

# TXtract: Taxonomy-Aware Knowledge Extraction for Thousands of Product Categories

Giannis Karamanolakis\*

Columbia University

New York, NY 10027, USA

gkaraman@cs.columbia.edu {junmaa, lunadong}@amazon.com

Jun Ma, Xin Luna Dong

Amazon.com

Seattle, WA 98109, USA

{junmaa, lunadong}@amazon.com

## Abstract

Extracting structured knowledge from product profiles is crucial for various applications in e-Commerce. State-of-the-art approaches for knowledge extraction were each designed for a single category of product, and thus do not apply to real-life e-Commerce scenarios, which often contain thousands of diverse categories. This paper proposes TXtract, a taxonomy-aware knowledge extraction model that applies to thousands of product categories organized in a hierarchical taxonomy. Through category conditional self-attention and multi-task learning, our approach is both scalable, as it trains a single model for thousands of categories, and effective, as it extracts category-specific attribute values. Experiments on products from a taxonomy with 4,000 categories show that TXtract outperforms state-of-the-art approaches by up to 10% in F1 and 15% in coverage across *all* categories.

## 1 Introduction

Real-world e-Commerce platforms contain billions of products from thousands of different categories, organized in hierarchical taxonomies (see Figure 1). Knowledge about products can be represented in structured form as a catalog of product attributes (e.g., *flavor*) and their values (e.g., “strawberry”). Understanding precise values of product attributes is crucial for many applications including product search, recommendation, and question answering. However, structured attributes in product catalogs are often sparse, leading to unsatisfactory search results and various kinds of defects. Thus, it is invaluable if such structured information can be extracted from product profiles such as product titles and descriptions. Consider for instance the “Ice Cream” product of Figure 1. The corresponding title can potentially



Figure 1: A hierarchical taxonomy with various product categories and the public webpage of a product assigned to “Ice Cream” category.

be used to extract values for attributes, such as “Ben & Jerry’s” for *brand*, “Strawberry Cheesecake” for *flavor*, and “16 oz” for *capacity*.

State-of-the-art approaches for attribute value extraction (Zheng et al., 2018; Xu et al., 2019; Rezk et al., 2019) have employed deep learning to capture features of product attributes effectively for the extraction purpose. However, they are all designed without considering the product categories and thus cannot effectively capture the diversity of categories across the product taxonomy. Categories can be substantially different in terms of applicable attributes (e.g., a “Camera” product should not have *flavor*), attribute values (e.g., “Vitamin” products may have “fruit” *flavor* but “Banana” products should not) and more generally, text patterns used to describe the attribute values (e.g., the phrase “infused with” is commonly followed by a *scent* value such as “lavender” in “Hair Care” products but not in “Mattresses” products).

In this paper, we consider attribute value extraction for real-world hierarchical taxonomies with thousands of product categories, where directly

\* Work performed during internship at Amazon.

applying previous approaches presents limitations. On the one extreme, ignoring the hierarchical structure of categories in the taxonomy and assuming a single “flat” category for all products does not capture category-specific characteristics and, as we will show in Section 5, is not effective. On the other extreme, training a separate deep neural network for each category in the product taxonomy is prohibitively expensive, and can suffer from lack of training data on small categories.

To address the limitations of previous approaches under this challenging setting, we propose a framework for *category-specific* attribute value extraction that is both efficient and effective. Our deep neural network, TXtract, is *taxonomy-aware*: it leverages the hierarchical taxonomy of product categories and extracts attribute values for a product conditional to its category, such that it automatically associates categories with specific attributes, valid attribute values, and category-specific text patterns. TXtract is trained on all categories in parallel and thus can be applied even on small categories with limited labels.

The key question we need to answer is *how to condition deep sequence models on product categories*. Our experiments suggest that following previous work to append category-specific artificial tokens to the input sequence, or concatenate category embeddings to hidden neural network layers is not adequate. There are two key ideas behind our solution. First, we use the category information as context to generate category-specific token embeddings via conditional self-attention. Second, we conduct multi-task training by meanwhile predicting product category from profile texts; this allows us to get token embeddings that are discriminative of the product categories and further improve attribute extraction. Multi-task training also makes our extraction model more robust towards wrong category assignment, which occurs often in real e-Commerce websites.<sup>1</sup>

To the best of our knowledge, TXtract is the first deep neural network that has been applied to attribute value extraction for hierarchical taxonomies with thousands of product categories. In particular, we make three contributions.

---

<sup>1</sup>Examples: (1) an ethernet cable assigned under the “Hair Brushes”: <https://www.amazon.com/dp/B012AE5EP4>; (2) an eye shadow product assigned under “Travel Cases”: <https://www.amazon.com/dp/B07BBM5B33>. Screenshots of these product profiles are taken in 12/2019 and available at the Appendix.

1. We develop TXtract, a taxonomy-aware deep neural network for attribute value extraction from product profiles for multiple product categories. In TXtract, we capture the *hierarchical* relations between categories into *category embeddings*, which in turn we use as context to generate category-specific token embeddings via conditional self-attention.
2. We improve attribute value extraction through multi-task learning: TXtract jointly extracts attribute values and predicts the product’s categories by sharing representations across tasks.
3. We evaluate TXtract on a taxonomy of 4,000 product categories and show that it substantially outperforms state-of-the-art models by up to 10% in F1 and 15% in coverage across *all* product categories.

Although this work focuses on e-Commerce, our approach to leverage taxonomies can be applied to broader domains such as finance, education, and biomedical/clinical research. We leave experiments on these domains for future work.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 presents background and formally defines the problem. Section 4 presents our solution and Section 5 describes experimental results. Finally, Section 6 concludes and suggests future work.

## 2 Related Work

Here, we discuss related work on attribute value extraction (Section 2.1), and multi-task learning/meta-learning (Section 2.2).

### 2.1 Attribute Value Extraction from Product Profiles

Attribute value extraction was originally addressed with rule-based techniques (Nadeau and Sekine, 2007; Vadic et al., 2012; Gopalakrishnan et al., 2012) followed by supervised learning techniques (Ghani et al., 2006; Putthividhya and Hu, 2011; Ling and Weld, 2012; Petrovski and Bizer, 2017; Sheth et al., 2017). Most recent techniques consider open attribute value extraction: emerging attribute values can be extracted by sequence tagging, similar to named entity recognition (NER) (Putthividhya and Hu, 2011; Chiu and Nichols, 2016; Lample et al., 2016; Yadav and Bethard, 2018). State-of-the-art methods employ deep learning for sequence tagging (Zheng et al.,

2018; Xu et al., 2019; Rezk et al., 2019). However, all previous methods can be adapted to a small number of categories and require many labeled datapoints per category.<sup>2</sup> Even the Active Learning method of Zheng et al. (2018) requires humans to annotate at least hundreds of carefully selected examples per category. Our work differs from previous approaches as we consider thousands of product categories organized in a hierarchical taxonomy.

