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 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...
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.1

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.

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; Vandic 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.,

---

1Examples: (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.
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. 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., 2019; Rezk et al., 2019): each token of the input sequence $x = (x_1, \ldots, x_T)$ 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 $x = (x_1, \ldots, x_T)$ from its profile, where $T$ is the sequence length. Let $a$ be a target attribute for extraction. Attribute extraction identifies sub-sequences 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 $a =$ 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
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 \((x_1, \ldots, x_T)\) as low-dimensional, real-valued vectors:

\[
h_1, \ldots, h_T = \text{ProductEnc}(x_1, \ldots, x_T) \in \mathbb{R}^d.\]

(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 sub-category of “Beer”. In particular, we embed each product category \(c\) (taxonomy node) into a low-dimensional latent space:

\[
e_c = \text{CategoryEnc}(c) \in \mathbb{R}^m.\]

(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 \((\tilde{h}_1, \ldots, \tilde{h}_T)\) by conditioning on the category embedding \(e_c\):

\[
\tilde{h}_1, \ldots, \tilde{h}_T = \text{CondSelfAtt}((h_1, \ldots, h_T), e_c).\]

(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:

\[
\alpha_{t,t'} = \sigma(w^T_{\alpha} g_{t,t'} + b_{\alpha}), \quad t, t' = 1..T.\]

(4)

We compute scores \(g_{t,t'}\) using both the token embeddings \(h_t, h_{t'}\) and the category embedding \(e_c\):
\[ g_{t,t'} = \tanh(W_1h_t + W_2h_{t'} + W_3e_c + b_g), \] (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:

\[ \bar{h}_t = \sum_{t'=1}^{T} \alpha_{t,t'} \cdot h_{t'}. \] (6)

### 4.1.4 CRF Layer

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

\[ y_1, \ldots, y_T = \text{CRF}(\bar{h}_1, \ldots, \bar{h}_T). \] (7)

We then extract attribute values as valid subsequences of the input tokens \((x_1, \ldots, x_T)\) 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 \( \hat{y}_{i1}, \ldots, \hat{y}_{iT} \):

\[ L_a = - \sum_{i=1}^{N} \log Pr(\hat{y}_{i1}, \ldots, \hat{y}_{iT} | x_i, c_i). \] (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 \((x_1, \ldots, x_T)\) as a single vector \( h \in \mathbb{R}^n \) computed through the weighted combination of the ProductEnc’s embeddings \((h_1, \ldots, h_T)\):

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

This weighted combination allows tokens that are more informative for a product’s category to get higher “attention weights” \( \beta_1, \ldots, \beta_T \in [0, 1] \). For example, we expect \( x_1 \) “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:

\[ \beta_t = \text{softmax}(u_c^T \tanh(W_c h_t + b_c)), \] (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:

\[ p_1, \ldots, p_c = \text{sigmoid}(Wdh + b_d), \] (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 \((\hat{c}_K, \hat{c}_{K-1}, \ldots, c_1)\), 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:

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

(12)

For the nodes in the path from $\hat{c}$ to the root ($\hat{c}_K, \hat{c}_{K-1}, \ldots, \hat{c}_1$), we define positive labels $y_c = 1$ and weights $w_c$ that are exponentially decreasing ($w^0, w^1, \ldots, w^{K-1}$), 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):

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

(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 (subtrees) 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 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. 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 harmony mean of Prec and Rec. To get a global picture of the model’s performance, we consider micro-average scores (Ma*), 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. For a fair comparison, we consider the same configuration as OpenTag for the ProductEnc (BiLSTM) 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

---

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

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

5https://keras.io/

6We 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.
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 $x_1^T, \ldots, x_{PT}^T$ and a special separator token (SEP) to the beginning of a product’s text, similar to Johnson et al. (2017):

   $$x' = (x_1^T, \ldots, x_{PT}^T, \text{SEP}, x_1, \ldots, x_T)$$

   Tokens $x_1^T, \ldots, x_{PT}^T$ 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): $h_t = h_t \odot (W_2 h_t + W_5 e_c)$, where $W_4 \in \mathbb{R}^{p \times d}$, $W_5 \in \mathbb{R}^{p \times m}$ are trainable matrices, and $\odot$ 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
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 hierarchical 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. 7 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

---

7We 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.

