OA-Mine: **Open-World Attribute Mining for E-Commerce Products with Weak Supervision**

Xinyang Zhang[1], Chenwei Zhang[2], Xian Li[2], Xin Luna Dong[3]
Jingbo Shang[4], Christos Faloutsos[5], Jiawei Han[1]
[1]University of Illinois at Urbana-Champaign [2]Amazon.com, Inc. [3]Meta (Facebook)
[4]University of California, San Diego [5]Carnegie Mellon University
[1]{xz43,hanj}@illinois.edu [2]{cwzhang, xianlee}@amazon.com
[3]lunadong@fb.com [4]jshang@ucsd.edu [5]christos@cs.cmu.edu

ABSTRACT

Automatic extraction of product attributes from their textual descriptions is essential for online shopper experience. One inherent challenge of this task is the emerging nature of e-commerce products — we see new types of products with their unique set of new attributes constantly. Most prior works on this matter mine new values for a set of known attributes but cannot handle new attributes that arose from constantly changing data. In this work, we study the attribute mining problem in an open-world setting to extract novel attributes and their values. Instead of providing comprehensive training data, the user only needs to provide a few examples for a few known attribute types as weak supervision. We propose a principled framework that first generates attribute value candidates and then groups them into clusters of attributes. The candidate generation step probes a pre-trained language model to extract phrases from product titles. Then, an attribute-aware fine-tuning method optimizes a multitask objective and shapes the language model representation to be attribute-discriminative. Finally, we discover new attributes and values through the self-ensemble of our framework, which handles the open-world challenge. We run extensive experiments on a large distantly annotated development set and a gold standard human-annotated test set that we collected. Our model significantly outperforms strong baselines and can generalize to unseen attributes and product types.

CCS CONCEPTS

• **Information systems** → **Web mining**.

KEYWORDS

Open-world product attribute mining, weak supervision.

ACM Reference Format:
Xinyang Zhang[1], Chenwei Zhang[2], Xian Li[2], Xin Luna Dong[3], Jingbo Shang[4], Christos Faloutsos[5], Jiawei Han[1]. 2022. OA-Mine: Open-World Attribute Mining for E-Commerce Products with Weak Supervision. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3485447.3512035

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France

ACM ISBN 978-1-4503-9096-5/22/04…$15.00
https://doi.org/10.1145/3485447.3512035

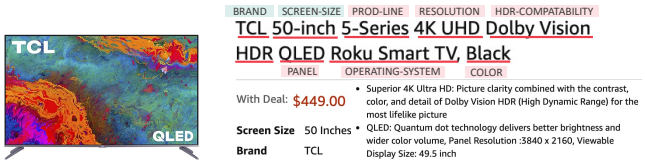


Figure 1: An example product title with known attributes annotated in green and new attributes in red. Sellers usually pack important product attribute values in the title for maximum exposure.

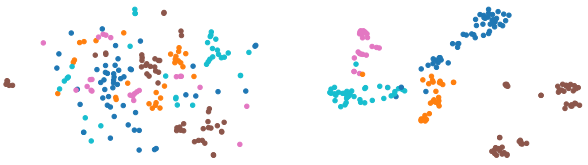
1 INTRODUCTION

Product attributes are essential for online shoppers, as they provide valuable information to help search, recommendation, and product comparison. The massive, constantly changing products bring the open-world challenge. Products we already know may gain new attributes over time – "HDR compatibility" barely comes to mind when people buy a TV ten years ago, but is relevant for many shoppers nowadays. Not to mention, new types of products with distinctively new sets of attributes may emerge in the future. Automatic extraction of product attributes has to meet the open-world challenge. The model must be able to discover both *attribute types* (e.g., brand, referred to as "attributes" hereafter) and *attribute values* (e.g., Samsung, referred to as "values"). Moreover, they must operate with limited supervision, as human annotation can never keep up with the forever-expanding product catalog.

Existing works on attribute mining mostly carry a closed-world assumption on attributes. Pioneer studies such as OpenTag [22] bear "open" in the name, but their open-world assumption is on *values only*. They assume the set of attributes of interest is given as input to the model. The model is trained to find more values for the given attributes, but does not expand to unseen attributes without additional training data. Moreover, they usually rely on extensive human labeling [22] or noisy distantly supervised data [14, 18, 19].

In this paper, we study the open-world attribute mining problem with weak supervision. We let the user provide a few example values for a few known attributes. Neither the attributes nor the values given as examples are required to be comprehensive. In addition, we also assume the type of the products are known a priori, as each product type holds a distinct set of applicable attributes (e.g., "resolution" applies to TVs but not shoes), and the attribute mining process should be done with respect to each product type. With limited supervision and abundant product raw text, we aim to discover new attributes and values for products of different types.

Product titles, as the cleanest and most prominent part of product text, possess a number of unique properties. As shown in Figure 1, product titles are concise, non-repeating, not written in a way as a



(a) BERT trained w/ in-domain corpus (b) After attribute-aware fine-tuning

Figure 2: T-SNE visualization of attribute value embedding. Points colored by attribute. After our attribute-aware fine-tuning, embedding is more reflective of attribute type.

general domain natural language. More specifically, we highlight three observations from the data. First, to maximize exposure of products to customers, sellers usually pack the highlights of their product in the title (observation 1: *title first*). Second, a product title rarely contains irrelevant information, and is a collection of attribute values (observation 2: *bag-of-values*). Third, with limited space in the title, the values seldom repeat (observation 3: *value exclusiveness*). For example, if one word describes the resolution of a TV product, the rest of the title will mention other attributes such as screen size. These observations pave the way for a mining-based approach that exploits the concise product titles, is tailored to the language of products, and discovers new attributes and values in an open-world setting.