## 2.2 Multi-Task/Meta- Learning

Our framework is related to multi-task learning (Caruana, 1997) as we train a single model simultaneously on all categories (tasks). Traditional approaches consider a small number of different tasks, ranging from 2 to 20 and employ hard parameter sharing (Alonso and Plank, 2017; Yang et al., 2017; Ruder, 2019): the first layers of neural networks are shared across all tasks, while the separate layers (or “heads”) are used for each individual task. In our setting with thousands of different categories (tasks), our approach is efficient as we use a *single* (instead of thousands) head and effective as we distinguish between categories through low-dimensional category embeddings.

Our work is also related to meta-learning approaches based on task embeddings (Finn et al., 2017; Achille et al., 2019; Lan et al., 2019): the target tasks are represented in a low-dimensional space that captures task similarities. However, we generate category embeddings that reflect the *already available, hierarchical* structure of product categories in the taxonomy provided by experts.

## 3 Background and Problem Definition

We now provide background on open attribute value extraction (Section 3.1) and define our problem of focus (Section 3.2).

### 3.1 Open Attribute Value Extraction

Most recent approaches for attribute value extraction rely on the open-world assumption to discover attribute values that have never been seen during training (Zheng et al., 2018). State-of-the-art approaches address extraction with deep sequence tagging models (Zheng et al., 2018; Xu et al.,

<sup>2</sup>Zheng et al. (2018) considered 3 categories: “Dog Dood,” “Cameras,” and “Detergent.” Xu et al. (2019) consider 1 category: “Sports & Entertainment.” Rezk et al. (2019) considered 21 categories and trained a separate model for each category.

Input	Ben	&	Jerry’s	black	cherry	cheesecake	ice	cream
Output	O	O	O	B	I	E	O	O

Table 1: Example of input/output tag sequences for the “flavor” attribute of an ice cream product.

2019; Rezk et al., 2019): each token of the input sequence  $x = (x_1, \dots, x_T)$  is assigned a separate tag from {B, I, O, E}, where “B,” “I,” “O,” and “E” represent the beginning, inside, outside, and end of an attribute, respectively. (Not extracting any values corresponds to a sequence of “O”-only tags.) Table 1 shows an input/output example of *flavor* value extraction from (part of) a product title. Given this output tag sequence, “black cherry cheesecake” is extracted as a *flavor* for the ice cream product.

### 3.2 Problem Definition

We represent the product taxonomy as a tree  $C$ , where the root node is named “Product” and each taxonomy node corresponds to a distinct product category:  $c \in C$ . A directed edge between two nodes represents the category-to-subcategory relationship. A product is assigned to a category node in  $C$ . In practice, there are often thousands of nodes in a taxonomy tree and the category assignment of a product may be incorrect. We now formally define our problem as follows.

**DEFINITION:** Consider a product from a category  $c$  and the sequence of tokens  $x = (x_1, \dots, x_T)$  from its profile, where  $T$  is the sequence length. Let  $a$  be a target attribute for extraction. Attribute extraction identifies subsequences of tokens from  $x$ , each sub-sequence representing a value for  $a$ .

For instance, given (1) a product title  $x =$  “Ben & Jerry’s Strawberry Cheesecake Ice Cream 16 oz,” (2) a product category  $c =$  “Ice Cream,” and (3) a target attribute  $\alpha = flavor$ , we would like to extract “Strawberry Cheesecake” as a *flavor* for this product. Note that *we may not see all valid attribute values during training*.

## 4 TXtract Model: Taxonomy-Aware Knowledge Extraction

In this paper, we address open attribute value extraction using a taxonomy-aware deep sequence tagging model, TXtract. Figure 2 shows the model architecture, which contains two key components: attribute value extraction and product category

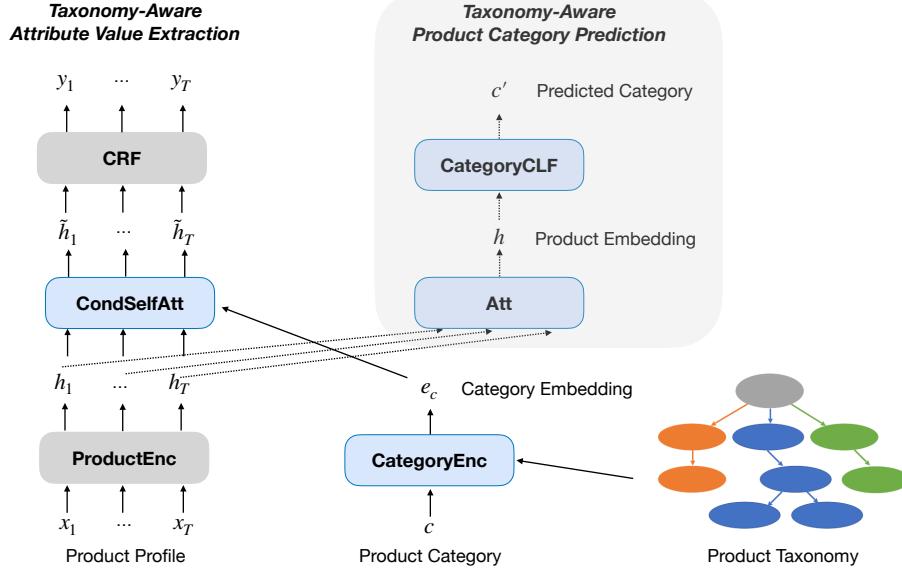


Figure 2: TXtract architecture: tokens  $(x_1, \dots, x_T)$  are classified to BIOE attribute tags  $(y_1, \dots, y_T)$  by conditioning to the product’s category embedding  $e_c$ . TXtract is jointly trained to extract attribute values and assign a product to taxonomy nodes.

prediction, accounting for the two tasks in multi-task training. Both components are taxonomy aware, as we describe next in detail.

#### 4.1 Taxonomy-Aware Attribute Value Extraction

TXtract leverages the product taxonomy for attribute value extraction. The underlying intuition is that knowing the product category may help infer attribute applicability and associate the product with a certain range of valid attribute values. Our model uses the category embedding in conditional self-attention to guide the extraction of category-specific attribute values.

