Enhancement and Analysis of TARS Few-shot Learning Model for Product Attribute Extraction from Unstructured Text

Pascual Martinez-Gomez, Giorgos Papachristoudis, Jon Blauvelt, Eric Rachlin, Saul Simhon
{gomepasc,geopap,jbblauve,eerac,ssimhon}@amazon.com
Amazon
USA

ABSTRACT

Few-shot learning techniques rely on generalizations of a base model trained on a large training set in order to transfer learn to more specialized tasks. Such techniques extend unsupervised learning models by allowing for fine-tuning a general model to a domain of interest using a relatively low number of training samples. This is especially important for applications where data may be non-homogeneous and difficult to label, such as product catalog data that vary drastically from category to category and require domain expertise for labeling.

In this paper, we explore the application of a few-shot learning model to infer structured product attribute values from unstructured descriptive text. We experiment with Task-Aware Representation of Sentences (TARS) model [3] applied to catalog data, and further extend the model to support training on negative examples. We demonstrate that with a few labeled examples, we can train a specialized model for each attribute of a product category, and provide results of comparable quality to current state-of-the art techniques. Results show that with only 40 training examples per product attribute, the model, when cross trained over attributes, produces improved recall and comparable precision over an existing baseline model that relies on tens of thousands of examples per product attribute. To further exemplify the generalization power, experiments are conducted using synthetic training data, extracted automatically from search queries. This completely eliminates the need for manually labeled examples and further leverages customer behavioral signals to help prioritize model training by what customers deem important.

1 INTRODUCTION

Customer experiences for eCommerce are subject to the quality and completeness of the underlying catalog data. A well-structured catalog allows for consistent and rich user experiences, including improved product discovery, product comparison and purchase decisions. Conversely, an inconsistent or incomplete catalog degrades user experience and increases buyer's regret.

Maintaining a robust structured catalog at scale is a challenging problem. First, due to the evolution of the market, the shape of structured catalog data, including the attribute definition and corresponding possible values, is never complete nor static. Second, the heterogeneous nature of the data from the thousands of different product categories poses a challenge in maintaining correctness over the diversity of structure and taxonomy. Finally, the structure of data from different vendors or manufacturers is often incomplete or inconsistent. This poses a dichotomy between frictionless integration of large amounts of product data and consistent buyer experiences for discovery, comparison, and product purchasing decisions.

A system that can automatically infer a consistent set of structured product facts from unstructured and diverse descriptive texts of those products is key to help address such issues. Product descriptive texts are readily available on many eCommerce sites, where important product facts are often expressed for marketing and SEO purposes. Translating such texts to consistent structured attribute values allows us to populate missing attributes at scale and validate if existing structured attributes and unstructured texts are self-consistent.

The challenge in developing such a system stems for the fact that product catalog data is not homogeneous. The vocabulary, semantics and attributes vary drastically in the intentionally stratified catalog data (from one category to another). For example, term *"HP"* may refer to the brand Hewlett-Packard for printer products or to the engine's horsepower for lawnmower products. Knowledge that *"glass"* or *"ceramic"* are potential materials for a drinking cup does not inform us what materials may exist for a shirt, nor whether the *collar style* is an applicable shirt attribute. Conversely, there are some commonalities that can be learned and shared across product categories. For example, *"red"* may be a valid color across multiple products. Learning a model or developing heuristics for predicting the correct attribute values requires labeling and analysis across each attribute of each product category, as well as recognizing those entities that can be shared across products. This becomes a tedious non-scalable and expensive task that often requires domain expertise.

In this paper, we explore the application of a few-shot learning model on catalog data. We demonstrate that with only a few training examples, we can train a robust attribute value prediction model that has similar performance to an existing state-of-the-art system [11] that requires orders of magnitude more training data. This allows us to quickly bootstrap incomplete or noisy catalog data with structured attribute values, reducing the time and cost of human labelers.

1.1 Overview

The problem of attribute value extraction is cast as a classification problem, where we aim to compute the probabilities over a set of possible canonical attribute values for a product given the product's unstructured descriptive text. We thus guarantee consistent attribute value extraction, a key property for the quality of catalog data. Using transfer learning over a BERT language model, we extrapolate a small human-created training set of unstructured text labeled with attribute-value tuples. We aim to classify attribute values for a new product beyond the trivial case; where the canonical values themselves may not explicitly appear anywhere in the product's descriptive texts, extrapolating on alternative or synonymous features that may be found in the text.

We first pretrain the model across all training examples (over all attributes) in order to learn common knowledge across attributes. We then fine tune a specialized model for each product attribute by training it only on the labeled data for the corresponding attribute and product category. We produce one model for each attribute/-category pair, irrespective to whether the attributes carry the same semantics across product categories. For example, we may train a model for the attribute *screen-size* for computer monitors, and likewise train a separate model for *screen-size* for laptop computers.

We accomplish this by leveraging the TARS model architecture [3]. The model consists of a deep network that generalizes a pre-trained NLP BERT model with support for fine-tuning for text classification tasks. We further extend TARS to support explicit use of negative examples in training data – the current TARS model only supports implicit negative examples that are computed internally. This improvement allows us to better detect non-applicable attributes, where the model should not predict any value for structured attributes that do not apply to the particular product.