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

(a) Title = Controlled Labs Purple Wraath 90 Servings - Purple Lemonade
ASIN = B00CX96KTQ
Category = Vitamins & Dietary Supplements
OpenTag (flavor) = (empty)
TXtract (flavor) = “purple lemonade”

(b) Title = Click - Espresso Protein Drink Vanilla Latte - 16 oz.
ASIN = B005P0LKTU
Category = Sports Nutrition
OpenTag (flavor) = “espresso”
TXtract (flavor) = “vanilla latte”

(c) Title = Mason Vitamins Melatonin 500 mcg Fast Meltz Tablets, Fruit, 60 Count
ASIN = B015K3Y728
Category = Vitamins & Dietary Supplements
OpenTag (flavor) = (empty)
TXtract (flavor) = “fruit”

(d) Title = HP95(TM) Fashion Glitter Matte Eye Shadow Powder Palette Single Shimmer Eyeshadow (10#)
ASIN = B07BBM5B33
Category = Eyeshadow
OpenTag (scent) = palette
TXtract (scent) = (empty)

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


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. 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.\(^9\) 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 \( \gamma = 0.5, \omega = 1 \), 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.

\(^8\)https://keras.io/
\(^9\)We use the public code in provided by Nickel and Kiela (2017): https://github.com/facebookresearch/poincare-embeddings
Consider these available items

Cat7 Snagless Shielded (Stp/sftp) Ethernet Patch Cable in Yellow 10 Feet
by WELDED

2 ratings

Currently unavailable.
We don't know when or if this item will be back in stock.

- EFFECTIVE: Sunatoria Black Mask is the perfect blackhead remover for normal to oily skin; High-quality black charcoal is known to provide superior cleansing, blackhead removal and acne treatment.
- ABSOLUTELY SAFE: This charcoal peel off mask has undergone necessary testing with the FDA USA, and has available MSDS, GMP, ISO Certification information; The black charcoal face mask does not cause redness, allergic reactions or skin irritations.
- NATURAL ACTIVE INGREDIENTS: Black Mask is made of high quality all-natural ingredients including Oregon

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

<table>
<thead>
<tr>
<th>Attr.</th>
<th>Grocery Products</th>
<th>Baby Products</th>
<th>Beauty Products</th>
<th>Health Products</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flavor</td>
<td>OpenTag</td>
<td>4364</td>
<td>79.6</td>
<td>60.3</td>
</tr>
<tr>
<td></td>
<td>TXtract</td>
<td>8697</td>
<td>89.1</td>
<td>64.9</td>
</tr>
<tr>
<td>Scent</td>
<td>OpenTag</td>
<td>446</td>
<td>75.5</td>
<td>56.8</td>
</tr>
<tr>
<td></td>
<td>TXtract</td>
<td>565</td>
<td>87.4</td>
<td>61.2</td>
</tr>
<tr>
<td>Brand</td>
<td>OpenTag</td>
<td>5150</td>
<td>68.8</td>
<td>62.9</td>
</tr>
<tr>
<td></td>
<td>TXtract</td>
<td>6944</td>
<td>78.9</td>
<td>67.4</td>
</tr>
<tr>
<td>Ingred.</td>
<td>OpenTag</td>
<td>3402</td>
<td>82.5</td>
<td>40.5</td>
</tr>
<tr>
<td></td>
<td>TXtract</td>
<td>6155</td>
<td>87.3</td>
<td>43.1</td>
</tr>
</tbody>
</table>