We propose OA-Mine, a principled framework for **O**pen-world product **A**ttribute **MIN(E)**ing with weak supervision. Our framework has two main steps, *candidate value generation* and *attribute value grouping*. The first step of our framework is designed based on the *title first* and the *bag-of-values* data observations. It probes an in-domain fine-tuned language model that captures compositional relations between words, and segments product titles into phrases. These phrases are candidates that will either be marked as an attribute value or be excluded as noise in our subsequent step.

Once we obtain the attribute value candidates, our framework finds the major attributes for each product type, and puts the candidates into different attribute groups. We rely on a proper value embedding to propagate supervisions from seed values to other values when they together form a clique, and also to discover new values when new cliques emerge. We find that pre-trained language model embedding (e.g., BERT embedding) cannot fully capture attribute-specific information, even after MLM [4] pre-training using in-domain corpus, as seen in Figure 2. We propose an attribute-aware fine-tuning method, which takes full advantage of *value exclusiveness* of product titles as well as the weak supervision, and optimizes a multitask objective function. After the fine-tuning, embedding becomes more expressive in distinguishing values from different attributes. With well-tuned embedding, we then apply a self-ensemble-based inference method. A density-based clustering component, which can discover open-world new attributes and scrap noise from the candidates, is joint force with a classification component, which improves the model recall by finding more values of the discovered attributes. Our OA-Mine framework improves itself with iterative training, where the discovered attribute types and values help gradually shape the value space for each attribute, as the model becomes more knowledgeable.

We run extensive experiments on over 750K products from 100 product types. In addition to distantly annotated development data,

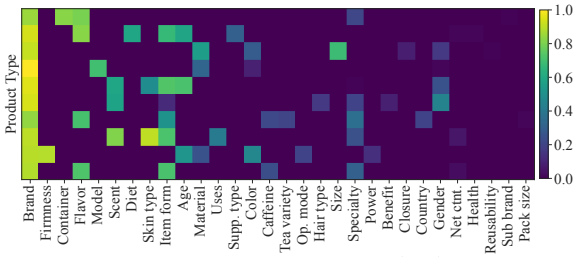


Figure 3: Attribute coverage. The cell color at (x, y) is the coverage of attribute x in product type y. 100% means all values of this attribute are found in existing structured information in product profiles.

we recruit crowd workers and domain experts to label a gold standard test set for end-to-end evaluation. Our experiments show the superiority of our framework over various state-of-the-art baseline methods. Besides, we also show the model's ability to discover unseen attributes by holding out attributes from training, and demonstrate the model's strong generalization by evaluating on product types with zero supervision.

Our main contributions can be summarized as follows:

- We propose a principled framework for open-world product attribute mining. The framework discovers both new attributes types and new attribute values.
- We propose a novel attribute-aware representation fine-tuning framework for pre-trained language models. The framework combines human given weak supervision with abundant unlabeled product text, and adapts the language model to distinguish values from different product attributes.
- We release a human annotated evaluation dataset for end-to-end evaluation of the open-world product attribute mining task.

2 PRELIMINARIES

In this section, we formally define the open-world product attribute mining task, and specify our input and output. In addition, we offer some additional insights through data analysis.

Problem Definition. Open-world attribute mining for one type of product t takes a set of products $\mathcal{P}_t$ and aims to discover a set of applicable *attributes* $\mathcal{A}_t$ for those products. Each applicable attribute $A \in \mathcal{A}_t$ is represented by a collection of *attribute values* $A = \{v_1, v_2, ...v_n\}$, which are raw text phrases. The problem is guided by weak supervision, where the user specify a few known values $S_A = \{v_1, v_2, ..., v_k\}$ for a few known attributes $\mathcal{A}_S \subset \mathcal{A}_t$. S_A is called the seed set of attribute A. The seed set size $|S_A|$ and the number of known attributes $|\mathcal{A}_S|$ are small. The predictions are attribute clusters $\bar{\mathcal{A}}_t$ and the goal is to make the predictions $\bar{\mathcal{A}}_t$ close to the ground truth $\mathcal{A}_t$.

The problem is defined for a specific type of product t, because each product type has its unique sets of applicable attributes $\mathcal{A}_t$. Product types are identified by their surface names. Putting all types of products together $\mathcal{P} = \bigcup_{t \in T} \mathcal{P}_t$, the open-world attribute mining aims to discover all the attributes $\mathcal{A}_t$ for $t \in T$.

In this paper, we use product titles from $\mathcal{P}$, and we assume the number of products $|\mathcal{P}|$ is large.

Data Analysis on Attribute Coverage. To show the necessity of open-world product attribute mining, we collected 1,943 product profiles from 10 product types on Amazon.com for data analysis. They contain raw text, as well as structured information, including

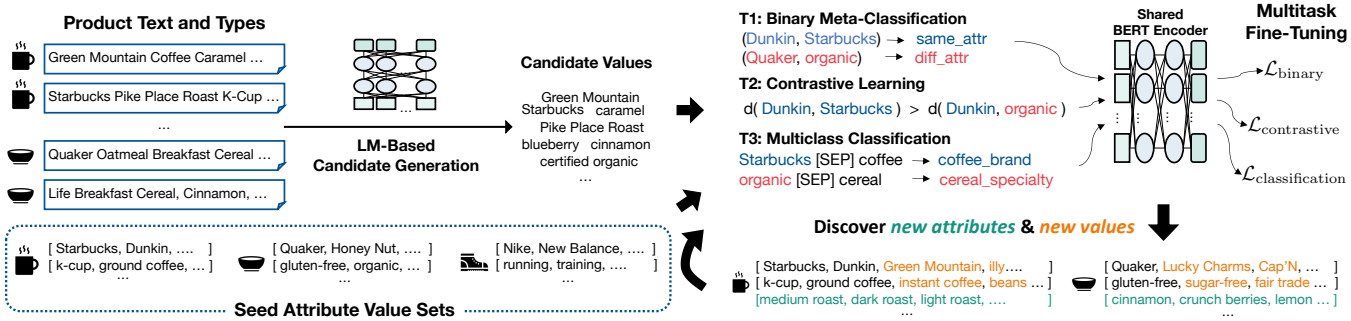


Figure 4: Overview of our proposed OA-Mine **framework. Starting from product text, we generate candidate attribute values by probing a pre-trained language model (LM). Combined with user given seed sets, we fine-tune the LM with a multitask objective. Finally, we discover new attributes and values with self-ensembled based inference. Predictions are used for training in next framework iteration.**

some attribute values already known. We hired human workers to annotate attribute values mentioned in product titles for a comprehensive analysis. The details of the data collection process can be found in Section 6.1.

