PAVE: Lazy-MDP based Ensemble to Improve Recall of Product Attribute Extraction Models

Kushal Kumar
kushlku@amazon.com
Amazon.com
Bengaluru, Karnataka, India

Anoop Saladi
saladias@amazon.com
Amazon.com
Bengaluru, Karnataka, India

ABSTRACT

E-commerce stores face the challenge of missing and inconsistent attribute values in the product detail pages and have to impute them on behalf of their vendors. Traditional approaches formulate the problem of attribute extraction *(AE)* from product profiles as natural language tasks such as information extraction or text classification. Such models typically operate at high precision but may yield low recall especially on attributes with an open vocabulary due to 1) missing or incorrect information in product profiles, 2) generalization errors due to lack of contextual understanding, and 3) confidence thresholding to operate at high precision. In this work, we present **PAVE**: **P**roduct **A**ttribute **V**alue **E**nsemble, a novel reinforcement learning model that uses *Lazy-MDP* formalism to solve for low recall by aggregating information from a sequence of product neighbors. We train a policy network using *Proximal Policy Optimization* that learns to choose the correct value from the sequence. We observe consistent improvement in recall across all open attributes compared to traditional *AE* models with an average lift of 10.3% with no drop in precision. Our method surpasses simple aggregation methods like nearest neighbor, majority vote and binary classifier ensembles and even outperforms *AE* models for closed attributes. Our approach is scalable, robust to noisy product neighbors and generalizes well on unseen attributes.

CCS CONCEPTS

• **Theory of computation** → **Reinforcement learning**; • **Information systems** → **Information extraction**; • **Computing methodologies** → *Ensemble methods*; *Policy iteration.*

KEYWORDS

Lazy-MDP; reinforcement learning; proximal policy optimization; product attribute extraction; e-commerce; confidence thresholding

ACM Reference Format:
Kushal Kumar and Anoop Saladi. 2022. PAVE: Lazy-MDP based Ensemble to Improve Recall of Product Attribute Extraction Models. In *Proceedings of 31st ACM International Conference on Information and Knowledge Management (CIKM '22).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

CIKM '22, October 17–21, 2022, Atlanta, Georgia, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

E-commerce stores often have millions of products in their catalog. A large share of these products are listed by third-party vendors. Due to the scale of these third-party listings, manual inspection on every product detail provided by the vendor is infeasible. These details are thus shown directly on the product detail pages. Poor linguistic proficiency of vendors, lack of holistic understanding of global customers and disparity between vendor and e-commerce interpretations of some fields in the listing template lead to errors in the listed information [20]. In particular, this results in low quality of product attribute information that leads to customer dissatisfaction. However, this problem can be solved when the information is present in free-text fields like title, bullet or even images from where it can be extracted (see a hypothetical example in Figure 1). Developing models for *attribute extraction (AE)* are critical to improve catalog quality at scale. These models can be used to backfill millions of products and even correct erroneous products by inspecting catalog values that are inconsistent with model predictions. Improved catalog quality increases customer satisfaction while elevating other systems such as search systems for queries with attribute value mentions, systems like product recommendation that consume these attributes, product widgets for easy discoverability of attributes, among others.

Product attribute extraction is a long studied problem in literature covering various aspects like emerging entities, scalability and accuracy. Most of the work on AE pose the problem as natural language processing *(NLP)* tasks using the product's profiles, such as text classification [13, 20, 39], semantic matching [26, 27, 31], information extraction *(IE)* using Named Entity Recognition *(NER)*

Title	Garnier Skin Naturals, Charcoal, Face Serum Sheet Mask (Black), 28g & Garnier Skin Naturals
Bullet	Garnier Skin Naturals, Charcoal, Face Serum Sheet Mask (Black), 28gGarnier introduces a new generation of face masks for women that infuses skin with 1 week of serum with 1 mask. Garnier black serum mask is a breakthrough black tissue mask technology that offers double purifying and hydrating efficacy. Use Garnier black serum mask if you have dull skin with clogged and enlarged pores.
Scent	Blank

Figure 1: An example face sheet mask product where the vendor did not provide the scent attribute value during product listing. However, ground truth for scent (in green) is present in the product's profile and can be extracted.

like sequence labeling [21, 29, 44] and question answering [2, 35]. These are competent approaches that are easy to actuate as they can operate at a high precision. However, they may suffer from low recall when deployed, more so for open attributes that have large and often evolving vocabulary, due to following reasons.