Our approach further allows us to complement or completely replace human-labeled data with synthetic training data produced by rules, limiting the risks that the model may overfit. We experiment with training a model in a weakly supervised fashion by generating synthetic training data using matching rules between the text and the predefined dictionary of attribute values. Further, instead of using descriptive texts from catalog data, we experiment with search queries. This introduces attribute predictions from data that is based on customer behavior. For the purposes of having comparable results, we sample the synthetic training data from search queries following the same distribution of values seen in the human-labeled data for catalog text.

2 RELATED WORK

Early methods for attribute value extraction from unstructured text (or more generally, entity extraction) were primarily rule-based or statistics-based. Domain specific seed vocabularies were used and extrapolated to help detect related values seen in the corpus [2, 8, 9]. Such methods are limited in their coverage and quality. Other early NER-based methods attempted to learn sequential patterns from the corpus to provide wider coverage. Work by [5] trained an SVM classifier over a sequential representation model (Hidden Markov Model) of text. However, such methods lacked the expressive power of the underlying model to represent a wide variety of text, limiting the scope to consistently formulated product titles, tables, or similarly formed semi-structured texts.

More recent approaches to attribute value extraction leverage deep learning techniques to model and process sequences of tokens and their contexts. OpenTag [11] represents the attribute extraction problem as a sequence tagging problem and applies a BiLSTM-CRF network with attention layers to perform schema matching from free text. Similar approaches based on BiLSTM-CRF network were taken to extract product content from HTML documents [4]. A drawback of this work is the need to train on tens of thousands of examples. Alternative approaches aim to leverage existing learned models for reducing the training data and improving recall. Wang *et. al.* [10] develop a hybrid model that combines a pre-trained BERT model with a contextual BERT-style transformer layer for a QA-style attribute extraction system, where the attribute name and text provide the question and context respectively, and the predicted answer is the value. Their model was trained over 127K examples for 2.7K unique attributes and 10K unique values, and results demonstrated capability for predicting values for new attributes, never seen in training (zero-shot), with limited precision and recall.

Value extraction techniques such as these extract new or variant forms of the values found in unstructured texts. This is particularly useful for attribute discovery, though it can be problematic in applications for backfilling catalog data with consistent/canonical attribute values. Our work assumes that we are given the preferred canonical representation of all applicable attribute values *a-priori*, and our goal is to predict the correct associations to products, guaranteeing that the prediction results only produce catalog consistent values. We do not take the assumption that the literal canonical values can always be found in the products' unstructured descriptions, nor that the values are uniquely matched (for example, we may find both leather and cotton tokens in a product's descriptive texts, where one may refer to the inner material and the other to the outer material.) We therefore cast the problem as an NLP classification problem. We apply the TARS model [3] for fine-tuning a few-shot text classification model that is specialized for each attribute. We further extend [3] by allowing for explicit definition of negative training examples. Our approach complements the model's current data augmentation technique, where negative examples are extrapolated internally by computing in the embedding space the most similar labels to the positive training examples.

3 PROBLEM DEFINITION

Given unstructured descriptive text fragments corresponding to a product, we wish to predict the most suitable value for a given attribute for that product. Instead of attempting to extract a suitable value directly from text, we cast the problem as a classification problem. We first assume that we are given *a priori* a set of possible values (enumeration) that the attribute can take on. Such enumerations are readily available in existing (though incomplete) catalog, or in predefined catalog schema. We then assume that only one value can be assigned to an attribute. Finally, we do not assume that all attributes are applicable to all products, and thus a key property of the model is that it must not predict values for non-applicable attributes.

Our problem then becomes a problem of computing the probability distribution across all enumeration values for an attribute given a product's descriptive text. The enumeration value with maximum probability for each attribute is selected as the most suitable for that attribute. If all enumeration values have low probability (under some predefined threshold) or a predefined "unknown" label is predicted, then no value is selected and the attribute is assumed as not applicable.

Let $\mathcal{A}_z = \{a_1^z, a_2^z, \ldots, a_n^z\}$ denote the set of attributes under a product domain Z (i.e., a *product category*). The product categories define a class of products that share the same form and utility of, and are defined as part of the catalog schema. Each attribute a_i^z is defined as $a_i^z := \langle k(a_i^z), \mathcal{V}(a_i^z)\rangle$, where $k(a_i^z)$ is the attribute name and $\mathcal{V}(a_i^z) = v_1, v_2, \ldots, v_m$ is the set of unique attribute values which we refer to as *enumeration values*. We denote a product classified under product domain as q^z and a product's descriptive text as s_{q^z}. Our goal is to compute the probability distribution for all enumeration values given the descriptive text and the attribute in the product domain: $P(v_j|s_{q^z}, a_i^z)$.

4 APPROACH

Our overall approach to the few-shot learning problem is to transfer learn an NLP model to the specialized task of classifying attribute values for a given attribute. The premise is that with the right language model, we can simulate a human's or domain expert's abilities of inferring structured attribute values by simply reading product descriptive texts. Recent approaches for text classification tasks with transfer learning comprise architectures with a base language model, typically BERT-style transformer architecture, followed by a dense classification layer. The dimensionality of the classification layer, and the fine-tuning of the entire model is optimized specifically for one task. A new task generally involves a new classification layer configured with different labels and dimensionality and trained only for the new task. While such an approach in beneficial to transfer learning from the base language model to a single specialized task, however, there is no clear way to share the knowledge from one task to the next. In our goal, we recognize that there is cross-information sharing between attributes and across product categories. We therefore wish to first leverage the training data across all attributes, while also specializing the model to a specific attribute. For this purpose, TARS (Task-Aware Representation of Sentences) model is an ideal candidate for this problem.