We would like to know what percentage of attribute values are already presented in the structured information of those online product profiles. In total, the human annotators found 51 distinct attributes from all the products across 10 product types. Of those, only 30 attributes are found in existing structured product profiles. Figure 3 shows the percentage of values identified by humans compared to the structured attribute information in product profiles. Rows represent product types, and columns represent attributes. Comparing different rows, we can see different types of products have different applicable attributes. Judged by the color of the cells, lots of values are apparently missing. The data analysis clearly shows the need for open-world attribute mining, as we see missing attributes and values in the existing product profiles.

3 FRAMEWORK OVERVIEW

OA-Mine is a two-step framework for weakly supervised open-world product attribute mining (Figure 4), with *candidate value generation* followed by *attribute value grouping*. The first step aims to harvest phrases that are likely attribute values from raw text product titles. Data observations from Section 1 show that product titles are collections of attribute values. Our model segments the titles into phrases, essentially breaking down bags-of-attribute-values into individual value candidates. Established phrase mining tools cannot meet our needs, as pre-trained NLP pipelines [1, 7] are not adapted to the product text, while the unsupervised methods [6] typically has assumptions that do not work well on the product titles (shown in Section 6.2). We designed a language model probing-based technique, which captures the compositional relations between words for effective phrase segmentation.

Although product titles are primarily bags-of-attribute-values, there are exceptions where things that are not attribute values can be mentioned (e.g., "packaging may vary"). In our first step, we leave them as candidate values and let our second step remove them from the candidate pool. We design our first step to be recall-focused and the second step to be noise-aware, as information lost from the first step cannot be recovered later, while noise can be effectively handled.

Having obtained the attribute value candidates, we find novel attributes and put the candidates into attribute groups in our next step. Our model navigates the embedding feature space of candidate values and discovers dense areas that are likely attribute clusters. Fundamentally, the embedding of candidate values has to be attribute-aware, a requirement we have shown that pre-trained language model embedding cannot satisfy. To achieve this, we draw on the *value exclusiveness* observation from data and combine it effectively with the limited supervision we have from seed values. We optimize a multitask objective, with a binary meta-classification loss to generalize across product types, a contrastive loss to enforce embedding similarity resembles attribute-level similarity, and a classification objective to impose attribute distinctiveness.

Once we have well-trained embedding, a self-ensemble of different parts of the model first discovers new attributes and removes noise by DBSCAN [5] clustering, and then puts more values into attribute groups with classification. Finally, the model improves itself by leveraging the prediction for iterative training.

4 CANDIDATE VALUE GENERATION

The first step of our framework aims to generate attribute value candidates from product titles. As product titles are mostly bags-of-attribute-values, we formalize the task as a product title segmentation task. For example, for the product in Figure 1, we would like to segment its title into "TCL | 50-inch | 5-Series | 4K UHD |..." Our goal of this step is to include as many value candidates as possible, i.e., it is recall-focused. We allow phrases that are not attribute values to be kept, but do not want actual attribute values to be lost.

We leverage a pre-trained language model for candidate generation. As observed in prior work [6, 8, 15], pre-trained language models can capture phrases through the attention scores computed by the Transformer model.

We first fine-tune a BERT [4] language model using the Masked Language Model (MLM) [4] objective on the product text. This step is necessary because the product text differs from the general domain natural language text on which BERT is pre-trained. Then we probe the BERT model to generate a score that resembles how likely two words are from the same phrase.

Language Model Probing for Phrase Score. Given a product title as a sequence of words $W = w_1 w_2 ... w_n$, we compute the likelihood $s(w_i, w_{i+1})$ of two adjacent words belonging to the same phrase. We use the following strategy [15]:

$$s(w_i, w_{i+1}) = d\left(\text{BERT}(W/\{w_i\})_i, \text{BERT}(W/\{w_i, w_{i+1}\})_i\right) \quad (1)$$

$W/\{\cdot\}$ denotes the original text with words in the set under slash replaced with [MASK]. $\text{BERT}(\cdot)$ computes the sequence embedding by applying the BERT model, and $(\cdot)_i$ takes the i-th embedding, i.e., the token embedding of w_i. $d(\cdot,\cdot)$ is a distance function. We use cosine distance in our model.

Phrasal Segmentation. With phrase score $s(w_i, w_{i+1})$ of each adjacent words computed, we can set a threshold for phrasal segmentation. Each pair of words with $s(w_i, w_{i+1})$ above the threshold will be merged into a phrase, and those below the threshold will be segmented into different phrases. We use a small validation set to select the threshold. Empirically we found the model and a single threshold generalize very well across different product types.

One drawback of relying entirely on the language model is that it does not guarantee the completeness of the phrases. For example, "QLED TV" may be combined into one phrase in one product, but separated into two words in another product. This is because the BERT representation is context-dependent, and cannot leverage global frequency signals from the whole corpus. We, therefore, apply frequent pattern mining to combine words that frequently co-occur in the corpus, which further improves the completeness of the attribute value candidates.

5 ATTRIBUTE VALUE GROUPING

Once we have generated candidate values from product titles, we are ready to group them into attribute clusters. Given the open-world challenge in our problem setting, standard token classification approaches cannot suffice. Fully unsupervised clustering methods also fall short, as they rely on the pre-trained representation of phrases, which is not attribute-aware (Figure 2).