Firstly, the product data can be noisy and uninformative for an attribute that gets exacerbated on tail products. Traditional NLP formulations fail to address this problem from a recall perspective. Secondly, open attribute models may select out-of-context answers for both seen and unseen attribute values leading to recall loss. *Flavor* attribute (bolded) generally occurs as a prefix to the product name (underlined), for example - *"Bengal bay spiced **orange & basil** tonic water with organic ingredients"*. Open attribute models exploit such semi-structures for generalizability but are error prone on products that don't follow them, for example *"Creamix pan cake premix **rose** 4000g"*. Similarly, these models also tend to select common training values that are out-of-context in test data. Consider a white toothbrush with description *"The CleanMaximiser technology **turns green bristles yellow** and indicates the time to change for best cleaning"*, a *color* attribute model may select "green" or "yellow" as the product color from this description. Thirdly, confidence thresholding to achieve high precision leads to low recall by dropping correct low confident predictions. Moreover, AE models typically do not consume all product profiles (such as manufacturer notes, product videos, customer reviews, etc.) to ensure low model complexity and truncate noise as these profiles are often lengthy and vague. Human annotators, however, use them when tagging ground truth labels as they can effectively filter out noise. This may also contribute to low recall of open AE models. Even closed attribute models can operate at low recall for minority classes [25] and due to shifts in class distributions on test products [2].

Low recall of the AE models reduces its coverage when backfilling omitting a large number of products with missing or incorrect attribute values. Hence, it becomes an important problem to solve and usually the adopted solution is iterative to tune existing models or train afresh using training data from products omitted by previous models and if it fails, enhance the model architecture to learn better representations from more data ([9, 15]). These solutions evolve slowly and depend on the specific attribute requiring manual inputs (for example, annotation for active learning, choosing modality or multi-tasks for an attribute). Moreover, they seldom build on top of existing models and still suffer from low recall due to data quality issues. We intend to solve these challenges by devising a simple yet effective algorithm that is agnostic of attribute type, builds on top of existing AE models and easy to scale to hundreds of attributes. To begin, we train baseline AE models by fine-tuning pre-trained BERT model [10] using publicly available products in a locale within *amazon.com* (see Section 5.2.1).

We analyze recall misses on test sets that were created through manual audits (see Section 5.1). We find that 37% test products did not even contain ground truth in the input text. However, when we looked at product neighbors (see Section 4.1.1) of these test products, we found that the ground truth was present as a catalog value within top 20 neighbors for 77% of them (see Figure 2). Hence, the recall can be significantly improved by accurately ensembling values from neighbors. However, even the nearest neighbor value was only

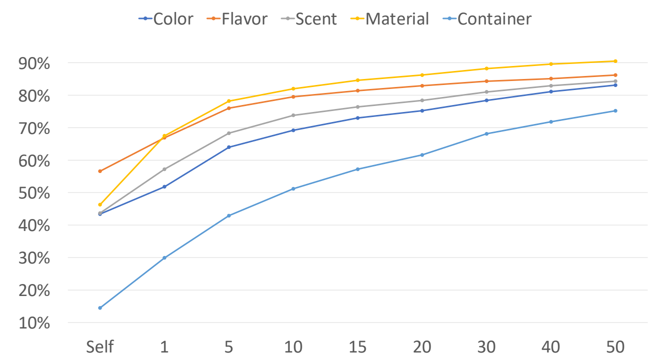

Figure 2: Ground truth coverage in neighbor catalog values where product neighbors are ranked from 1-50 basis their cosine similarity of the product embeddings. For *Self*, we consider both catalog value and baseline AE model prediction.

54% accurate for the given product, thus a sub-optimal ensemble would have a low precision. To solve this challenge, we propose **PAVE**: **P**roduct **A**ttribute **V**alue **E**nsemble, a novel ensemble model using reinforcement learning *(RL)*, taking inspiration from Liu et. al. work [18] and Lazy-MDP formalism [14]. Our RL agent scans lazily through each product neighbor to decide whether the neighbor attribute value is relevant or not and emits the best value at the end of the episode. Our approach not only handles noise to preserve precision, but also allows dynamic neighbor length (see Section 3.1). To the best of our knowledge, we are first to apply RL in the domain of product attribute extraction and propose several novel techniques to handle the aforementioned challenges with traditional solutions. Concretely, we make the following contributions:

- We propose the attribute value ensembling task as a novel RL based Lazy-MDP that improves recall of traditional AE models without dropping precision.
- We introduce novel intermediate rewards for training, and emit confidence scores to control precision-recall tradeoff.
- We evaluate our models on real world e-commerce datasets and compare against strong baselines like BERT based AE models and multiple ensemble methods.
- Our proposed approach is scalable, easy to adopt in an e-commerce setting, robust to noisy product neighbors and generalizes well on unseen attributes.