4.1 TARS Architecture

The TARS architecture provides two key properties that are relevant to our problem. First, the model allows us to pre-train across multiple tasks, and leverage knowledge earned from one product attribute or category to the next. If there are enough similarities between tasks (for example, the color attribute across certain product categories, or RAM memory between smartphones and tablets), then the model can provide zero-shot learning for one task given the learnings from another. Second, the model leverages the semantics of the labels explicitly. This is a key feature for our problem where the semantic similarity of attribute values plays a significant role in recognizing the correspondence of text to the canonical values.

Instead of a traditional dense layer for classification with a predefined task-specific dimensionality based on the labels, the model transforms the multi-class classification problem to m binary classification problems, where m are the different classes (labels) for the specific task. Predicting the probability of each class separately allows us to compute an approximation of the distribution across all labels:

$$f_{a^z}(v_j, s_{q^z}) = P(\mathtt{True}|(v_j, a^z, s_{q^z})), \forall j \in 1, \ldots, m, \qquad (1)$$

where s_{q^z} is the descriptive text of a product under domain Z, a^z is a specific attribute and v_j are all possible enumeration values for this attribute.

Internal to the TARS model, m binary examples are generated from each original example. For example, if value v_j appears in text s_{q^z} for attribute a^z, TARS will create m binary examples of the following form: $(v_1\mathtt{[SEP]}s_{q^z}, 0), \ldots, (v_{j-1}\mathtt{[SEP]}s_{q^z}, 0), (v_j\mathtt{[SEP]}s_{q^z}, 1), (v_{j+1}\mathtt{[SEP]}s_{q^z}, 0), \ldots (v_m\mathtt{[SEP]}[s_{q^z}, 0)$. As we can see, value v_j becomes part of training input and label represents whether variable v_j appears in text s_{q^z}.

In practice, m can be prohibitively large. To save computational resources, TARS randomly generates a few examples to form the "negative" set. This number is typically between 2–5 due to the large underlying architecture and the slow processing times per example. TARS performs a "smart" selection of the negative examples by selecting labels that are closest to the positive label. This way, there is a higher chance that the classifier would separate successfully the positive from the negative labels (for a given attribute). This idea stems from face detection literature and the Siamese Network architecture [7], where in order to build a successful face detection model, the negative examples are selected as close as possible to the positive example so as the classifier learns to be more specific about what constitutes a positive example. The tradeoff is that inference must be performed for each value to compute the full distribution.

4.2 Explicit Negative Examples

The original TARS algorithm makes the underlying assumption that each training example is assigned to at least one positive label. In our case, we experiment with a test schema where $\sim 60\%$ of the products have at least one randomly selected non-applicable attribute. That is, the attribute under consideration does not describe any valid product fact of the product in question. For example, attribute *topstyle* is not applicable for all swimwear products, but only women's swimwear. In other words, we often know the products that are not applicable for a given attribute rather than the correct attribute value for a given product. In these cases, we want to have a way to force the algorithm to learn specific negative labels for an example.

We propose an expansion of the original TARS algorithm that takes negative feedback as well. Each example is assigned to a number of positive and negative labels. Assuming that a product q^z has positive labels denoted by $\mathcal{P}$ and negative labels denoted by $\mathcal{N}$, this information can be represented as $(v_j\mathtt{[SEP]}s_{q^z}, 1), \forall j \in \mathcal{P}$, $(v_j\mathtt{[SEP]}s_{q^z}, 0), \forall j \in \mathcal{N}$. We construct a modified version of TARS model that takes both positive and negative feedback by converting the original example to $(v_j\mathtt{[SEP]}s_{q^z}, 1), \forall j \in \mathcal{P}$, $(v_k\mathtt{[SEP]}s_{q^z}, 0), \forall v_k \in \text{neighbor}(v_j) \forall j \in \mathcal{P}$, $(v_j\mathtt{[SEP]}s_{q^z}, 0), \forall j \in \mathcal{N}$. Here, we denote by $\text{neighbor}(v_j)$ the two closest neighbors of a positive label in terms of cosine similarity between the embeddings of label names. With this modification, we force the algorithm to not only learn positive labels but also negative labels attached to specific text.

5 TRAINING

5.1 TARS Base Model

The TARS base model is a BERT model (*bert-base-uncased* [1]) that is pretrained on BookCorpus, a dataset consisting of 11,038 unpublished books, and English Wikipedia (excluding lists, tables and headers). The model is then cross-task pretrained using a variety of datasets for different classification tasks, including Amazon reviews, Yelp reviews, DBPedia, and several others [3]. Certain abbreviated task labels have been extrapolated to help capture their semantics (e.g., "Sci/Tech" is renamed to "Science Technology", and ratings of 1–5 are named based on sentiment tokens). The resulting model is one that encodes language representation for a variety of sentiment and categorical classification tasks.

5.2 Cross-Attribute Training

The key advantage of TARS is that the model (encoder and decoder) can be shared across tasks, and that each task can transfer learn from the other [3]. Training a new task is equivalent to continuing the training from a previous task, leveraging the knowledge of the data in a previous task that may be applicable to the new task. We therefore cross-train a base TARS model using all attribute/category pairs.

Each training example consists of the descriptive text (a concatenation of product title and bullet points found in eCommerce sites), the attribute (e.g. *material*), and the correct attribute value (e.g. *"glass"*). The descriptive text is truncated after 512 words to satisfy the BERT model length constraint. We cross-train the model using 10 episodes (iterations) and only 40 examples for each attribute/category pair (interlaced). Each episode trains using one epoch for each pair.