##### 4.1.1 Product Encoder

The product encoder (“ProductEnc”) represents the text tokens of the product profile  $(x_1, \dots, x_T)$  as low-dimensional, real-valued vectors:

$$h_1, \dots, h_T = \text{ProductEnc}(x_1, \dots, x_T) \in \mathbb{R}^d. \quad (1)$$

To effectively capture long-range dependencies between the input tokens, we use word embeddings followed by bidirectional LSTMs (BiLSTMs), similar to previous state-of-the-art approaches (Zheng et al., 2018; Xu et al., 2019).

##### 4.1.2 Category Encoder

Our category encoder (“CategoryEnc”) encodes the hierarchical structure of product categories

such that TXtract understands expert-defined relations across categories, such as “Lager” is a subcategory of “Beer”. In particular, we embed each product category  $c$  (taxonomy node) into a low-dimensional latent space:

$$e_c = \text{CategoryEnc}(c) \in \mathbb{R}^m. \quad (2)$$

To capture the hierarchical structure of the product taxonomy, we embed product categories into the  $m$ -dimensional Poincaré ball (Nickel and Kiela, 2017), because its underlying geometry has been shown to be appropriate for capturing both similarity and hierarchy.

##### 4.1.3 Category Conditional Self-Attention

The key component for taxonomy-aware value extraction is category conditional self-attention (“CondSelfAtt”). CondSelfAtt generates category-specific token embeddings  $(\tilde{h}_i \in \mathbb{R}^d)$  by conditioning on the category embedding  $e_c$ :

$$\tilde{h}_1, \dots, \tilde{h}_T = \text{CondSelfAtt}((h_1, \dots, h_T), e_c). \quad (3)$$

To leverage the mutual interaction between all pairs of token embeddings  $h_t, h_{t'}$  and the category embedding  $e_c$  we use self-attention and compute pairwise sigmoid attention weights:

$$\alpha_{t,t'} = \sigma(w_\alpha^T g_{t,t'} + b_\alpha), \quad t, t' = 1..T. \quad (4)$$

We compute scores  $g_{t,t'}$  using both the token embeddings  $h_t, h_{t'}$  and the category embedding  $e_c$ :

$$g_{t,t'} = \tanh(W_1 h_t + W_2 h_{t'} + W_3 e_c + b_g), \quad (5)$$

where  $W_1 \in \mathbb{R}^{p \times d}$ ,  $W_2 \in \mathbb{R}^{p \times d}$ ,  $W_3 \in \mathbb{R}^{p \times m}$ ,  $w_\alpha \in \mathbb{R}^p$  are trainable attention matrices and  $b_g \in \mathbb{R}^p$ ,  $b_\alpha \in \mathbb{R}$ , are trainable biases. The  $T \times T$  attention matrix  $A = a_{t,t'}$  stores the pairwise attention weights. The contextualized token embeddings are computed as:

$$\tilde{h}_t = \sum_{t'=1}^T \alpha_{t,t'} \cdot h_{t'}. \quad (6)$$

#### 4.1.4 CRF Layer

We feed the contextualized token representations  $\tilde{h} = (\tilde{h}_1, \dots, \tilde{h}_T)$  to CRFs to get the sequence of BIOE tags with the highest probability:

$$y_1, \dots, y_T = \text{CRF}(\tilde{h}_1, \dots, \tilde{h}_T). \quad (7)$$

We then extract attribute values as valid subsequences of the input tokens  $(x_1, \dots, x_T)$  with B/I/E tags (see Section 3.1).

#### 4.1.5 Training for Attribute Value Extraction

Our training objective for attribute value extraction is to minimize the negative conditional log-likelihood of the model parameters on  $N$  training products  $x_i$  with ground truth labels  $\hat{y}_{i1}, \dots, \hat{y}_{iT}$ :

$$L_a = - \sum_{i=1}^N \log Pr(\hat{y}_{i1}, \dots, \hat{y}_{iT} | x_i, c_i) \quad (8)$$

We train our model on all categories in parallel, thus leveraging for a given category products from related categories. To generate training sequence labels from the corresponding attribute values, we use the distant supervision framework of Mintz et al. (2009), similar to Xu et al. (2019), by generating tagging labels according to existing (sparse) values in the Catalog.

## 4.2 Taxonomy-Aware Product Category Prediction

We now describe how we train TXtract for the auxiliary task of product category prediction through multi-task learning. Our main idea is that by encouraging TXtract to predict the product categories using only the product profile, the model will learn token embeddings that are discriminative of the product categories. Thus, we introduce an inductive bias for more effective category-specific attribute value extraction.

### 4.2.1 Attention Layer

Our attention component (“Att”) represents the product profile  $(x_1, \dots, x_T)$  as a single vector  $h \in \mathbb{R}^n$  computed through the weighted combination of the ProductEnc’s embeddings  $(h_1, \dots, h_T)$ :

$$h = \sum_{t=1}^T \beta_t \cdot h_t. \quad (9)$$

This weighted combination allows tokens that are more informative for a product’s category to get higher “attention weights”  $\beta_1, \dots, \beta_T \in [0, 1]$ . For example, we expect  $x_t = \text{“frozen”}$  to receive a relatively high  $\beta_t$  for the classification of a product to the “Ice Cream” category. We compute the attention weights as:

$$\beta_t = \text{softmax}(u_c^T \tanh(W_c h_t + b_c)), \quad (10)$$

where  $W_c \in \mathbb{R}^{q \times d}$ ,  $b_c \in \mathbb{R}^q$ ,  $u_c \in \mathbb{R}^q$  are trainable attention parameters.

### 4.2.2 Category Classifier

Our category classifier (“CategoryCLF”) classifies the product embedding  $h$  to the taxonomy nodes. In particular, we use a sigmoid classification layer to predict the probabilities of the taxonomy nodes:

$$p_1, \dots, p_{|C|} = \text{sigmoid}(W_d h + b_d), \quad (11)$$

where  $W_d \in \mathbb{R}^{|C| \times d}$  and  $b_d \in \mathbb{R}^{|C|}$  are trainable parameters. We compute sigmoid (instead of softmax) node probabilities because we treat category prediction as *multi-label* classification, as we describe next.

### 4.2.3 Training for Category Prediction

Training for “flat” classification of products to thousands of categories is not effective because the model is fully penalized if it does not predict the exact true category  $\hat{c}$  while at the same time ignores parent-children category relations. Here, we conduct “hierarchical” classification by incorporating the hierarchical structure of the product taxonomy into a *taxonomy-aware* loss function.