2 RELATED WORK

Product attribute extraction is a widely studied domain and most of the recent work can be classified into following categories:

(1) **Multi-task learning:** Clark et. al. [6] propose cross-view training using multiple auxiliary predictors on unlabeled data, Karamanolakis et. al. [15] add taxonomy prediction task and Wang et. al. [35] add language modeling task. Other work use multi-task learning to train a single model that work well on multiple attributes [21, 41].
(2) **Multi-modal learning:** Logan et. al. [19] released Multimodal Attribute Extraction (*MAE*) dataset in 2017 and proposed a simple multimodal model with gated fusion layer,

Zhu et. al. [46] used transformer and ResNet encoders and used self-attention to fuse information while Zhang et. al. [43] proposed Multimodal Graph Fusion model and used Graph Neural Networks to fuse different modalities.

(3) **Few shot learning:** Li et. al. [17] proposed MetaNER and use MAML ([12]) to train a meta learner that quickly adapts to unseen attributes, Yang et. al. [42] proposed a nearest neighbor based few shot model, Wang et. al. [38] proposed self-training model with a meta objective to handle noisy pseudo labels from teacher while Cui et. al. [8] ranked entity templates on different answer spans using pre-trained BART.

(4) **Noise-robust learners:** Chen et. al. [4] proposed masked adversarial model that uses adversarial perturbations on embeddings with an adversarial loss, Zhou et. al. [45] exploit learning properties on noisy labels and use multiple *NER* models with different initializations while Meng et. al. [22] use generalized cross entropy loss to reduce gradient update on false negatives combined with self-training for stability.

These approaches are noteworthy improvements over vanilla IE models and can lift AE performance on products with attribute information in their profiles. However, they fail to address missing and noisy attribute information from recall perspective. Other works that improve vanilla IE technique particularly on recall either predict entire label distribution [16], or use inferred entity dictionaries or augment *k*NN based label distribution to sequence tagging [36]. They too fail to address missing information and are not extendible to an e-commerce scale due to human-in-the-loop. Use of reinforcement learning is not new to information extraction domain. Wang et. al. [37] use *RL* to automatically concatenate different embeddings and is the SOTA on CoNLL 2003 task [33]. Narasimhan et. al. [24] and Liu et. al. [18] use RL for emerging entities by acquiring external evidences using search results. While the latter showed that their formulation work well with emerging entities, it still does not apply directly to an e-commerce setting as 1) long tail products do not have good search results, 2) value edit distances are less insightful and even counter-intuitive (see Section 4.2), and 3) DQN models are unstable [1] and have large number of hyper-parameters making them hard to tune for all attributes.

3 PROBLEM FORMULATION

An *attribute* is a relation between a product and a value that describes some product characteristic. For example, in Figure 1, the product and the value *charcoal* are linked by an attribute - *scent*. It is essential to categorize these attributes to choose the right AE problem formulation. *Open* attributes have large value spaces (more than 20 − 30 or even 100) that can evolve with time, for example *flavor, material,* etc., whereas *closed* attributes have a well defined and fixed value space, for example *target gender* has a fixed value space - {*male, female* and *unisex*}. AE task is defined differently for both attribute types as follows.

Definition 3.1. Open Attribute Extraction (OAE): Given a product P with textual data $x_1, x_2, x_3, ...$ from product profiles $p_1, p_2, p_3, ...$ and a target attribute a, extract all attribute values by predicting the tag sequence $\{t_{i_1}, t_{i_2}, t_{i_3}, ...t_{i_{n_i}}\}$ for each token sequence of the textual data ($x_i = \{w_{i_1}, w_{i_2}, w_{i_3}, ...w_{i_{n_i}}\}$), where $t_\theta \in \{B_a, I_a, O_a\}$.

Definition 3.2. Closed Attribute Extraction (CAE): Given a product P with textual data $x_1, x_2, x_3, ...$ from product profiles $p_1, p_2, p_3, ...$ and a target attribute a, classify text $x^P = concat(x_1, x_2, x_3, ...)$ into a fixed set of attribute classes C, where C is the value space of a.