### Table 7: Product category classification results

<table>
<thead>
<tr>
<th>MT type</th>
<th>Grocery Products</th>
<th>Baby Products</th>
<th>Beauty Products</th>
<th>Health Products</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AUPR</td>
<td>F1</td>
<td>Prec</td>
<td>Rec</td>
</tr>
<tr>
<td>flat</td>
<td>45.9</td>
<td>21.4</td>
<td>63.3</td>
<td>13.7</td>
</tr>
<tr>
<td>hierarchical</td>
<td>47.3</td>
<td>29.7</td>
<td>68.4</td>
<td>19.9</td>
</tr>
</tbody>
</table>

### 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).

<table>
<thead>
<tr>
<th>Model</th>
<th>TX</th>
<th>MT</th>
<th>Vocabulary</th>
<th>Coverage (%)</th>
<th>Micro-average</th>
<th>Macro-average</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenTag</td>
<td>-</td>
<td>-</td>
<td>6,756</td>
<td>73.2</td>
<td>57.5</td>
<td>54.6</td>
</tr>
<tr>
<td>Title+id</td>
<td>✓</td>
<td>-</td>
<td>6,400</td>
<td>69.1</td>
<td>55.7</td>
<td>53.3</td>
</tr>
<tr>
<td>Title+name</td>
<td>✓</td>
<td>-</td>
<td>5,328</td>
<td>70.6</td>
<td>56.9</td>
<td>54.2</td>
</tr>
<tr>
<td>Title+path</td>
<td>✓</td>
<td>-</td>
<td>4,608</td>
<td>64.6</td>
<td>54.3</td>
<td>51.9</td>
</tr>
<tr>
<td>Concat-wemb-Euclidean</td>
<td>✓</td>
<td>-</td>
<td>9,768</td>
<td>76.3</td>
<td>60.1</td>
<td>57.4</td>
</tr>
<tr>
<td>Concat-wemb-Poincaré</td>
<td>✓</td>
<td>-</td>
<td>8,684</td>
<td>74.3</td>
<td>60.6</td>
<td>57.7</td>
</tr>
<tr>
<td>Concat-LSTM-Euclidean</td>
<td>✓</td>
<td>-</td>
<td>9,255</td>
<td>75.9</td>
<td>60.1</td>
<td>57.5</td>
</tr>
<tr>
<td>Concat-LSTM-Poincaré</td>
<td>✓</td>
<td>-</td>
<td>8,893</td>
<td>75.2</td>
<td>60.8</td>
<td>57.9</td>
</tr>
<tr>
<td>Gate-Poincaré</td>
<td>✓</td>
<td>-</td>
<td>9,690</td>
<td>77.1</td>
<td>60.6</td>
<td>57.7</td>
</tr>
<tr>
<td>CondSelfAtt-Poincaré</td>
<td>✓</td>
<td>-</td>
<td>12,558</td>
<td>83.1</td>
<td>61.9</td>
<td>58.3</td>
</tr>
<tr>
<td>MT-flat</td>
<td>-</td>
<td>✓</td>
<td>8,699</td>
<td>72.2</td>
<td>60.9</td>
<td>57.8</td>
</tr>
<tr>
<td>MT-hier</td>
<td>-</td>
<td>✓</td>
<td>9,528</td>
<td>73.4</td>
<td>61.5</td>
<td>58.3</td>
</tr>
<tr>
<td>Concat &amp; MT-hier</td>
<td>✓</td>
<td>✓</td>
<td>9,316</td>
<td>74.6</td>
<td>62.3</td>
<td>59.0</td>
</tr>
<tr>
<td>Gate &amp; MT-hier</td>
<td>✓</td>
<td>✓</td>
<td>10,845</td>
<td>80.0</td>
<td>61.1</td>
<td>57.9</td>
</tr>
<tr>
<td>CondSelfAtt &amp; MT-hier (TXtract)</td>
<td>✓</td>
<td>✓</td>
<td>13,093</td>
<td>83.9</td>
<td>63.3</td>
<td>59.3</td>
</tr>
</tbody>
</table>