The insight behind our loss function is that a product assigned under  $\hat{c}$  could also be assigned under any of the ancestors of  $\hat{c}$ . Thus, we consider hierarchical multi-label classification and encourage TXtract to assign a product to all nodes in the path from  $\hat{c}$  to the root, denoted by  $(\hat{c}_K, \hat{c}_{K-1}, \dots, \hat{c}_1)$ , where  $K$  is the level of the

node  $\hat{c}$  in the taxonomy tree. The model is thus encouraged to learn the hierarchical taxonomy relations and will be penalized less if it predicts high probabilities for ancestor nodes (e.g., "Beer" instead of "Lager" in Figure 1).

Our minimization objective is the *weighted* version of the binary cross-entropy (instead of *unweighted* categorical cross-entropy) loss:<sup>3</sup>

$$L_b = \sum_{c \in C} w_c (y_c \cdot \log p_c + (1 - y_c) \cdot \log(1 - p_c)), \quad (12)$$

For the nodes in the path from  $\hat{c}$  to the root ( $\hat{c}_K, \hat{c}_{K-1}, \dots, \hat{c}_1$ ), we define positive labels  $y_c = 1$  and weights  $w_c$  that are exponentially decreasing ( $w^0, w^1, \dots, w^{K-1}$ ), where  $0 < w \leq 1$  is a tunable hyper-parameter. The remaining nodes in  $C$  receive negative labels  $y_c = 0$  and fixed weight  $w_c = w^{K-1}$ .

### 4.3 Multi-task Training

We jointly train TXtract for attribute value extraction and product category prediction by combining the loss functions of Eq. (8) and Eq. (12):

$$L = \gamma \cdot L_a + (1 - \gamma) \cdot L_b, \quad (13)$$

where  $\gamma \in [0, 1]$  is a tunable hyper-parameter. Here, we employ multi-task learning, and share ProductEnc across both tasks.

## 5 Experiments

We empirically evaluated TXtract and compared it with state-of-the-art models and strong baselines for attribute value extraction on 4000 product categories. TXtract leads to substantial improvement across all categories, showing the advantages of leveraging the product taxonomy.

### 5.1 Experimental Settings

**Dataset:** We trained and evaluated TXtract on products from public web pages of Amazon.com. We randomly selected 2 million products from 4000 categories under 4 general domains (sub-trees) in the product taxonomy: Grocery, Baby product, Beauty product, and Health product.

**Experimental Setup:** We split our dataset into training (60%), validation (20%), and test (20%) sets. We experimented with extraction of *flavor*, *scent*, and *brand* values from product titles, and

<sup>3</sup>For simplicity in notation, we define Eq 12 for a single product. Defining for all training products is straightforward.

with *ingredient* values from product titles and descriptions. For each attribute, we trained TXtract on the training set and evaluated the performance on the held-out test set.

**Evaluation Metrics:** For a robust evaluation of attribute value extraction, we report several metrics. For a test product, we consider as true positive the case where the extracted values match at least one of the ground truth values (as some of the ground truth values may not exist in the text) and do not contain any wrong values.<sup>4</sup> We compute *Precision* (Prec) as the number of "matched" products divided by the number of products for which the model extracts at least one attribute value; *Recall* (Rec) as the number of "matched" products divided by the number of products associated with attribute values; and *F1* score as the harmonic mean of Prec and Rec. To get a global picture of the model's performance, we consider micro-average scores (Mi\*), which first aggregates products across categories and computes Prec/Rec/F1 globally. To evaluate per-category performance we consider macro-average scores (Ma\*), which first computes Prec/Rec/F1 for each category and then aggregates per-category scores. To evaluate the capability of our model to discover (potentially new) attribute values, we also report the *Value vocabulary* (Vocab) as the total number of unique attribute values extracted from the test set (higher number is often better); and *Coverage* (Cov), as the number of products for which the model extracted at least one attribute value, divided by the total number of products.

For product category (multi-label) classification we reported the area under Precision-Recall curve (AUPR), Precision, Recall, and F1 score.

**Model Configuration:** We implemented our model in Tensorflow (Abadi et al., 2016) and Keras.<sup>5</sup> For a fair comparison, we consider the same configuration as OpenTag for the ProductEnc (BiLSTM)<sup>6</sup> and CRF components. For model configuration details see the appendix.

**Model Comparison:** We compared our model with state-of-the-art models in the literature and

<sup>4</sup>For example, if the ground-truth is  $[v_1]$  but the system extracts  $[v_1, v_2, v_3]$ , the extraction is considered as incorrect.

<sup>5</sup><https://keras.io/>

<sup>6</sup>We expect to see further performance improvement by considering pre-trained language models (Radford et al., 2018; Devlin et al., 2019) for ProductEnc, which we leave for future work.

Attr.	Model	Vocab	Cov	Micro F1	Micro Prec	Micro Rec	Macro F1	Macro Prec	Macro Rec
<i>Flavor</i>	OpenTag	6,756	73.2	57.5	70.3	49.6	54.6	68.0	47.3
	TXtract	<b>13,093</b>	<b>83.9</b> ↑14.6%	<b>63.3</b> ↑10.1%	<b>70.9</b> ↑0.9%	<b>57.8</b> ↑16.5%	<b>59.3</b> ↑8.6%	<b>68.4</b> ↑0.6%	<b>53.8</b> ↑13.7%
<i>Scent</i>	OpenTag	10,525	75.8	70.6	<b>87.6</b>	60.2	59.3	<b>79.7</b>	50.8
	TXtract	<b>13,525</b>	<b>83.2</b> ↑9.8%	<b>73.7</b> ↑4.4%	86.1 ↓1.7%	<b>65.7</b> ↑9.1%	<b>59.9</b> ↑10.1%	78.3 ↓1.8%	<b>52.1</b> ↑2.6%
<i>Brand</i>	OpenTag	48,943	73.1	63.4	81.6	51.9	51.7	75.1	41.5
	TXtract	<b>64,704</b>	<b>82.9</b> ↑13.4%	<b>67.5</b> ↑6.5%	<b>82.7</b> ↑1.3%	<b>56.5</b> ↑8.1%	<b>55.3</b> ↑7.0%	<b>75.2</b> ↑0.1%	<b>46.8</b> ↑12.8%
<i>Ingred.</i>	OpenTag	9,910	70.0	35.7	46.6	29.1	20.9	34.6	16.7
	TXtract	<b>18,980</b>	<b>76.4</b> ↑9.1%	<b>37.1</b> ↑3.9%	<b>48.3</b> ↑3.6%	<b>30.1</b> ↑3.3%	<b>24.2</b> ↑15.8%	<b>37.4</b> ↑8.1%	<b>19.8</b> ↑18.6%
Average relative increase			↑11.7%	↑6.2%	↑1.0%	↑9.3%	↑10.4%	↑6.8%	↑11.9%

