
<td></td>
</tr>
<tr>
<td>Material</td>
<td>Wood</td>
<td>53</td>
<td>28</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ceramic</td>
<td>230</td>
<td>71</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cotton</td>
<td>373</td>
<td>277</td>
<td>305</td>
<td></td>
</tr>
<tr>
<td>Style</td>
<td>Classic</td>
<td>45</td>
<td>28</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modern</td>
<td>102</td>
<td>96</td>
<td>68</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Art Deco</td>
<td>35</td>
<td>28</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>Brand</td>
<td>P***</td>
<td>51</td>
<td>7</td>
<td>33</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S***</td>
<td>58</td>
<td>58</td>
<td>58</td>
<td></td>
</tr>
<tr>
<td></td>
<td>H***</td>
<td>45</td>
<td>45</td>
<td>45</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Performance of MMT4 in image-text matching in comparison with CLIP.

<table>
<thead>
<tr>
<th>Model</th>
<th>ROC-AUC</th>
<th>PR-AUC</th>
<th>R@0.80</th>
<th>R@0.85</th>
<th>R@0.90</th>
<th>R@0.95</th>
<th>R@0.97</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMT4</td>
<td>0.979</td>
<td>0.975</td>
<td>0.996</td>
<td>0.991</td>
<td>0.972</td>
<td>0.873</td>
<td>0.763</td>
</tr>
<tr>
<td>CLIP [24]</td>
<td>0.996</td>
<td>0.995</td>
<td>0.996</td>
<td>0.996</td>
<td>0.991</td>
<td>0.980</td>
<td>0.963</td>
</tr>
</tbody>
</table>

attribute correction and validation, especially where text sources miss attribute values. Additionally, title generation from image outperformed the state-of-the-art captioning model by generating key phrases in the title. Those can be used for improving Search in e-commerce. Although image-text matching as a generation problem did not perform better than tagging it as classification problem with CLIP, the 98% PR-AUC attained after only fine-tuning MMT4 on less than 2M samples warrants taking the next steps in fine-tuning the model on larger datasets.

As the next steps, we are planning to pre-train and fine-tune MMT4 on larger datasets (with more attributes). The fine-tuned model will be used for a series of downstream tasks such as product data inconsistency detection and attribute correction and the results will be compared with the state-of-the-art models. Finally, we will use the fine-tuned MMT4 to provide more information for other models to improve their performance (teacher pseudo-labeling). For example, generated product type from image and text can help the product type classification pipeline in the e-commerce data warehouse.

ACKNOWLEDGMENTS

Special thanks to Changhe Yuan and Shioulin Sam from Amazon IRIS-Science, and Yi Xu from M5-Search team for their crucial reviews and comments.

REFERENCES