5.3 Attribute-Specialized Training

Once a base model is computed, we train a specialized model specific to the attribute/category pair. This is accomplished by taking the set of 40 training examples corresponding to an attribute/category pair and training over the base model using 20 epochs. While the examples have already been visited when cross-training the model, the 20-epoch retraining with data only relevant to the attribute in question further fine-tunes the model.

5.4 Human Annotated Training Set

We use (product text, attribute, value) triplets manually annotated by humans on random uniform samples of popular consumable products. Humans were provided instruction to examine products' text profile (title, description, bullet points), images, and apply their own knowledge of the products to make the annotation decisions; moreover, humans were also allowed to not annotate values if they could not find them, or deemed the attribute inapplicable. We include these triplets in our training and evaluation and count as a precision miss if the model produces an attribute value when humans could not find a suitable value.

We collect approximately 14,000 examples with at least 190 examples per attribute/category pair. For each, a random sample of 40 examples are used for training and 150 for validation. In total, we have obtained data for 51 attribute/category pairs, over 39 unique product categories and 8 unique attributes.

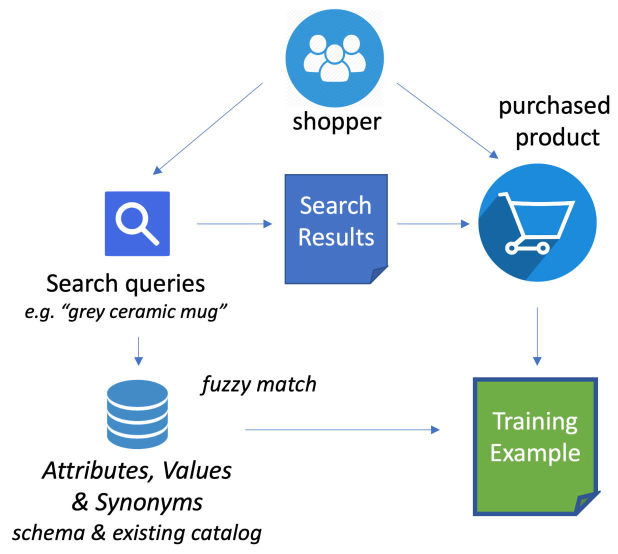


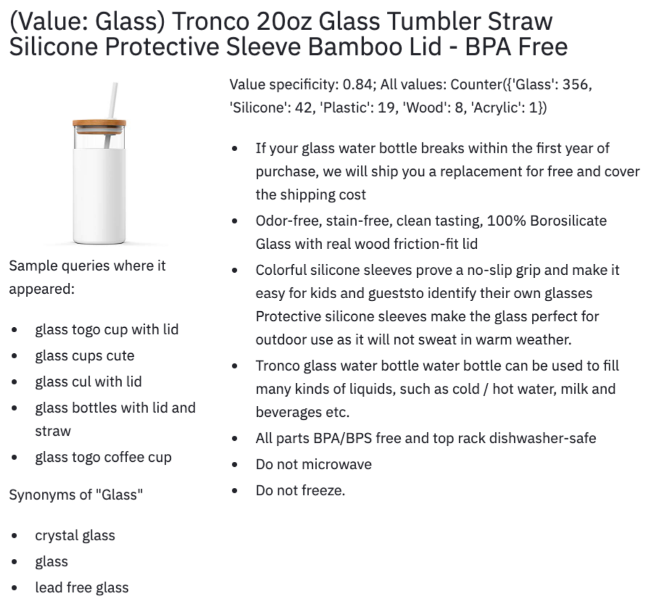
Figure 1: Overall flow for generating synthetic training examples using customer search queries.

5.5 Synthetic Training Set from Customer Signals

We explore a weakly supervised approach, where high quality (product text, attribute, value) triplets are identified from customer behaviors when searching and purchasing products. This allows us to reduce the human annotation effort to only those attribute/product categories where there is not enough customer signals to automatically generated the data, or where negative examples are required. It also allows us to prioritize model training to data that customers care about, an aspect not available directly when only using catalog data.

To that end, we collect an experimentation data set of search queries that lead directly to product purchases, and associate those queries to the product category corresponding to the products that were purchased (Figure 1). For example, the query *"55 inch 4K samsung tv"* may lead to a purchase of a television. From the corresponding product category, we first identify from the schema and existing catalog all the attributes and their associated enumerations (possible values). For each attribute, we then simply apply regular expression matching rules between the enumerations to the query text. While this type of training data only captures token matches to canonical values, we rely on the BERT model to learn and extrapolate the semantics and context from these exact match examples. The tradeoff in this approach however is that we are not able to distinguish between attributes that carry the same or similar potential values (e.g., interior color versus exterior color).

Customers that select the product act as the validators for the correct property of that product, as related to their search query. The correctness on a case-by-case basis is therefore limited by the search results quality as well as the overall customer experience. For example, customers may search for a *plastic* drinking-cup and end clicking on and purchasing a *glass* drinking-cup, either because



(Value: Glass) Tronco 20oz Glass Tumbler Straw Silicone Protective Sleeve Bamboo Lid - BPA Free

Value specificity: 0.84; All values: Counter({'Glass': 356, 'Silicone': 42, 'Plastic': 19, 'Wood': 8, 'Acrylic': 1})

- If your glass water bottle breaks within the first year of purchase, we will ship you a replacement for free and cover the shipping cost
- Odor-free, stain-free, clean tasting, 100% Borosilicate Glass with real wood friction-fit lid
- Colorful silicone sleeves prove a no-slip grip and make it easy for kids and gueststo identify their own glasses Protective silicone sleeves make the glass perfect for outdoor use as it will not sweat in warm weather.
- Tronco glass water bottle water bottle can be used to fill many kinds of liquids, such as cold / hot water, milk and beverages etc.
- All parts BPA/BPS free and top rack dishwasher-safe
- Do not microwave
- Do not freeze.