Table 2: Extraction results for *flavor*, *scent*, *brand*, and *ingredients* across 4,000 categories. Across all attributes, TXtract improves OpenTag by 11.7% in coverage, 6.2% in micro-average F1, and 10.4% in macro-average F1.

introduced additional strong baselines:

1. “OpenTag”: the model of [Zheng et al. \(2018\)](#). It is a special case of our system that consists of the ProductEnc and CRF components without leveraging the taxonomy.
2. “Title+\*”: a class of models for conditional attribute value extraction, where the taxonomy is introduced by artificially appending extra tokens  $x'_1, \dots, x'_{T'}$  and a special separator token (<SEP>) to the beginning of a product’s text, similar to [Johnson et al. \(2017\)](#):  

$$x' = (x'_1, \dots, x'_{T'}, \text{<SEP>}, x_1, \dots, x_T)$$
 Tokens  $x'_1, \dots, x'_{T'}$  contain category information such as unique category id (“Title+id”), category name (“Title+name”), or the names of all categories in the path from the root to the category node, separated by an extra token <SEP2> (“Title+path”).
3. “Concat-\*”: a class of models for taxonomy-aware attribute value extraction that concatenate the category embedding to the word embedding (-wemb) or hidden BiLSTM embedding layer (-LSTM) instead of using conditional self-attention. We evaluate Euclidean embeddings (“Concat-\*‐Euclidean”) and Poincaré embeddings (“Concat-\*‐Poincaré”).
4. “Gate”: a model that leverages category embeddings  $e_c$  in a gating layer ([Cho et al., 2014](#); [Ma et al., 2019](#)):  $\tilde{h}_t = h_t \otimes \sigma(W_4 h_t + W_5 e_c)$ , where  $W_4 \in \mathbb{R}^{p \times d}$ ,  $W_5 \in \mathbb{R}^{p \times m}$  are trainable matrices, and  $\otimes$  denotes element-wise multiplication. Our conditional self-attention is different as it leverages pairwise instead of single-token interactions with category embeddings.
5. “CondSelfAtt”: the model with our conditional self-attention mechanism (Section 4.1.3). CondSelfAtt extracts attribute values but does not predict the product category.

6. “MT-\*”: a multi-task learning model that jointly performs (*not* taxonomy-aware) attribute value extraction and category prediction. “MT-flat” assumes “flat” categories, whereas “MT-hier” considers the hierarchical structure of the taxonomy (Section 4.2.3).

7. “TXtract”: our model that jointly performs *taxonomy-aware* attribute value extraction (same as CondSelfAtt) and *hierarchical* category prediction (same as MT-hier).

Here, we do not report previous models (e.g., BiLSTM-CRF) for sequence tagging ([Huang et al., 2015](#); [Kozareva et al., 2016](#); [Lample et al., 2016](#)), as OpenTag has been shown to outperform these models in [Zheng et al. \(2018\)](#). Moreover, when considering attributes separately, the model of [Xu et al. \(2019\)](#) is the same as OpenTag, but with a different ProductEnc component; since we use the same ProductEnc for all alternatives, we expect/observe the same trend and do not report its performance.

## 5.2 Experimental Results

Table 2 reports the results across all categories. For detailed results see Figure 6 in Appendix. Over all categories, our taxonomy-aware TXtract substantially improves over the state-of-the-art OpenTag by up to 10.1% in Micro F1, 14.6% in coverage, and 93.8% in vocabulary (for *flavor*).

Table 3 shows results for the four domains of our taxonomy under different training granularities: training on all domains versus training only on the target domain. Regardless of the configuration, TXtract substantially outperforms OpenTag, showing the general advantages of our approach. Interestingly, although training a single model on all of the four domains obtains lower F1 for *Flavor*, it obtains better results for *Scent*: training fewer models does not necessarily lead to

Domain		Attr.	OpenTag/TXtract
Train	Test		Micro F1
all Grocery	Grocery	<i>Flavor</i>	60.3 / <b>64.9</b> ↑7.6%
	Grocery		65.4 / <b>70.5</b> ↑7.8%
all Baby	Baby	<i>Flavor</i>	54.4 / <b>63.0</b> ↑15.8%
	Baby		69.2 / <b>71.8</b> ↑3.8%
all Beauty	Beauty	<i>Scent</i>	76.9 / <b>79.5</b> ↑3.4%
	Beauty		76.9 / <b>79.0</b> ↑2.7%
all Health	Health	<i>Scent</i>	63.0 / <b>69.1</b> ↑9.7%
	Health		60.9 / <b>63.5</b> ↑4.3%

Table 3: Evaluation results for each domain under training configurations of different granularity. TXtract outperforms OpenTag under all configurations.

lower quality and may actually improve extraction by learning from neighboring taxonomy trees.

### 5.3 Ablation Study

Table 4 reports the performance of several alternative approaches for *flavor* value extraction across all categories. OpenTag does not leverage the product taxonomy, so it is outperformed by most approaches that we consider in this work.

**Implicit vs. explicit conditioning on categories.** “Title+\*” baselines fail to leverage the taxonomy, thus leading to lower F1 score than OpenTag: implicitly leveraging categories as artificial tokens appended to the title is not effective in our setting.

Representing the taxonomy with category embeddings leads to significant improvement over OpenTag and “Title+\*” baselines: even simpler approaches such as “Concat-\*‐Euclidean” outperform OpenTag across all metrics. However, “Concat-\*” and “Gate-\*” do not leverage category embeddings as effectively as “CondSelfAtt”: conditioning on the category embedding for the computation of the pair-wise attention weights in the self-attention layer appears to be the most effective approach for leveraging the product taxonomy.