We identify two main requirements for discovering new attributes and values: (1) The candidate value representation must be attribute-aware, placing values from the same attribute close to each other, and those from different attributes far apart; (2) The model must generalize to unseen attributes and to various product types. Bearing these two requirements in mind, we first design a multitask fine-tuning approach that focuses on attribute awareness and generalization. Then we apply a self-ensemble inference to discover new attributes and values.

5.1 Attribute-Aware Representation Fine-Tuning

Given the candidate attribute values, we use a shared encoder to obtain their embedding, and then we apply three different objective functions to fine-tune the model.

Attribute Value Encoder. Given a candidate attribute value v and its textual context in the product title W, we use a BERT [4] encoder to generate the token representation of the whole sequence. Next, we apply an average pooling on the token representations of the attribute value to obtain its representation $\boldsymbol{h}_v$. Specifically,

$$\hat{\boldsymbol{H}} = \left[\hat{\boldsymbol{h}_1}\hat{\boldsymbol{h}_2}...\hat{\boldsymbol{h}_n}\right] = \text{BERT}(W) \quad (2)$$

$$\boldsymbol{h}_v = \sum\nolimits_w \left(\hat{\boldsymbol{h}_w} \cdot 1\left[w \in v\right]\right) / \sum\nolimits_w 1\left[w \in v\right] \quad (3)$$

The characteristic function $1[\cdot]$ equals to 1 for words in the candidate value, and 0 for the words in the context, thus masking out the hidden representation from the context.

Recall that our input seed sets are collections of attribute values without context. When encoding the values from the seed sets, we string match them to their occurrences in the products, and use those occurrences for their contextualized representation.

Binary Meta-Classification. The ultimate goal of fine-tuning is to embed values from the same attribute close to each other and embed values from different attributes far apart. We build a binary classification model f to directly optimize for this goal: given a pair of values v_1, v_2, let $f(v_1, v_2)$ be 1 if v_1, v_2 are from the same attribute, and -1 otherwise.

We use a BERT bi-encoder [10] with cosine similarity objective as the model f.

$$f(u, v) = \text{cosine\_sim}\left(\boldsymbol{h}_u, \boldsymbol{h}_v\right) \quad (4)$$

$$\mathcal{L}_{\text{binary}} = \sum_{(u,v)\in P} \left\|1 - f(u,v)\right\|^2 + \sum_{(u,v)\in N} \left\|-1 - f(u,v)\right\|^2 \quad (5)$$

P and N are positive and negative example sets respectively. We choose BERT bi-encoder as opposed to the standard BERT cross-encoder [4] because the bi-encoder is more efficient at inference time, as we will see in Section 5.2.

Note that we would like the model to capture the attribute distinctiveness, not just on one attribute of one product type. Rather, we would like to build a *meta-classifier* which generalizes to all attributes and all product types. We carefully sample the positive training pairs P and the negative training pairs N from both the seed sets and the unlabeled corpus. We generate positive pairs from values in the same attribute seed set, and negative pairs from different seed sets. In addition, based on the *value exclusiveness* data observation, we generate strong negative training data by first sampling a product title from the corpus, and then sampling a pair of values from the same title. Our method generates higher quality pairs compared to random negative sampling.

Contrastive Learning. In addition to directly optimizing the binary meta-classification loss, we find it beneficial to bind together the positive and the negative training pairs with a contrastive learning loss [2, 3, 11, 12]. The contrastive learning objective takes a triplet of $(v_{\text{anc}}, v_{\text{pos}}, v_{\text{neg}})$, and enforce the embedding distance from the anchor v_{anc} to the positive example v_{pos} to be smaller than that from the anchor v_{anc} to the negative example v_{neg}.

Contrastive learning has different instantiations in the choice of the loss function. Recent studies usually explicitly generate positive pairs with data augmentation but implicitly generate negative pairs [3]. Since we can explicitly generate strong negative pairs, we resort to a triplet loss function [11]:

$$\mathcal{L}_{\text{contrastive}} = \sum_{(v_a, v_p, v_n)} \max\left(\left\|f(v_a, v_p)\right\|^2 - \left\|f(v_a, v_n)\right\|^2 + \alpha, 0\right) \quad (6)$$

α is a constant margin value. The triplets are generated by binding our positive pairs P and negative pairs N.

Multi-class Classification. The binary and the contrastive loss functions would shape the hidden feature space to be attribute-aware, as they manipulate the embedding space, so values that

fall into the same attribute are pulled closer while values that belong to different types are pushed away from each other. However, neither of those loss functions enforce the class distinctiveness of attribute values. Adding class distinctiveness by a multi-class classification loss would push the embedding from different attributes further apart. It offers additional benefits at inference time, shown in Section 5.2, where it would improve the recall of the model.

Let us assume we have already discovered the complete set of attributes. Now, we can build a multi-class classifier to put each value into an attribute.

The challenge is scaling up to multiple product types, as each product type has a distinct set of applicable attributes. For example, the flavor of coffee can be very different from that of cereal. Most prior works do not distinguish product types and are suboptimal.

We want to avoid building a separate classifier for each product type, while maintaining the product type awareness of the model. As such, we feed both the attribute value with context and the product type surface name to the multi-class classification model, delimited by the [SEP] special token. The output label space is the union of applicable attributes for all product types, e.g., "coffee_flavor", "tea_brand". We use the cross-entropy loss function for classification.

$$\hat{\boldsymbol{y}} = \text{Softmax}(\text{Linear}(\text{BERT}(W[\text{SEP}]t))) \tag{7}$$

$$\mathcal{L}_{\text{classification}} = \text{CrossEntropy}(\hat{\boldsymbol{y}}, \boldsymbol{y}) \tag{8}$$

W is the attribute value v in its context, t is the product type surface name, and $\boldsymbol{y}$ is the one-hot encoding of the class label.