Sample queries where it appeared:

- glass togo cup with lid
- glass cups cute
- glass cul with lid
- glass bottles with lid and straw
- glass togo coffee cup

Synonyms of "Glass"

- crystal glass
- glass
- lead free glass

Figure 2: Synthetic training data inferred from a test sample of search queries leading to a drinking cup purchase. This product, from the drinking cup category, is automatically annotated with the value *"glass"* for attribute *material.*

they changed their minds or because the product information was incorrect. To avoid generating erroneous triplets, we set a minimum of 5 purchases per triplet. Moreover, we require a product-value pair to be associated with at least 80% of the queries that match to values for the same attribute. Figure 2 shows an example for the attribute *material*, where the matching value *"glass"* has occurred in 356 queries whereas the matching value *"silicone"* appeared in 42 queries. Hence, the value *"glass"* accounts for the majority (84% of the queries) and it is selected to be the value for this product-attribute.

We observe in this data that the label distribution for each attribute/category pair is different to the human-labeled training data, which is where the model will be evaluated. For that reason, we consider three different settings:

(1) Use a random sample of search-derived (product-text, attribute, value) triplets, up to a maximum of 40 examples.
(2) Unbias the search-derived triplets by doing a weighted sampling with replacement to match to the label distribution of the evaluation data. In a real scenario, this resampling would only be possible if either we have an evaluation set or we know the real frequencies of values in each attribute/-category pair, e.g. by collecting value frequencies from an existing catalog.
(3) The same as (2) above, with the addition of negative examples for non-applicable attributes to help TARS learn when to supress predictions.

Method	Precision	Recall	F_1 score
OpenTag	.84	.65	.72
TARS w/o cross-train	.75	.64	.67
TARS with cross-train	.84	.72	.76

Table 1: Results for 39 categories and 8 attributes. OpenTag trained on an average of 27,000 examples by weak supervision, TARS trained on a disjoint split of 40 examples from the gold dataset (manually annotated). *TARS w/o cross-train* denotes TARS with only fine-tuning the base model on 40 examples for each attribute (separately). *TARS with cross-train* denotes TARS including cross-attribute pre-training prior to fine-tuning.

6 RESULTS

Precision, recall and F_1 score as evaluation metrics. We compare results with OpenTag [11], trained on an average of 27K examples obtained with weak supervision using Snorkel [6]. When OpenTag produces a predicted label that may not be the canonical representation (e.g. in open-vocabulary attributes), we keep the prediction label "as-is". The evaluation uses a fuzzy token overlaps between such predicted values and gold values (including potential synonyms); if there is at least one token overlapping, then it counts as a correct prediction for OpenTag.

6.1 Quantitative Evaluation

Table 1 shows results of TARS with and without cross-task training. As we observe, precision is significantly lower than OpenTag (TARS precision 75% versus OpenTag 84%) when there is no preliminary cross-task training. However, TARS significantly increases its precision and recall when we do preliminary cross-task training on the disjoint training splits of the gold data (TARS precision 84% versus OpenTag 84%; TARS recall 72% versus OpenTag 65%) which highlights the importance of models that can be trained across product attributes/categories to learn the commonalities.

Table 2 shows the top 10 attribute/category pairs for which TARS performs best when compared to OpenTag. In most of those cases, TARS recall is higher than OpenTag. We believe this is a result of leveraging training data from other attribute/category pairs during the cross-task training stage, as well as having a label distribution in the training data that better resembles that of the evaluation set (it is a random disjoint split of 40/150 training and evaluation).

Table 3 shows the top 10 attribute/category pairs for which TARS performs worst when compared to OpenTag. In most of those cases, TARS recall is much lower than OpenTag. Examining the individual evaluation examples, we observe that these attribute/category pairs have a high rate of missing values in the gold data, either because the value is not apparent and humans left them blank, or because the attribute is inapplicable to those products. For that reason, TARS tends to predict the label *unknown* which we use to suppress TARS prediction, hence affecting recall.

Product Category	attribute	TARS F1	OT F1	TARS Prec.	OT Prec.	TARS recall	OT recall	F1 diff
Sugar	item form	0.84	0.38	0.89	0.43	0.79	0.34	+0.45
Insect Repellent	item form	0.84	0.43	0.87	0.82	0.81	0.29	+0.41
Tooth brush	age range	0.96	0.59	0.94	0.85	0.98	0.45	+0.37
Lip Balm	skin type	0.69	0.40	0.75	0.71	0.63	0.28	+0.29
Tooth Whitener	flavor	0.84	0.58	0.93	0.81	0.76	0.45	+0.26
Vitamin	age range	0.96	0.72	0.96	0.96	0.97	0.58	+0.24
Sunscreen	age range	0.92	0.70	0.91	0.94	0.92	0.56	+0.22
Snack food	age range	0.94	0.73	0.95	0.93	0.93	0.60	+0.21
Pill Dispenser	material	0.75	0.55	0.73	0.94	0.77	0.39	+0.20
Cereal	item form	0.89	0.70	0.93	0.89	0.85	0.57	+0.19

Table 2: Top 10 attribute/category pairs for which TARS performs better than OpenTag (OT). In most of these cases, TARS produces a significantly higher recall which influences F_1 score. We believe TARS produces higher recall because it is able to better generalize by learning from other product categories and attributes during cross-task training.