**Multi-task Learning.** In Table 4, both MT-flat and MT-hier, which do not condition on the product taxonomy, outperform OpenTag on attribute value extraction: by learning to predict the product category, our model implicitly learns to condition on the product category for effective attribute value extraction. MT-hier outperforms MT-flat: leveraging the hierarchical structure of the taxonomy is more effective than assuming flat categories. Table 5 shows that category prediction is more effective when considering the hierarchi-

Model	TX	MT	Micro F1
OpenTag	-	-	57.5
Title+id	✓	-	55.7 ↓3.1%
Title+name	✓	-	56.9 ↓1.0%
Title+path	✓	-	54.3 ↓5.6%
Concat-wemb-Euclidean	✓	-	60.1 ↑4.5%
Concat-wemb-Poincaré	✓	-	60.6 ↑5.4%
Concat-LSTM-Euclidean	✓	-	60.1 ↑4.5%
Concat-LSTM-Poincaré	✓	-	60.8 ↑5.7%
Gate-Poincaré	✓	-	60.6 ↑5.4%
CondSelfAtt-Poincaré	✓	-	61.9 ↑7.7
MT-flat	-	✓	60.9 ↑5.9%
MT-hier	-	✓	61.5 ↑7.0%
Concat & MT-hier	✓	✓	62.3 ↑8.3%
Gate & MT-hier	✓	✓	61.1 ↑6.3%
CondSelfAtt & MT-hier	✓	✓	<b>63.3</b> ↑10.1%

Table 4: Ablation study for *flavor* extraction across 4,000 categories. “TX” column indicates whether the taxonomy is leveraged for attribute value extraction (Section 4.1). “MT” column indicates whether multi-task learning is used (Section 4.2).

Category Prediction	AUPR	F1	Prec	Rec
Flat	0.61	53.9	74.2	48.0
Hierarchical	<b>0.68</b>	<b>62.7</b>	<b>80.4</b>	<b>56.9</b>

Table 5: Performance of product classification to the 4,000 nodes in the taxonomy using flat versus hierarchical multi-task learning.

cal structure of the categories into our taxonomy-aware loss function than assuming flat categories.

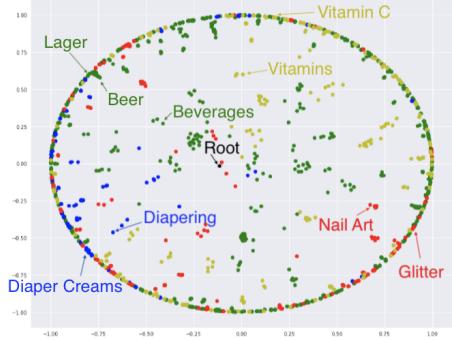
### 5.4 Visualization of Poincaré Embeddings

Poincaré embeddings effectively capture the hierarchical structure of the product taxonomy: Figure 3a plots the embeddings of product categories in the 2-dimensional Poincaré disk.<sup>7</sup> Figure 3b plots the embeddings trained in the 50-dimensional Poincaré ball and projected to the 2-dimensional Euclidean space through t-SNE (Maaten and Hinton, 2008).

### 5.5 Examples of Extracted Attribute Values

Figure 4 shows examples of product titles and attribute values extracted by OpenTag or TXtract. TXtract is able to detect category-specific values: in Figure 4a, “Purple Lemonade” is a valid *flavor* for “Vitamin Pills” but not for most of other categories. OpenTag, which ignores product categories, fails to detect this value while TXtract

<sup>7</sup>We train 2-dimensional Poincaré embeddings only for visualization. In our experiments we use  $d = 50$  dimensions.



(a) Taxonomy embeddings in the 2-dimensional Poincaré disk, where the distance of points grows exponentially to the radius. Leaf nodes are placed close to the boundary of the disk.



(b) Taxonomy embeddings projected from the 50-dimensional Poincaré ball to the 2-dimensional Euclidean space using t-SNE. Small clusters correspond to taxonomy sub-trees.

Figure 3: Poincaré embeddings of taxonomy nodes (product categories). Each point is a product category. Categories are colored based on the first-level taxonomy where they belong (green: Grocery products, blue: Baby products, red: Beauty products, yellow: Health products). Related categories in the taxonomy (e.g., categories belonging to the same sub-tree) have similar embeddings.

**Title** = Controlled Labs Purple Wraith 90 Servings - Purple Lemonade



**ASIN** = B00CX96KTQ  
**Category** = Vitamins & Dietary Supplements  
**OpenTag (flavor)** = (empty)  
**TXtract (flavor)** = “purple lemonade”

(a)

**Title** = Mason Vitamins Melatonin 500 mcg Fast Meltz Tablets, Fruit, 60 Count



**ASIN** = B015K3Y728  
**Category** = Vitamins & Dietary Supplements  
**OpenTag (flavor)** = (empty)  
**TXtract (flavor)** = “fruit”

(c)

**Title** = Click - Espresso Protein Drink Vanilla Latte - 16 oz.



**ASIN** = B005P0LKTU  
**Category** = Sports Nutrition  
**OpenTag (flavor)** = “espresso”  
**TXtract (flavor)** = “vanilla latte”

(b)

**Title** = HP95(TM) Fashion Glitter Matte Eye Shadow Powder Palette Single Shimmer Eyeshadow (10#)



**ASIN** = B07BBM5B33  
**Category** = Eyeshadow  
**OpenTag (scent)** = palette  
**TXtract (scent)** = (empty)

(d)

Figure 4: Examples of extracted attribute values from OpenTag and TXtract.

successfully extracts it as a *flavor*. TXtract also learns attribute applicability: in Figure 4d, OpenTag erroneously extracts “palette” as *scent* for an “Eyeshadow” product, while this product should not have *scent*; on the other hand, TXtract, which considers category embeddings, does not extract any *scent* values for this product.

## 6 Conclusions and Future Work

We present a novel method for large-scale attribute value extraction for products from a taxonomy with thousands of product categories. Our proposed model, TXtract, is both efficient and effective: it leverages the taxonomy into a deep neural network to improve extraction quality and can extract attribute values on all categories in parallel.

TXtract significantly outperforms state-of-the-art approaches and strong baselines under a taxonomy with thousands of product categories. Interesting future work includes applying our techniques to different taxonomies (e.g., biomedical) and training a model for different attributes.

## Acknowledgments

The authors would like to sincerely thank Ron Benson, Christos Faloutsos, Andrey Kan, Yan Liang, Yaqing Wang, and Tong Zhao for their insightful comments on the paper, and Gabriel Blanco, Alexandre Manduca, Saurabh Deshpande, Jay Ren, and Johanna Umana for their constructive feedback on data integration for the experiments.