In Definition 3.1, *OAE* is a NER task to predict *B,I,O* tokens for each attribute (see section 2.2 in [44]). *OAE* can also be defined as a QA task to predict answer spans in the token sequence [2]. *CAE* is a text classification task where the classes are the set of normalized values for the attribute. These formulations do not address missing and noisy attribute information that may result in low model recall. One way to solve for low recall is by ensembling attribute values from product neighbors as defined below.

Definition 3.3. Attribute Value Ensembling from Neighbors: Given a product P, target attribute a, a list of M neighbor attribute values $C_P = [v_1, v_2, v_3...v_M]$ from P and its L nearest product neighbors $\{P, P_1, P_2, P_3, ...P_L\}$ $with$ $L + 1 <= M$[1] and associated catalog data, choose the correct attribute value from C_P for the product P.

3.1 Limitations with non-sequential ensembles

There are numerous ensembling techniques to solve for Definition 3.3. It can be formulated as a ranking problem [3] to find the best ranked neighbor, or as a matching problem [34] using an attribute support set, or as a link prediction problem [32] on an incomplete bipartite graph of products and attribute values, among others. We foresee certain limitations with such non-sequential ensembles:

- **Noisy neighbors**: Product neighbors are noisy and not always relevant to the given product. Non-sequential ensemble model needs to learn from all the neighbors jointly making it hard to locate the correct answer.
- **Product variants**: Variants of a given product [11] are very similar products with incorrect values. Non-sequential learners may give more importance to such products due to likely positive correlation of correct answer with similarity score.
- **Fixed number of neighbors**: Fixing neighbor length (L) for all products is sub-optimal as each product has different number of good quality neighbors.
- **Complexity**: Non-sequential ensembles that learn dense representations of products using all associated data are complex, reduce interpretability and may fail to segregate neighbors with very similar associated data.

Sequential learners are better suited as they exploit ranking among the neighbors and can stop early. This reduces processing noisy neighbors that may occur later in the sequence while allowing dynamic neighbor length for each product. Moreover, conventional RL is more suitable compared to contextual bandits as the problem can be setup with a correlated sequence of states (see Section 4.1.2). With RL-based Lazy-MDP formulation it is possible to promote the top neighbor value and change only if there is a strong signal. This handles product variants and also noisy neighbors while increasing interpretability. We therefore, formulate the problem of attribute value ensemble as the following sequential learning task.

[1]For any particular product neighbor, attribute values can be extracted from multiple sources like values extracted from traditional AE models, value in the catalog provided by vendor, value present on other e-commerce websites, etc.

Definition 3.4. Sequential Attribute Value Ensembling from Neighbors: Given a product P, target attribute a, an ordered sequence of neighbor attribute values C_P and associated catalog data, learn an optimal policy π^* that maximizes the sum of expected rewards by choosing the correct neighbor value from C_P.

4 PAVE: LAZY-MDP BASED ENSEMBLE

We propose PAVE model that solves Definition 3.4 using *Lazy Markov Decision Process (Lazy-MDP)* formulation [14]. A *Lazy-MDP* is a tuple $M_+ = (M, \bar{a}, \bar{\pi}, \eta)$, where M is the standard MDP tuple (S, A, R, T, γ, h) of state, actions, reward, transition probability, discount factor and horizon, $\bar{a}$ is the *lazy action* that defers control to the default policy $\bar{\pi}$ and η is the cost of a non-lazy action $\in A$ [14]. Similar to previous works [18, 24], we adopt model-free RL approach, but explore beyond deep Q-learning [23] due to large variance in optimality of deep Q-networks [1] and multiple hyper-parameters that need to be tuned for each attribute individually. PAVE is a policy network model trained using *Proximal Policy Optimization (PPO)*. PPO trains a stochastic policy using a surrogate objective based on *advantage estimates* to efficiently apply gradient updates within policy trust regions [30]. PPO is easy to tune, uses stochastic exploration and has low variance in its optimal solution due to trust regions. Although we propose PAVE for open attributes where low recall problem generally occurs, we also test on closed attributes for completeness. We describe the training dataset creation, components of our Lazy-MDP and the overall algorithm below.

4.1 Training Dataset Creation

See Figure 3 for the high level training process for the PAVE model, given a product P.