Product Category	attribute	TARS F1	OT F1	TARS Prec.	OT Prec.	TARS recall	OT recall	F1 diff
Skin Moisturizer	scent	0.16	0.57	0.40	0.50	0.10	0.65	−0.41
Eyelid Color	skin type	0.51	0.80	0.82	0.77	0.38	0.83	−0.29
Mascara	skin type	0.68	0.95	0.91	0.92	0.55	0.98	−0.27
Skin Concealer	skin type	0.48	0.74	0.76	0.76	0.35	0.71	−0.26
Sunscreen	scent	0.30	0.52	0.75	0.86	0.19	0.38	−0.22
Nail Polish	finish type	0.42	0.62	0.61	0.91	0.32	0.47	−0.20
Face Makeup	skin type	0.52	0.71	0.84	0.69	0.38	0.72	−0.19
Makeup PrimerR	skin type	0.57	0.72	0.67	0.69	0.50	0.75	−0.15
Eyelid Color	finish type	0.50	0.65	0.52	0.73	0.49	0.58	−0.14
Sunscreen	item form	0.77	0.91	0.72	0.92	0.82	0.89	−0.14

Table 3: Top 10 attribute/category pairs for which TARS performs worse than OpenTag. In most of these cases, TARS produces a significantly lower recall which influences F_1 score. This is caused by having large proportions of (product text, attribute, value) triplets where the value is judged to be absent by humans, either because it is not explicit in the product description and image or because the attribute is not applicable or common to those products.

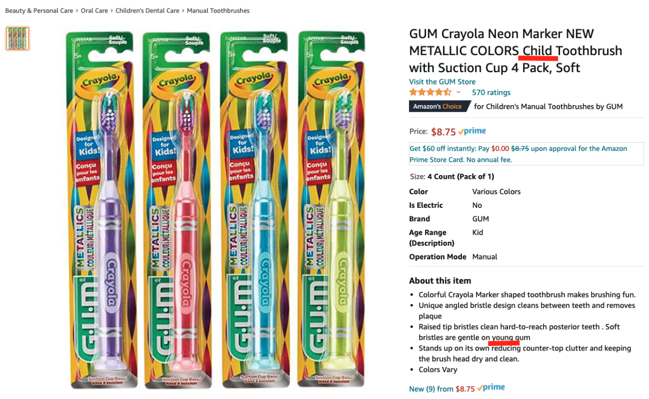
Table 4 shows TARS results when trained with at most 40 (product text, attribute, value) triplets obtained from search queries as described in Section ??. When using a uniform sample to construct the training data, TARS precision is much lower than OpenTag's (56% vs. 83%). Both precision and recall increase when we unbias the training set by sampling with replacement to match the distribution of the benchmark set: precision increases to 59% from 56% and recall increases to 76% from 66%. From search queries, we are unable to extract plausible negative examples at this stage; however, we can still measure the upper bound in performance if we were able to produce these negative examples. To that end, we sample examples from the evaluation set that have no label. When introducing these examples in the training set, precision has a significant increase (82% from 59%), recall decreases and F_1 score increases, producing comparable if not better results than OpenTag.

This evidences the importance of unbiasing the label distribution when using (product text, attribute, value) triplets obtained from search data and the need to produce good negative examples for TARS to learn when not to predict a label. For example, for the category Eyelid Color and attribute *age range*, most (product text, attribute, value) triplets that we extracted from search logs had the value *"youth"* whereas most products annotated by humans had the value *"adult"*; this mismatch in the label distribution is likely caused by customers omitting the keyword *"adult"* when searching for eyelid color products because that may be implicitly understood.

Method	Prec	Recall	F_1 score
OpenTag	.83	.62	.70
TARS uniform set	.56	.66	.57
TARS unbiased set	.59	.76	.63
TARS unbiased set w. *unk*	.82	.68	.72

Table 4: TARS trained on (product text, attribute, value) triplets extracted from customer behavior in search logs. This evaluation is only on 30 attribute/category pairs which is a subset of the data in Table 1 for which we have collected triplets from search queries. *TARS uniform set* denotes TARS trained using at most 5 products per value, up to a maximum of 40 examples. *TARS unbiased set* denotes TARS trained using at most 40 products, using the same label distribution as it appears in the evaluation set. *TARS unbiased set w. unk* denotes TARS trained using at most 40 products, using the same label distribution as it appears in the evaluation set and introducing examples from the evaluation set for which TARS should not predict values.

Lastly, we show an experiment where we compare default TARS against our modified version where we incorporate explicitly negative feedback. We obtain negative feedback from examples that human evaluators marked as non-applicable; in other words, a product that a human evaluator decided that a specific attribute does not apply to it. We train the modified version of TARS with negative



Beauty & Personal Care › Oral Care › Children's Dental Care › Manual Toothbrushes

GUM Crayola Neon Marker NEW METALLIC COLORS Child Toothbrush with Suction Cup 4 Pack, Soft

Visit the GUM Store

570 ratings

Amazon's Choice for Children's Manual Toothbrushes by GUM

Price: $8.75 prime

Get $60 off instantly: Pay $0.00 ~~$8.75~~ upon approval for the Amazon Prime Store Card. No annual fee.