## References

- Mart’ in Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283.
- Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless Fowlkes, Stefano Soatto, and Pietro Perona. 2019. Task2vec: Task embedding for meta-learning. *arXiv preprint arXiv:1902.03545*.
- Hector Martinez Alonso and Barbara Plank. 2017. When is multitask learning effective? semantic sequence prediction under varying data conditions. In *EACL 2017-15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1–10.
- Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.
- Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370.
- Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.
- 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*.
- Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR.org.
- Rayid Ghani, Katharina Probst, Yan Liu, Marko Krema, and Andrew Fano. 2006. Text mining for product attribute extraction. *ACM SIGKDD Explorations Newsletter*, 8(1):41–48.
- Vishrawas Gopalakrishnan, Suresh Parthasarathy Iyengar, Amit Madaan, Rajeev Rastogi, and Srinivasan Sengamedu. 2012. Matching product titles using web-based enrichment. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 605–614. ACM.
- Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.
- Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2017. Google’s multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.
- Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Zornitsa Kozareva, Qi Li, Ke Zhai, and Weiwei Guo. 2016. Recognizing salient entities in shopping queries. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 107–111.
- Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*, pages 260–270.
- Lin Lan, Zhenguo Li, Xiaohong Guan, and Pinghui Wang. 2019. Meta reinforcement learning with task embedding and shared policy. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 2794–2800. International Joint Conferences on Artificial Intelligence Organization.
- Xiao Ling and Daniel S Weld. 2012. Fine-grained entity recognition. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.
- Chen Ma, Peng Kang, and Xue Liu. 2019. Hierarchical gating networks for sequential recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM.
- Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.
- Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.
- David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.
- Maximillian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. In *Advances in neural information processing systems*, pages 6338–6347.

- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.
- Petar Petrovski and Christian Bizer. 2017. Extracting attribute-value pairs from product specifications on the web. In *Proceedings of the International Conference on Web Intelligence*, pages 558–565. ACM.
- Duangmanee Pew Putthividhya and Junling Hu. 2011. Bootstrapped named entity recognition for product attribute extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1557–1567. Association for Computational Linguistics.
- Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. <https://blog.openai.com/language-unsupervised>.
- Martin Rezk, Laura Alonso Alemany, Lasguido Nio, and Ted Zhang. 2019. Accurate product attribute extraction on the field. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 1862–1873. IEEE.
- Sebastian Ruder. 2019. *Neural Transfer Learning for Natural Language Processing*. Ph.D. thesis, National University Of Ireland, Galway.
- Amit P. Sheth, Axel Ngonga, Yin Wang, Elizabeth Chang, Dominik Slezak, Bogdan Franczyk, Rainer Alt, Xiaohui Tao, and Rainer Unland, editors. 2017. *Proceedings of the International Conference on Web Intelligence, Leipzig, Germany, August 23-26, 2017*. ACM.
- Damir Vandic, Jan-Willem Van Dam, and Flavius Frasincar. 2012. Faceted product search powered by the semantic web. *Decision Support Systems*, 53(3):425–437.
- 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*, pages 5214–5223.
- Vikas Yadav and Steven Bethard. 2018. A survey on recent advances in named entity recognition from deep learning models. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2145–2158.
- Zhilin Yang, Ruslan Salakhutdinov, and William W Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. In *Proceedings of the International Conference on Learning Representations*.
- 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*, pages 1049–1058. ACM.

## A Appendix

For reproducibility, we provide details on TXtract configuration (Section A.1). We also report detailed evaluation results (Section A.2).

### A.1 TXtract Configuration

We implemented our model in Tensorflow (Abadi et al., 2016) and Keras.<sup>8</sup> To achieve a fair comparison with OpenTag (Zheng et al., 2018), and to ensure that performance improvements stem from leveraging the product taxonomy, we use exactly the same components and configuration as OpenTag for ProductEnc: We initialize the word embedding layer using 100-dimensional pre-trained Glove embeddings (Pennington et al., 2014). We use masking to support variable-length input. Each of the LSTM layers has a hidden size of 100 dimensions, leading to a BiLSTM layer with  $d = 200$  dimensional embeddings. We set the dropout rate to 0.4. For CategoryEnc, we train  $m = 50$ -dimensional Poincaré embeddings.<sup>9</sup> For CondSelfAtt, we use  $p = 50$  dimensions. For Att, we use  $q = 50$  dimensions. For multi-task training, we obtain satisfactory performance with default hyper-parameters  $\gamma = 0.5$ ,  $w = 1$ , while we leave fine-tuning for future work. For parameter optimization, we use Adam (Kingma and Ba, 2014) with a batch size of 32. We train our model for up to 30 epochs and quit training if the validation loss does not decrease for more than 3 epochs.

### A.2 Extra Results

Table 6 reports extraction results (of TXtract trained on all domains) for each domain separately. Table 7 reports category classification results for each domain separately. Table 8 reports several evaluation metrics for our ablation study.

---

<sup>8</sup><https://keras.io/>

<sup>9</sup>We use the public code in provided by Nickel and Kiela (2017): <https://github.com/facebookresearch/poincare-embeddings>



Figure 5: Snapshot of <https://www.amazon.com/dp/B012AE5EP4>. This ethernet cable has been erroneously assigned under “Hair Brushes” category. (The assignment can be seen on the top left part of the screenshot.)

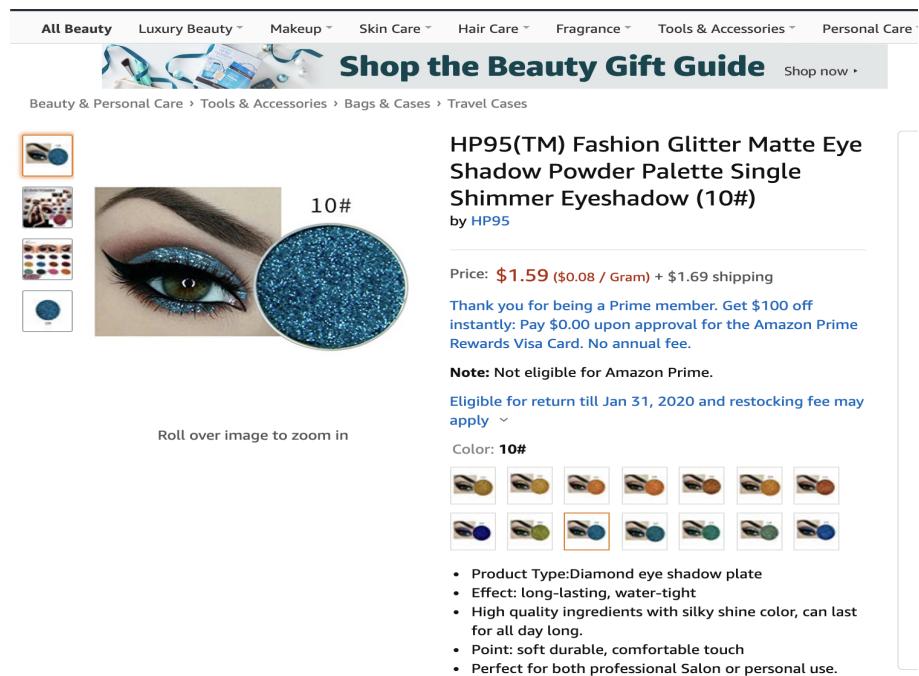


Figure 6: Snapshot of <https://www.amazon.com/dp/B07BBM5B33>. This eye shadow product has been erroneously assigned under “Travel Cases” category. (The assignment can be seen on the top left part of the screenshot.)