Note that we only use the attributes with seed values as our classes at the beginning of the training. We expand to more attributes through iterative training, where the predictions from the last iteration will reveal the complete attribute landscape.

5.2 Self-Ensemble Based Inference

Once the model has been fine-tuned, we are ready to run an open-world inference to discover new attributes and values, given the attribute-aware embedding for each value.

Attribute Discovery with DBSCAN. Thanks to the bi-encoder training with binary and contrastive objectives, the similarity of value embedding now reflects the likelihood of two values coming from the same attribute.

We opt for DBSCAN [5] to discover new attributes as it has several desirable properties. First, its local density requirement consolidates the attribute-level similarity computed by pairwise distance. Second, it discovers new attributes by finding dense cliques in the embedding space. Third, it excludes values that it deems noise if the embedding of a value is far from any dense clusters.

Improving Recall with Classifier Inference. We run the multi-class classifier component after the fine-tuning. Each occurrence of a candidate attribute value is classified into an attribute cluster previously discovered by the DBSCAN model. The corpus level prediction of an attribute value is generated by majority voting of each occurrence of that value.

Ensembling both Modules. Empirically, we have observed that the DBSCAN inference is very precision focused, leaving a large number of values out in a "noise" cluster. Whereas the classifier inference is recall-driven, expanding the model's coverage effectively. We, therefore, combine the attribute discovery ability of the DBSCAN inference with the classifier. Once DBSCAN discovered attribute clusters with high precision, we run the classifier on the noise cluster to include more values that were mistakenly excluded from the prediction.

Algorithm 1: Attribute Value Grouping of OA-Mine

Input: Seed sets S for all product types T, candidate attribute values C, pre-trained model Θ.

repeat
  // Attribute-Aware Representation Fine-Tuning
  Initialize empty sets $D_{\text{binary}}, D_{\text{contrastive}}, D_{\text{classification}}$;
  Initialize empty prediction set P ;
  for $t \in T$ **do**
    $D_t \leftarrow \text{generate\_data}(S_t \cup P_t, C_t)$; // Sec 5.1
    Expand $D_{\text{binary}}, D_{\text{contrastive}}, D_{\text{classification}}$ with D_t ;
  $D \leftarrow D_{\text{binary}} \cup D_{\text{contrastive}} \cup D_{\text{classification}}$;
  Shuffle and batch D ;
  for $b \in D$ **do**
    Compute loss $L(\Theta)$
      $L(\Theta)$ = Eq. 5 for binary meta-classification
      $L(\Theta)$ = Eq. 6 for contrastive learning
      $L(\Theta)$ = Eq. 8 for classification
    Update model $\Theta \leftarrow \Theta - \epsilon \nabla(\Theta)$;
  // Self-Ensemble Inference
  $P \leftarrow \{\}$; // Empty previous predictions
  for $t \in T$ **do**
    $P_t \leftarrow \text{run\_inference}(C_t, \Theta)$; // Sec 5.2
    $P \leftarrow P \cup P_t$;
until *max_iter*;

5.3 Iterative Training

After one iteration of the framework, we have discovered new attributes and values. They are clusters of attribute values for different product types, and have the same format as our input seed sets. Therefore, we leverage the confident predictions as training data in the next iteration to boost the model performance further.

There are two key benefits of iterative training. First, it augments the existing labeled seed sets, expanding the minimal supervision. Second, it offers a more complete landscape of attributes in the dataset. In our open-world setting, we do not have comprehensive supervision for all attributes in the beginning. The model predictions will alleviate this issue, especially for the classification objective in our framework, which operates on a set of known attributes. The complete training algorithm is shown in Algorithm 1.

6 EXPERIMENTS

6.1 Datasets

Training Data. We collect publicly available product profile data from Amazon.com for 100 different product types. Besides product titles, which will be used as textual descriptions of products, we also collect structured attributes found in these profiles (see Figure 1 as an example). For each product type, we find its applicable attributes, e.g., "brand", "roast type" for coffee and "gender", "sports type"

Table 1: Evaluation on Attribute Value Candidate Generation. Methods are divided into pre-trained, distantly supervised, and unsupervised, from top to bottom.

Methods	Entity-Prec.	Entity-Rec.	Entity-F1	Corpus-Rec.
spaCy [7]	31.19	19.15	23.73	50.02
FlairNLP [1]	34.81	24.33	28.64	52.17
AutoPhrase [13]	26.58	29.67	28.04	32.39
UCPhrase [6]	35.01	19.66	25.18	37.50
OA-Mine	**42.53**	**53.29**	**47.30**	**64.10**

for shoes. For each applicable attribute, we examine the values mentioned in the product profiles that we collected, and take the most frequent values as the seed set for that attribute. We keep up to 5 seed values for each attribute.

Test Label Set. For end-to-end evaluation purposes, we manually selected products from 10 product types, sampled up to 200 products for each product type (1,943 products in total), and hired human workers for labeling. We created a labeling tool that allows human workers to highlight attribute values mentioned in product titles and label their type. We defined a few applicable attributes we already know for each product type and added an option for human workers to label new attributes not seen in the predefined set.

First, each product is labeled by 5 crowd workers on Amazon Mechanical Turk. Then, the labeled profiles are reviewed by a domain expert for consolidation. After the values for known attributes are consolidated, we review the dataset for another pass and assign proper attribute types to those previously labeled as "new attribute". On average, there are 11.5 attributes per product type marked by humans, and each attribute has 48.1 unique values.

The labels are gathered from human annotations, and are clusters of attribute values for the 10 selected product types. For example, labeled attribute values for coffee products are {Brand: [Starbucks, Dunkin, ...], Roast Type: [medium roast, dark roast, ...], ...}.