Size: **4 Count (Pack of 1)**

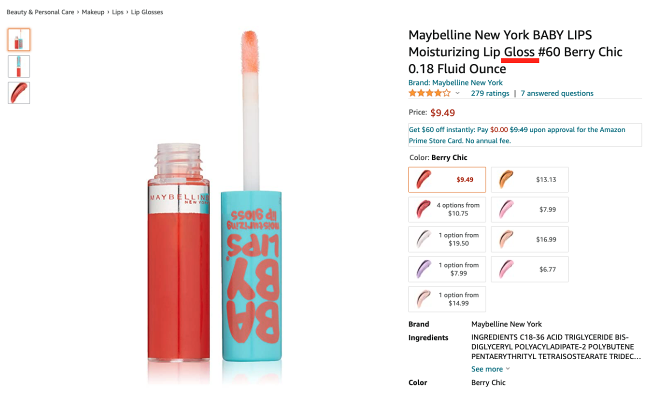
Color	Various Colors
Is Electric	No
Brand	GUM
Age Range (Description)	Kid
Operation Mode	Manual

About this item

- Colorful Crayola Marker shaped toothbrush makes brushing fun.
- Unique angled bristle design cleans between teeth and removes plaque
- Raised tip bristles clean hard-to-reach posterior teeth . Soft bristles are gentle on young gum
- Stands up on its own reducing counter-top clutter and keeping the brush head dry and clean.
- Colors Vary

New (9) from $8.75 prime

Figure 3: In this example, TARS predicted the label *"kid"* for attribute *age range* even if that word was not present in the title nor the bullet points; instead, the cues are *"child"* and *"young"* which TARS managed to relate to *"kid"*. The words *"kid"* and *"child"* are typical synonyms and they appear in WordNet.



Beauty & Personal Care › Makeup › Lips › Lip Glosses

Maybelline New York BABY LIPS Moisturizing Lip Gloss #60 Berry Chic 0.18 Fluid Ounce

Brand: Maybelline New York

279 ratings | 7 answered questions

Price: $9.49

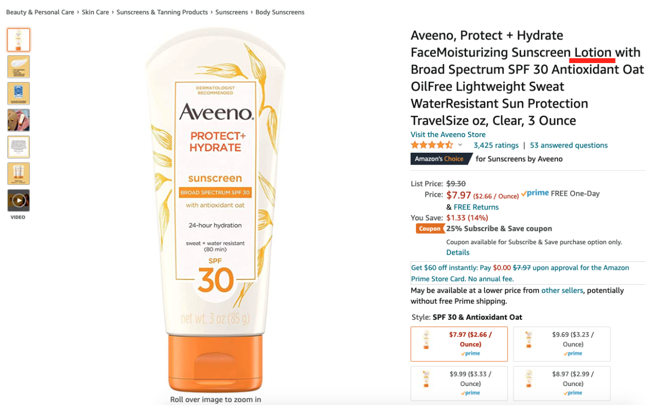
Get $60 off instantly: Pay $0.00 ~~$9.49~~ upon approval for the Amazon Prime Store Card. No annual fee.

Color: **Berry Chic**

$9.49	$13.13
4 options from $10.75	$7.99
1 option from $19.50	$16.99
1 option from $7.99	$6.77
1 option from $14.99	

Brand	Maybelline New York
Ingredients	INGREDIENTS C18-36 ACID TRIGLYCERIDE BIS-DIGLYCERYL POLYACYLADIPATE-2 POLYBUTENE PENTAERYTHRITYL TETRAISOSTEARATE TRIDEC... See more
Color	Berry Chic

Figure 4: In this example, TARS predicted *"glossy"* for attribute *finish type* while the text cue was *"gloss"*, a morphologically related word. The words *"glossy"* and *"gloss"* can be derived from each other using heuristic morphological analyzers.



Beauty & Personal Care › Skin Care › Sunscreens & Tanning Products › Sunscreens › Body Sunscreens

Aveeno, Protect + Hydrate FaceMoisturizing Sunscreen Lotion with Broad Spectrum SPF 30 Antioxidant Oat OilFree Lightweight Sweat WaterResistant Sun Protection TravelSize oz, Clear, 3 Ounce

Visit the Aveeno Store

3,425 ratings | 53 answered questions

Amazon's Choice for Sunscreens by Aveeno

List Price: ~~$9.30~~
Price: $7.97 ($2.66 / Ounce) prime FREE One-Day & FREE Returns
You Save: $1.33 (14%)
Coupon 25% Subscribe & Save coupon
Coupon available for Subscribe & Save purchase option only. Details

Get $60 off instantly: Pay $0.00 ~~$7.97~~ upon approval for the Amazon Prime Store Card. No annual fee.

May be available at a lower price from other sellers, potentially without free Prime shipping.

Style: **SPF 30 & Antioxidant Oat**

$7.97 ($2.66 / Ounce) prime	$9.69 ($3.23 / Ounce) prime
$9.99 ($3.33 / Ounce) prime	$8.97 ($2.99 / Ounce) prime

Roll over image to zoom in

Figure 5: In this example, TARS predicted *"liquid"* for attribute *item form* while the text cue was *"lotion"*, which is a synonym in this context. The words *"liquid"* and *"lotion"* are not so obviously related but TARS managed to learn the relationship for this attribute/category pair.

feedback, denoted by TARS^{-}, for six episodes and we compare the results against default TARS on the same test data for 52 attribute/-category pairs. The results are shown on Table 5. TARS^{-} shows an improvement by 3% on F1 score compared to default TARS and it especially outperforms on attribute/category tasks with large presence of non-applicable attribute examples.