4.1.1 Product Embeddings Space. We use pre-trained Google BERT base model (*bert_uncased_L-12_H-768_A-12*) and train it further on *MLM* and NSP tasks [10] on sampled product titles. Titles are used since they are least noisy. This BERT model is used to generate title embeddings. We collect 100 nearest neighbors based on cosine $similarity$ of these title embeddings for each train and test product. For every neighbor (or a candidate) product, we run baseline AE models (see Section 5.2.1) to obtain predictions along with $confidence$ scores. We also obtain catalog values and assign a constant $confidence$ score. These neighbors are ranked (see section 4.1.2) to create the product embeddings space.

4.1.2 Value Ranking. A natural way to rank the product embeddings space is using similarities between embeddings. However, this results in candidate values being shuffled in the sequence. Since the state is derived from the value (see Section 4.2) the decision process would not satisfy the 1^{st} order Markovian assumption:

$$p(S_{n+1}|S_n) = p(S_{n+1}|S_n, S_{n-1}, ...S_1) \tag{1}$$

where S_i is the state at i^{th} decision step. This is because the probability of occurrence of S_{t+1} or value v_{t+1} depends not only on v_t but also on when it occurred first. For example, a correct first candidate value has high probability of re-occurrence than an incorrect value appearing late in the sequence. Hence, knowing the entire history of states becomes important to learn an optimal policy. Showing historical states to the model adds to the complexity while reducing interpretability. We define $candidate\_confidence$ as $similarity \times confidence$ and sort the product embeddings space by $(candidate\_value, candidate\_confidence)$ tuple to preserve the 1^{st} order Markovian assumption. AE model prediction is always the first candidate value to impart structure to the sequence.

4.1.3 Reference Values. Reference values are used to provide information to the agent about the product category. These are used while creating the state to learn relevance of a candidate value in the category. We create a stratified sample of 50 with same value distribution as P's category as references. Random sampling is avoided so that PAVE observes the same state for repeated inference runs.

4.1.4 Noise Filtering and Augmentation. We apply frequency filtering on candidate and reference values to remove less common values that could be incorrect. We create attribute support set using confident predictions of AE models and remove values that have large minimum distance from the support set. Since AE models predict the correct value for a majority of products, RL agent learns a sub-optimal policy to predict the first value. Hence, we replace some training values by noise ("noise-1", "noise-2", etc.) to ensure uniform distribution of the correct value at different positions in the sequence for the first time while preserving the value ranking.

4.2 State

The state is created at each decision step using various features related to the candidate values. We found that edit distance is less meaningful and sometimes even counter-intuitive for e-commerce attributes. For example, edit distance of *alloy* from *acid* is less than from *alloy crystal emerald*, however *alloy* is more similar to latter. Pre-trained word embeddings like GloVe also do not solve the problem. For example *white* and *black* have closer GloVe representations than *white* and *white gown*. The attribute values are mostly categorical in nature, i.e. distance between say *black* and *blue* should be same as distance between *black* and *white*. Therefore, we use token similarity to compare any two attribute values v_1 and v_2, with token sequence t_i of length n_i for value v_i, $t_i = \{w_{i_1}, w_{i_2}, ...w_{i_{n_i}}\}$:

$$Sim(v_1, v_2) = \begin{cases} 1, & \exists\, w : w \in t_1 \cap t_2 \;\&\; w \notin stop\_words. \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Given a product P and a candidate value v from candidate product Q, we define following features that are relevant to our task.

(1) Confidence Scores - $Confidence(v, Q)$:
   - AE model confidence or fixed catalog score for v
   - Cosine similarity between embeddings of Q and P
   - *TF-IDF* similarity between profiles of Q and P

(2) Frequency scores (using Sim) - $Frequency(v)$:
   - Frequency of v in P's candidate values
   - Frequency of v in P's reference values

(3) Indicator scores - $Indicator(v)$:
   - Is v blank
   - Is v obtained from AE model prediction or catalog
   - Is v the last candidate
   - Is v mentioned in P's profile

Each feature value is normalized between $0 - 1$ and then rounded to a nearest decimal to create buckets. Given a product P, the PAVE model sees following values at each decision step n -