Development Label Set. In addition to the human-annotated test set on selected 10 attribute types, we create a more extensive development set covering all 100 product types. The labels are generated from the existing attribute information in the collected product profiles, using the same procedure described in the "Training Data" section for seed sets curation. Aligned with training seed sets, we only keep values in the development when they are discovered by our candidate generation step. On average, each attribute has 29.7 unique values.

With a limited budget, the development set would allow us to evaluate the performance of different models on 100 product types. Note that for most prior works on product attribute mining [14, 18, 19], the authors use the same method for gold-standard evaluation. While in this paper, the development set serves the purpose of relative performance comparison.

Reproducibility. Our code and data can be found at https://www.github.com/xinyangz/OAMine.[1]

[1]Note that although there were a few existing works [14, 18, 19, 22] on product attribute mining, only one of them [18] provided a subset of their experiment data, and the dataset was labeled with distant supervision, thus do not serve our needs.

6.2 Candidate Generation Performance

We compare the attribute value candidate generation module of our framework to the following baseline methods on the human-annotated test set:

- **spaCy [7]** is an industrial library with a pre-trained statistical noun phrase chunking model.
- **FlairNLP [1]** is a neural NLP library. We use its phrase chunking model [2] that is based on a neural sequence tagger.
- **AutoPhrase [13]** is a distantly-supervised phrase mining tool. It leverages a high quality phrase dictionary from Wikipedia and trains a statistical classifier based on POS-tags and statistical features of text spans.
- **UCPhrase [6]** is an unsupervised phrase mining framework. It first finds "core phrases" by mining frequent closed sequential patterns. It then uses the core phrases to train a CNN-based phrase classifier which leverage features from attention maps extracted from pre-trained language models.

Evaluation Metrics. We report micro-averaged entity level precision, recall, F1, which are computed per instance. In addition, we also report corpus-level recall where the predictions on all products are compared to the labeled set.

Results. As shown in Table 1, our model consistently outperform the baseline methods by a large margin. spaCy and FlairNLP are sub-optimal on our dataset because they are pre-trained on general domain corpora, with very different language characteristics than product titles. AutoPhrase performs poorly as it lacks in domain distant training dictionary. UCPhrase struggles on our dataset because (1) it can only capture multi-word phrases while attribute values can be single tokens, (2) the core phrases generated by pattern mining are low quality. Unlike long documents on which the authors designed their model, product text is noisy and repetitive, undermining the pattern mining method.

6.3 End-to-End Evaluation

The end-to-end evaluation runs each model on the complete training set, and compares their performance on the dev and test sets.

Note that there is no existing work that solves the open-world attribute mining problem to our knowledge. We include several sequence tagging-based models, which are the de-facto models for closed-world product attribute mining. We also combine the candidate generation results from our model with a few open-world classification and clustering models for evaluation.

- **BiLSTM-Tag** is a sequence tagging model with a bi-directional LSTM encoder. The model trains a stand alone tagging model for each known attribute in the seed set.
- **OpenTag [18]** improves the BiLSTM-Tag model by adding an attention layer. The original model adopts a CRF layer for prediction. However, we found that the CRF layer consistently underperforms a linear layer in our weakly supervised setting. Therefore, we use a linear prediction layer instead.
- **SU-OpenTag[18]**[3] is an end-to-end product attribute mining framework. It encodes product text and attribute surface name

[2]https://huggingface.co/flair/chunk-english

[3]A more recent work [19] improves the model by innovating on the CRF layer. However, CRF does not perform well in our weakly supervised setting. Therefore, we exclude [19] from comparison.

Table 2: End-to-end evaluation on development and test data. Results are average of 3 runs. Bold faced numbers indicate statistically significant results from t-test with 99% confidence.

		Dev Set (100 product types)				Test Set (10 product types)			
Method Type	**Method**	ARI	Jaccard	NMI	Recall	ARI	Jaccard	NMI	Recall
Sequence tagging (closed-world)	BiLSTM-Tag	0.299	0.354	0.422	0.565	0.175	0.219	0.374	0.162
	OpenTag [22]	0.244	0.324	0.334	0.593	0.160	0.247	0.357	0.165
	SU-OpenTag [18]	0.637	0.598	0.607	0.525	0.411	0.340	0.542	0.162
Unsupervised clustering	BERT+AG-Clus	0.249	0.446	0.585	0.742	0.386	0.308	0.504	**0.430**
	BERT+DBSCAN	0.133	0.146	0.507	0.131	0.385	0.412	0.575	0.186
Weakly sup. clustering	DeepAlign+ [21]	0.175	0.226	0.336	0.729	0.257	0.208	0.426	0.389
	OA-Mine (no multitask)	0.671	0.634	0.610	0.458	0.601	0.518	0.733	0.225
	OA-Mine	**0.704**	**0.689**	**0.629**	0.747	**0.712**	**0.650**	**0.781**	0.275

separately with BERT [4], and combines them with LSTM and attention layers. Unlike OpenTag, the model trains a single tagger for all the known attributes. We substitute the CRF layer in the original paper with a linear layer as it performs better.

- **BERT+AG-Clus** leverage BERT embedding for agglomerative hierarchical clustering. We use the attribute value candidates generated by our framework as the input to the clustering method.
- **BERT+DBSCAN** leverage BERT embedding for DBSCAN density-based clustering. We use the attribute value candidates generated by our framework as the input to the clustering method.
- **DeepAlign+ [21]** is an open-world intent classification framework with clustering and self training. We adapt it for open-world token classification.
- **OA-Mine (no multitask)** is an ablation of our model. It uses binary meta-classification as its only fine-tuning objective function. Meanwhile, it relies on DBSCAN only for inference, as the classifier component is turned off during training.