6.2 Qualitative Evaluation

In approximately 30% of TARS predictions, the predicted label does not explicitly appear in the products' descriptive texts. Instead, some *text cues* signal the presence of a certain attribute value which TARS uses to produce higher scores for labels that are semantically related. For example, Figure 3 shows how TARS identifies *"child"* and *"young"* to be related to the label *"kid"* since they are synonyms, Figure 4 shows how TARS recognizes *"gloss"* to be morphologically related to *"glossy"* and Figure 5 shows how TARS identifies *"lotion"* to be a synonym to *"liquid"* in the context of sunscreens.

7 CONCLUSION

In this work, a novel approach for inferring attribute values of products from unstructured descriptive text was presented. We demonstrated that we can reformulate the attribute extraction problem as a few-shot classification problem, enabling inference of canonical attribute values with a few examples. This allows us to populate structured attributes in a catalog with consistent representations of the values, improving catalog quality. We further demonstrated that we can train a model with only 40 human labeled samples per attribute/category pair and, when cross trained, produce comparable results to a leading method that uses, on average, 27K examples per model in a weakly supervised fashion. This was accomplished by applying an updated version of the TARS model, with support for negative examples. We demonstrated that when we use negative feedback the model better learns to capture examples where an attribute is not applicable for a product. Finally, we demonstrated that we can use customer signals in a weakly supervised fashion to train a model with comparable performance.

The specific parameters and number of training examples selected in our work were based on qualitative empirical analysis. Future extension to this work may include an active learning layer to optimize the number of examples for each model. Using online or active learning techniques, we can consider feedback loops to pinpoint which attribute values may require more examples. A metric on the entropy of the prediction can help guide the confidence of the model for each value. Additional extension to this work can include support for image data; integrating the text classification model with image features may further reinforce the quality, or conversely, may help identify inconsistencies between the semantics of the text and the posted images.

Product Category	attribute	TARS F1	TARS$^-$ F1	TARS Prec.	TARS$^-$ Prec.	TARS recall	TARS$^-$ recall	F1 diff
Dental Floss	flavor	0.91	0.93	0.92	0.93	0.91	0.92	+0.02
Lipstick	finish type	0.72	0.64	0.79	0.74	0.73	0.66	−0.08
Mascara	skin type	0.86	0.86	0.87	0.87	0.87	0.85	0
Honey	item form	0.90	0.92	0.90	0.93	0.91	0.93	+0.02
Nail Polish	finish type	0.54	0.51	0.62	0.57	0.57	0.51	−0.03
Tooth Whitener	flavor	0.94	0.94	0.94	0.94	0.94	0.95	0
Eyelid Color	age range	0.13	0.64	0.08	0.62	0.29	0.67	+0.51
all single pt, attribute pairs		0.78	0.81	0.83	0.82	0.80	0.81	+0.03

Table 5: We pick 7 attribute/category pairs randomly and we compare between default TARS and TARS where negative feedback is incorporated (TARS$^-$). We see that when negative feedback is incorporated we achieve a 3% increase on F1 score on average across all attribute/category pairs. Even though both models perform similarly across attribute/category pairs, TARS$^-$ does better on attribute/category pairs where there are many not applicable values (i.e., age range/eyelid color) pair). This explains the boost in F1 score on average.

REFERENCES

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805

[2] Rayid Ghani, Katharina Probst, Yan Liu, Marko Krema, and Andrew Fano. 2006. Text mining for product attribute extraction. *ACM SIGKDD Explorations Newsletter* 8, 1 (2006), 41–48.

[3] Kishaloy Halder, Alan Akbik, Josip Krapac, and Roland Vollgraf. 2020. Task-Aware Representation of Sentences for Generic Text Classification. In *Proceedings of the 28th International Conference on Computational Linguistics*. 3202–3213.

[4] Bill Yuchen Lin, Ying Sheng, Nguyen Vo, and Sandeep Tata. 2020. FreeDOM: A Transferable Neural Architecture for Structured Information Extraction on Web Documents. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20)*. 1092–1102.

[5] Duangmanee Putthividhya and Junling Hu. 2011. Bootstrapped named entity recognition for product attribute extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. 1557–1567.

[6] Alexander Ratner, Stephen H. Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid Training Data Creation with Weak Supervision. *Proc. VLDB Endow.* 11, 3 (Nov. 2017), 269–282.

[7] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. FaceNet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Jun 2015). https://doi.org/10.1109/cvpr.2015.7298682

[8] Keiji Shinzato and Satoshi Sekine. 2013. Unsupervised extraction of attributes and their values from product description. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*. 1339–1347.

[9] Damir Vandic, Jan-Willem Van Dam, and Flavius Frasincar. 2012. Faceted product search powered by the Semantic Web. *Decision Support Systems* 53, 3 (2012), 425–437.

[10] Qifan Wang, Li Yang, Bhargav Kanagal, Sumit Sanghai, D Sivakumar, Bin Shu, Zac Yu, and Jon Elsas. 2020. Learning to Extract Attribute Value from Product via Question Answering: A Multi-task Approach. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20)*. 47–55.

[11] 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* (London, United Kingdom) *(KDD '18)*. 1049–1058.