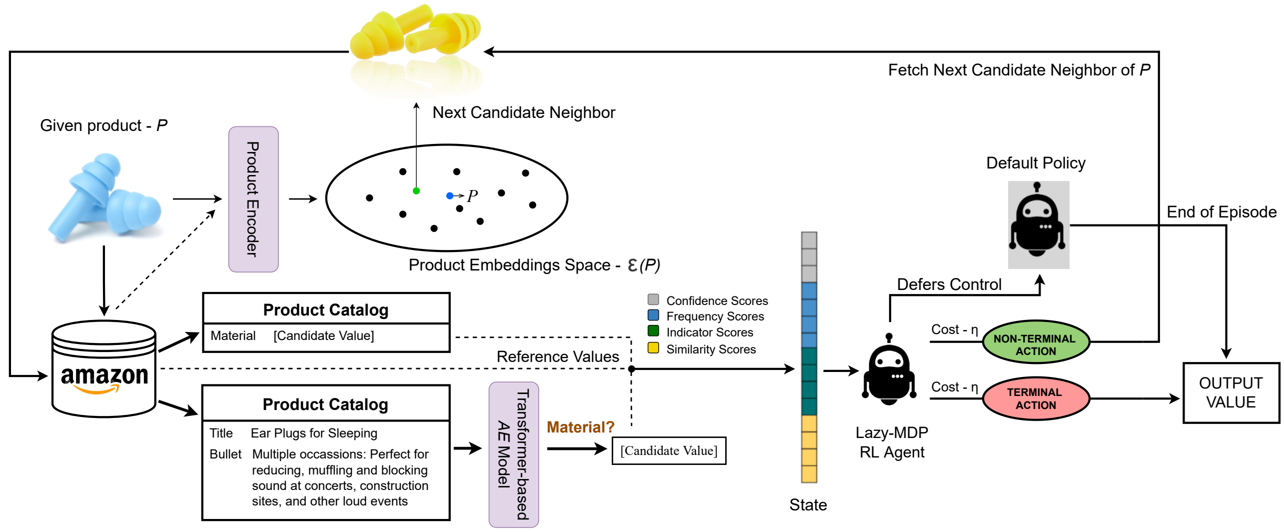

Figure 3: High level overview for training PAVE model. Given a product P, attribute - *material*, catalog value and AE model prediction are used as candidate values. PAVE model learns to choose the best value by either deferring to the default policy or taking an action with a cost of η. Next candidate neighbor of P is fetched to generate new candidate values and the decision process is repeated. At the end of the episode, the best candidate value associated to the last action is taken as the output value.

- av (AE model value) - AE model prediction on P's profiles
- vv (vendor value) - vendor provided value in P's catalog
- bv (best value) - best value so far from neighbor - P_{n*}
- cv (current value) - value from current neighbor - P_n

Note that any of these candidate values could be empty. Also, bv is initialized with av to promote AE model prediction. We define the marginal state (s^m) for value v from product Q as follows,

$$s_v^m = [Confidence(v, Q), Frequency(v), Indicator(v)] \quad (3)$$

State at decision step n (s_n) is defined by subtracting marginal state of all values from s_{bv}^m to assist the PAVE model to choose the candidate value that should replace bv. We also add *similarity scores* to the state to let the PAVE model know if a value is same as bv.

$$s_n = [s_{bv}^m, (s_{cv}^m - s_{bv}^m), (s_{av}^m - s_{bv}^m), (s_{vv}^m - s_{bv}^m), Sim(bv, cv), Sim(bv, av), Sim(bv, vv)] \quad (4)$$

4.3 Actions

Default policy $\bar{\pi}$ simply retains bv throughout the episode. This is a reasonable default policy to consider given the initial bv comes from an existing AE model. At each decision step, the PAVE model either defers control to $\bar{\pi}$ by taking action 0 (lazy action) or incurs a cost of η (see Section 4.4) and takes a non-lazy action as follows. The last bv is taken as the output at the end of episode.

- Non-terminal actions-
  - 1 (av) - replace bv with av, fetch next neighbor
  - 2 (vv) - replace bv with vv, fetch next neighbor
  - 3 (cv) - replace bv with cv, fetch next neighbor
- Terminal actions-
  - 4 (*stop*) - end episode
  - 5 (*blank*) - replace bv with *blank* and end episode

4.4 Rewards

Let the predicted value from the PAVE model be p with ground truth as g. Then, the reward at the end of episode is.

$$Reward(p, g) = \begin{cases} -\alpha, & p = blank. \\ 2 * Sim(p, g) - 1, & otherwise. \end{cases} \quad (5)$$

We penalize *blank* prediction less to encourage PAVE to not predict when there are no good candidate values. We set α to 0.4 empirically as it depends on the number of *blank* answers in the training data. We introduce *intermediate reward* to tip PAVE to an optimal policy. Given state s_n and action a_n, it is defined as follows:

$$IntReward(s_n, a_n) = \begin{cases} W \cdot (s_{av}^m - s_{bv}^m), & a_n = 1 \\ W \cdot (s_{av}^m - s_{vv}^m), & a_n = 2 \\ W \cdot (s_{av}^m - s_{cv}^m), & a_n = 3 \\ \beta, & a_n = 4 \\ \epsilon, & a_n = 5. \end{cases} \quad (6)$$