Experiment Setting. For sequence tagging-based models that use distant supervision, following [18], we string match our weak supervision sets to the product text to obtain the annotated corpus. The clustering models and the DeepAlign+ model can only handle open-world classification, but not value extraction. As such, we use the candidate values generated by our model as the input to those models. After model training, we run all the methods on the whole corpus for inference, and calculate the evaluation metrics on the product types from the development and test sets. Therefore, our evaluation is transductive. All models that utilize BERT use the same pre-trained model with in-domain MLM fine-tuning.

The main parameters of our model are candidate generation threshold, DBSCAN parameters, and number of iterations. We set the candidate generation threshold as described in Section 4. The DBSCAN parameters are selected after our attribute-aware fine-tuning. When generating fine-tuning data, a small validation set was held out. The model's distance threshold for binary meta-classification performs well for DBSCAN, as they all rely on pairwise cosine distances. We run iterative training until stable – up to 5 iterations in practice.

Evaluation Metrics. We compare the predicted attribute clusters with the ground truth clusters derived from human annotations. We use standard clustering evaluation metrics, namely **adjusted Rand index (ARI)**, **Jaccard**, **normalized mutual information (NMI)** for clustering quality evaluation, and **recall** for clustering coverage evaluation. We refer readers to [20, Chapter 17] for details of these metrics. All metrics except for recall are *computed on labeled attribute values* and micro-averaged across attributes and product types. The range of ARI is -1 to 1 and all the other metrics are 0 to 1, larger is better.

Results. As shown in Table 2, our model achieves the best performance across the board with the exception of recall on the test set. Recall of BERT+AG-Clus and DeepAlign+ are superior because they cannot remove noise from the candidates, and include everything in their predictions. We can clearly see the downsides of that, as the rest of the metrics indicating the quality of the predicted clusters are significantly lower than our model.

Sequence tagging-based models perform poorly for two main reasons. First, they require more training data to work, and second, they cannot discover new attributes. SU-OpenTag works better than the other two as it leverages BERT pre-trained embeddings.

Comparing our model to the BERT+DBSCAN model, we clearly see the necessity of attribute-aware fine-tuning. In fact, without fine-tuning, DBSCAN is extremely sensitive to parameters and performs poorly despite our tremendous effort in tuning it to work.

The DeepAlign+ method is surprisingly worse than the AG-Clus unsupervised clustering in many metrics. This is because the initialization that it based on is poor on our data, and the adaptive clustering component, i.e., self-training, fails to work on the poor initialization. Moreover, the BERT representation used by DeepAlign+ is not attribute-aware.

Comparing our model to the ablation without multitask objective and self-ensemble inference, we can see performance increase across the board, with both better quality and better recall in our main model. This demonstrates that multitask training boosts the attribute-awareness of the model, leading to better quality, while the self-ensemble generated better coverage.

6.4 Evaluation on New Attributes

To test model's ability on discovering new attributes, we run an experiment on the development set, as we have aligned attribute types for training seed sets and evaluation label sets. For all the available seed sets in all product types, we hold out 20% attribute types, and use the rest for training. After training and inference, we test the model's performance on the 20% held-out attributes. We report the performance from 5-fold cross-validation on held

Table 3: Performance on discovering new attributes. Experiment conducted with 5-fold cross-validation, where each fold holds out 20% attributes from training.

Methods	ARI	Jaccard	NMI	Recall
BERT+AG-Clus	0.215	0.372	0.308	0.832
BERT+DBSCAN	0.199	0.431	0.129	0.370
DeepAlign+	0.192	0.329	0.303	0.831
OA-Mine	0.599	0.743	0.489	0.688

Table 4: Comparing model predictions on unseen attributes during cross-validation. Red is error.

Attribute	Method	Predicted Cluster
Coffee Brand	BERT+AG-Clus	green mountain, folgers, coffee fool, maxwell house, coffee roasters, nescafe, eight o clock, ...
	DeepAlign+	gourmet, keurig brewers, starbucks, green mountain coffee, donut, dunkin donuts, ...
	OA-Mine	starbucks, green mountain, folgers, coffee fool, maxwell house, nescafe, san marco coffee, ...
Laundry Detergent Form	BERT+AG-Clus	powder, bottle, pacs, original, 2, pods, 32 loads, ...
	DeepAlign+	liquid, laundry, wash, pack, stain, natural, ...
	OA-Mine	liquid, powder, bottle, spray, carton, pods, soap, ...

out attributes. We exclude the sequence labeling-based methods, as they are closed-world and cannot discover new attributes.

The results are shown in Table 3. We observe that our method generates significantly higher quality clusters measured by ARI, Jaccard, and NMI scores. Note that BERT+AG-Clus and DeepAlign+ cannot exclude noise from the predictions, resulting in lower quality, albeit high recall.

Table 4 shows a case study of models' ability in discovering unseen attributes. The selected attributes are held out during training. The cluster that most resembles the target attribute is selected for each method. Values are sorted by frequency. We omit BERT+DBSCAN as it consistently under-performs our model. As we can see, our model generates the most accurate and semantically coherent clusters. BERT+AG-Clus and DeepAlign+ cannot remove noise from predictions, resulting in lower quality. DeepAlign+ also suffers from semantic drift from adaptive clustering.

6.5 Evaluation on New Product Types

We test the ability of our model to generalize to product types with no weak supervision seed sets. In this experiment, we exclude the seed sets for product types in the test set, and use the remaining seed sets from the rest of the 90 product types for training. The model has full access to the raw product text. After training, we run inference on the products not seen during training, and compute the performance on the test set. None of the existing attribute mining or open-world classification baselines can handle this challenging experiment setting. We compare to the unsupervised clustering methods and also to our main experiment result.

The results are seen in Table 5. Compared to the unsupervised clustering, our model generates significantly higher quality predictions, as measured by ARI, Jaccard, and NMI. Without access to supervision for these product types, our model performs understandably slightly worse than the full access model reported in the main experiment Table 2. The difference, however, is not large, showing the encouraging generalization of our model.