		Grocery Products				Baby Products				Beauty Products				Health Products			
Attr.	Model	Vocab	Cov	miF1	maF1	Vocab	Cov	miF1	maF1	Vocab	Cov	miF1	maF1	Vocab	Cov	miF1	maF1
<i>Flavor</i>	OpenTag	4364	79.6	60.3	59.0	264	53.1	54.4	45.0	832	45.8	41.1	32.0	1296	58.2	53.9	47.0
	TXtract	<b>8607</b>	<b>89.1</b>	<b>64.9</b>	<b>62.8</b>	<b>414</b>	<b>72.8</b>	<b>63.0</b>	<b>56.1</b>	<b>1684</b>	<b>61.3</b>	<b>46.5</b>	<b>35.6</b>	<b>2388</b>	<b>71.5</b>	<b>67.3</b>	<b>57.5</b>
<i>Scent</i>	OpenTag	446	75.5	56.8	48.4	<b>593</b>	69.7	35.7	20.3	7007	78.5	76.9	67.9	2479	68.1	63.0	47.5
	TXtract	<b>565</b>	<b>87.4</b>	<b>61.2</b>	<b>51.4</b>	589	<b>72.1</b>	<b>38.1</b>	<b>22.0</b>	<b>9048</b>	<b>85.6</b>	<b>79.5</b>	<b>68.4</b>	<b>3322</b>	<b>79.9</b>	<b>69.1</b>	<b>48.2</b>
<i>Brand</i>	OpenTag	5150	68.8	62.9	52.7	11166	72.2	66.0	54.0	15394	77.2	68.8	54.7	17233	71.2	57.8	45.9
	TXtract	<b>6944</b>	<b>78.9</b>	<b>67.4</b>	<b>55.1</b>	<b>14965</b>	<b>81.0</b>	<b>72.9</b>	<b>56.2</b>	<b>19821</b>	<b>85.1</b>	<b>72.7</b>	<b>57.2</b>	<b>22974</b>	<b>82.9</b>	<b>60.5</b>	<b>52.4</b>
<i>Ingred.</i>	OpenTag	3402	82.5	40.5	30.1	490	50.7	27.7	22.4	2767	65.1	<b>33.6</b>	<b>26.8</b>	3251	66.7	34.6	29.9
	TXtract	<b>6155</b>	<b>87.3</b>	<b>43.1</b>	<b>36.5</b>	<b>835</b>	<b>59.7</b>	<b>30.5</b>	<b>24.3</b>	<b>5539</b>	<b>70.6</b>	32.9	26.6	<b>6451</b>	<b>74.2</b>	<b>36.5</b>	<b>31.2</b>

Table 6: Extraction results for *flavor*, *scent*, *brand*, and *ingredients* for each of our 4 domains (sub-trees).

	Grocery Products				Baby Products				Beauty Products				Health Products			
MT type	AUPR	F1	Prec	Rec	AUPR	F1	Prec	Rec	AUPR	F1	Prec	Rec	AUPR	F1	Prec	Rec
flat	45.9	21.4	63.3	13.7	65.9	23.7	68.4	17.4	63.7	62.4	78.8	56.5	49.8	38.8	60.7	32.7
hierarchical	<b>47.3</b>	<b>29.7</b>	<b>68.4</b>	<b>19.9</b>	<b>68.5</b>	<b>29.4</b>	<b>72.6</b>	<b>22.9</b>	<b>72.1</b>	<b>71.5</b>	<b>83.1</b>	<b>66.4</b>	<b>56.3</b>	<b>47.7</b>	<b>74.6</b>	<b>39.8</b>

Table 7: Product category classification results

Model	TX	MT	Vocab Cov (%)		Micro-average			Macro-average		
			F1	Prec	F1	Prec	Rec	F1	Prec	Rec
OpenTag	-	-	6,756	73.2	57.5	70.3	49.6	54.6	68.0	47.3
Title+id	✓	-	6,400	69.1	55.7	70.6	46.9	53.3	68.9	45.1
Title+name	✓	-	5,328	70.6	56.9	71.2	48.4	54.2	69.1	46.3
Title+path	✓	-	4,608	64.6	54.3	72.0	44.6	51.9	69.1	43.2
Concat-wemb-Euclidean	✓	-	9,768	76.3	60.1	71.6	52.9	57.4	69.0	50.6
Concat-wemb-Poincaré	✓	-	8,684	74.3	60.6	73.4	52.7	57.7	70.2	50.6
Concat-LSTM-Euclidean	✓	-	9,255	75.9	60.1	71.9	52.8	57.5	69.4	50.6
Concat-LSTM-Poincaré	✓	-	8,893	75.2	60.8	72.9	53.2	57.9	70.3	50.9
Gate-Poincaré	✓	-	9,690	77.1	60.6	71.5	53.5	57.7	69.3	51.0
CondSelfAtt-Poincaré	✓	-	12,558	83.1	61.9	68.8	57.0	58.3	66.5	53.1
MT-flat	-	✓	8,699	72.2	60.9	74.7	52.4	57.8	70.3	50.5
MT-hier	-	✓	9,528	73.4	61.5	74.5	53.2	58.3	<b>70.9</b>	51.1
Concat & MT-hier	✓	✓	9,316	74.6	62.3	<b>75.0</b>	54.3	59.0	70.8	52.1
Gate & MT-hier	✓	✓	10,845	80.0	61.1	70.7	54.8	57.9	67.9	51.8
CondSelfAtt & MT-hier (TXtract)	✓	✓	<b>13,093</b>	<b>83.9</b>	<b>63.3</b>	70.9	<b>57.8</b>	<b>59.3</b>	68.4	<b>53.8</b>

Table 8: Results for *flavor* extraction across all categories. “TX” column indicates whether the taxonomy is leveraged for attribute value extraction (Section 4.1). “MT” column indicates whether multi-task learning is used (Section 4.2).