For actions 1 (av), 2 (vv) and 3 (cv), marginal state difference is normalized by taking a dot product with their feature weights W to compute intermediate rewards. W is set as inverse of average change of the state dimensions in the dataset and scaled down to ensure that the final reward is greater than the highest intermediate reward. To encourage early stopping, which helps in handling noise by processing less neighbors, we set a small positive β. Setting a small discount factor γ did not lead to early stopping, hence we set it to 1 in our experiments. For action 5 (*blank*), a small positive reward ϵ is given if the ground truth is not present in the candidate sequence else 0. At the end of episode, the total reward is $Reward{+}IntReward$. Cost of the non-lazy actions, η is subtracted from $IntReward$. There is no intermediate reward for the lazy action $\bar{a}$ as the default policy skips the decision step. We add a penalty $-\nu$ if PAVE switches to the same value. Hyperparameters β, η, ϵ & ν are selected using grid search on $[0, 0.1, 0.2, 0.3, 0.4, 1]$ and can vary across attributes.

4.5 Algorithm

See Algorithm 1 for the pseudo code to train our PAVE model using PPO. We set discount factor γ to 1 since a correct value is equally important regardless of its position in the candidate sequence. For our experiments, we set horizon h as 20 that was the elbow point for the oracle score on most of the attributes.

Algorithm 1 Pseudo code to train our PAVE model using PPO.

For an attribute, create training data X with product, catalog data (title, bullet, taxonomy) and ground truth $< P_i, Cat_i, g_i >$
Initialize actor network $\pi_\theta(a|s)$ and critic network $Q_w(a, s)$
for $x_i \in X$ **do**
  Collect 100 nearest neighbors for P_i based on product embedding *similarity* and create embeddings space $\mathcal{E}(P_i)$
  Run existing AE model on each product in $\mathcal{E}(P_i)$
  Obtain vendor value for each product in $\mathcal{E}(P_i)$
  Rank & process $\mathcal{E}(P_i)$ according to Section 4.1
  Sample reference values from P_i's taxonomy
end for
for epoch = 1, …,E **do**
  for $i = 1, \ldots, |X|$ **do**
    Pop candidate from $\mathcal{E}(P_i)$ and set value as av and bv
    Pop candidate from $\mathcal{E}(P_i)$ and set value as vv and cv
    Form the state s_1 using Equation (4)
    for $n = 1, \ldots, N$ (parallel actors) **do**
      $done \leftarrow FALSE$
      for $t = 1, \ldots$,h (horizon) **do**
        Sample $a_t \sim \pi_\theta(a|s_t)$ $(\in [0, 5])$
        $r_t \leftarrow 0$
        if $a_t \neq 0$ **then**
          $r_t \leftarrow \eta + IntReward(s_t, a_t)$
          $candidate\_values \leftarrow [av, vv, cv, bv, blank]$
          $bv \leftarrow candidate\_values[a_t - 1]$
          if a_t is a terminal action **then**
            $done \leftarrow TRUE$
            $r_t \leftarrow r_t + Reward(bv, g_i)$
          end if
        end if
        if ($done ==$ False) & $(t \leq h)$ & $(\mathcal{E}(P_i) \neq empty)$ **then**
          $cv \leftarrow$ pop candidate from $\mathcal{E}(P_i)$
        else
          break
        end if
        Form the state s_{t+1} using newly obtained bv & cv
        $\delta \leftarrow r_t + \gamma \max_{a_{t+1}} Q_w(s_{t+1}, a_{t+1}) - Q_w(s_t, a_t)$
        Update $w \leftarrow w + \alpha\gamma^t \delta \nabla_w Q_w(s_t, a_t)$
      end for
      Compute advantage estimates $A_i^n(s_1, a_1), A_i^n(s_2, a_2), \ldots$
    end for
  end for
  Update θ based on PPO's surrogate objective on a minibatch
end for

5 EXPERIMENTS

We test our models on attributes that can typically be inferred from textual product profiles (*title, bullet, OCR text from product images, etc.*). However, our algorithm can be extended to attributes that require both image and text information. We also perform ablation studies on different components we introduce as part of the PAVE model. We propose confidence thresholds for PAVE model outputs and also test PAVE's scalability and generalizability.