Table 5: Performance on new product types. Models tested on product types not seen during training.

Methods	ARI	Jaccard	NMI	Recall
BERT+AG-Clus	0.386	0.308	0.504	0.430
BERT+DBSCAN	0.385	0.412	0.575	0.186
OA-Mine	0.658	0.609	0.702	0.231

7 RELATED WORK

Product Attribute Mining. Most prior work on product attribute mining formalize the problem as a sequence labeling task. Inspired by named entity recognition models, earlier work leverage statistical models [9] for extraction. With the advancement of deep learning, the most prominent systems designed in recent years adopt BiLSTM-CRF [19, 22] or BERT-BiLSTM-CRF [18] architectures for attribute value extraction. Supervised method combined with active learning [22] was explored in OpenTag [22], while follow up works typically settle on distant supervision [14, 18, 19]. Besides sequence labeling-based methods, AVEQA [14] explored question answering-based models for attribute value extraction.

The major different between our work and the existing works is that we approach the problem with open-world assumptions on both attributes and values, while prior works assume the attributes of interest are given as input. For supervision, we adopt weak supervision, which is a small amount of high quality data, instead of distant supervision, where a large amount of noisy data is used.

Open-World Text Classification. Although few work studies entity extraction in the open-world setting, a few prior works have explored open-world text classification with different application scenarios. Xu et al. [17] proposed a model for open-world product classification, where they learn a meta-classifier to either match a new example to a seen class or to assign it to a new class. Zhang et al. [21] studies open-intent discovery, which is essentially open-world text classification, with an adaptive clustering and self-training-based approach. Most of these work relies on pre-trained language representation and adaptive clustering [16].

Our problem setting is more challenging than open-world text classification, as we have to extract attribute values from raw text. In terms of methodology, our proposed model can take advantage of unique characteristics of the product text, and can learn better representation. As seen in our experiments, the adaptive clustering-based methods do not work well out-of-the-box, as the product text is noisy and leads to poor convergence of adaptive clustering.

8 CONCLUSION AND FUTURE WORK

In this paper, we proposed a principled framework for open-world product attribute mining. The framework includes a novel candidate generation method, an attribute-aware representation fine-tuning model, and achieves open-world inference through self-ensemble. Our framework offers strong performance and generalize to unseen attributes and product types. In the future, we will extend our model to handle additional text fields besides product titles, such as using product introductions. Improving the classifier to make open-world inference is also a promising direction.

REFERENCES

[1] A. Akbik, T. Bergmann, D. Blythe, K. Rasul, S. Schweter, and R. Vollgraf. Flair: An easy-to-use framework for state-of-the-art nlp. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, 2019.

[2] G. Chechik, V. Sharma, U. Shalit, and S. Bengio. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, 11(36):1109–1135, 2010.

[3] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, 13–18 Jul 2020.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. 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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[5] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[6] X. Gu, Z. Wang, Z. Bi, Y. Meng, L. Liu, J. Han, and J. Shang. Ucphrase: Unsupervised context-aware quality phrase tagging. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 478–486, New York, NY, USA, 2021. Association for Computing Machinery.

[7] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd. spacy: Industrial-strength natural language processing in python. 2020.

[8] T. Kim, J. Choi, D. Edmiston, and S. goo Lee. Are pre-trained language models aware of phrases? simple but strong baselines for grammar induction. In *International Conference on Learning Representations*, 2020.

[9] D. Putthividhya and J. Hu. Bootstrapped named entity recognition for product attribute extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1557–1567, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics.

[10] N. Reimers and I. Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, Nov. 2019. Association for Computational Linguistics.

[11] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, 2015.

[12] M. Schultz and T. Joachims. Learning a distance metric from relative comparisons. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2004.

[13] J. Shang, J. Liu, M. Jiang, X. Ren, C. R. Voss, and J. Han. Automated phrase mining from massive text corpora. *IEEE Transactions on Knowledge and Data Engineering*, 30(10):1825–1837, 2018.

[14] Q. Wang, L. Yang, B. Kanagal, S. Sanghai, D. Sivakumar, B. Shu, Z. Yu, and J. Elsas. Learning to extract attribute value from product via question answering: A multi-task approach. KDD '20, page 47–55, New York, NY, USA, 2020. Association for Computing Machinery.

[15] Z. Wu, Y. Chen, B. Kao, and Q. Liu. Perturbed masking: Parameter-free probing for analyzing and interpreting BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4166–4176, Online, July 2020. Association for Computational Linguistics.

[16] J. Xie, R. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 478–487, New York, New York, USA, 20–22 Jun 2016. PMLR.

[17] H. Xu, B. Liu, L. Shu, and P. Yu. Open-world learning and application to product classification. In *The World Wide Web Conference*, WWW '19, page 3413–3419, New York, NY, USA, 2019. Association for Computing Machinery.

[18] H. Xu, W. Wang, X. Mao, X. Jiang, and M. Lan. 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, Florence, Italy, July 2019. Association for Computational Linguistics.

[19] J. Yan, N. Zalmout, Y. Liang, C. Grant, X. Ren, and X. L. Dong. AdaTag: Multi-attribute value extraction from product profiles with adaptive decoding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4694–4705, Online, Aug. 2021. Association for Computational Linguistics.

[20] M. J. Zaki and W. Meira. *Data Mining and Machine Learning: Fundamental Concepts and Algorithms*. Cambridge University Press, 2 edition, 2020.

[21] H. Zhang, H. Xu, T.-E. Lin, and R. Lyu. Discovering new intents with deep aligned clustering. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16):14365–14373, May 2021.

[22] G. Zheng, S. Mukherjee, X. L. Dong, and F. Li. Opentag: Open attribute value extraction from product profiles. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 1049–1058, New York, NY, USA, 2018. Association for Computing Machinery.