5.1 Dataset

To demonstrate performance on open attributes, we handpicked *color, flavor, scent, material* and *container* as these were some of the top attributes that customers care about in certain product categories. Similarly for closed attributes we picked *age range* and *target gender*. We collect samples for each attribute (titles, bullets, description, catalog attribute values) from publicly available product pages in a locale within *amazon.com*. We leverage manual annotators to label around 50 products per product category in each attribute to create our test set. This way we obtain around $12k$ products to test our models (see Table 1). *Training* dataset is obtained by concatenating textual profiles to create a fixed context token vector of size 128 and is used to train baseline AE models. *Taxonomy* dataset is sampled from the catalog such that there is no overlap with the *training* dataset. We also fetch upto 100 nearest neighbors and process these datasets (see section 4.1) to create product embeddings space and reference values for each train and test product.

5.2 Baselines

5.2.1 BERT AE models. Existing AE models are first baselines to our PAVE models. Since PAVE models consume neighbor information, they are likely have better recall, but the comparison is helpful on precision and F1 score for making deployment decisions. Based on the attribute type, we train these baseline AE models using weakly supervised labels provided by vendors. To ensure high quality of the training labels, we use *Snorkel* weak supervision pipeline [28] that learns a generative model over the *labeling functions (LFs)* (that are created from acceptable attribute values provided by catalog teams) followed by a discriminative model for generalization beyond LFs. For open attributes, we fine-tune HuggingFace pre-trained *BertForTokenClassification* model [40] to predict {*B,I,O*} tags to choose the correct attribute tokens through sequence labeling, while for closed attributes, we fine-tune HuggingFace pre-trained

Table 1: Dataset sizes across attributes (in thousands)

Attribute	Attribute Type	Train	Taxonomy	Test
Color	Open	143	145	1.1
Flavor	Open	102	63	2.1
Scent	Open	165	35	1.8
Material	Open	106	79	1.6
Container	Open	40	15	1.5
Joint (open)	Open	556	337	8.1
Age Range	Closed	41	95	1.7
Target Gender	Closed	30	118	1.5
Joint (closed)	Closed	71	159	3.2
Joint (all)	Mixed	627	496	12.3

	Color			**Flavor**			**Scent**			**Material**			**Container**			**All (macro)**		
	Pr%	Re%	F1%	Pr%	Re%	F1%	Pr%	Re%	F1%	Pr%	Re%	F1%	Pr%	Re%	F1%	Pr%	Re%	F1%
Baseline AE model.																		
BERT AE	75.7	39.0	51.5	50.9	56.3	53.5	70.7	41.8	52.5	74.3	43.4	54.8	64.1	13.8	22.7	67.1	38.9	47.0
Baseline ensemble models.																		
First ensemble	54.6	56.3	55.4	41.8	58.0	48.6	28.5	59.6	38.6	43.8	66.9	52.9	45.4	47.8	46.6	42.8	57.7	48.4
Confidence ensemble	50.3	51.9	55.1	42.0	58.7	49.0	27.3	56.7	36.9	41.3	63.2	50.0	40.9	43.0	41.9	40.4	54.7	45.8
Majority ensemble	55.0	56.2	55.6	42.4	58.1	49.0	29.9	59.4	39.8	44.1	66.6	53.1	46.1	46.4	43.9	43.5	57.3	48.7
Model ensemble	70.0	39.0	50.1	55.0	40.7	46.8	48.2	29.7	36.8	64.9	43.8	52.3	52.1	43.4	47.4	58.0	39.3	46.7
PAVE model and its variants.																		
PAVE	**67.8**	**50.5**	**57.9**	**51.8**	**59.8**	**55.5**	51.6	61.9	56.3	69.7	69.4	69.5	57.1	42.1	48.5	59.8	56.7	57.5
BERT AE + PAVE ensemble	67.4	50.2	57.5	49.4	58.9	53.7	51.4	61.3	55.9	69.1	65.0	67.0	56.7	39.6	46.6	58.8	55.0	56.1
PAVE-DQN	52.1	44.2	47.8	53.9	50.3	52.0	51.5	40.5	45.3	61.4	60.4	60.9	56.0	26.4	35.9	55.0	44.4	48.4
BERT AE + PAVE-DQN ensemble	54.8	53.8	54.3	48.6	58.8	53.2	54.4	54.0	54.2	61.5	61.6	61.5	54.1	28.4	37.2	54.7	51.3	52.1
Confidence thresholding on PAVE models.																		
PAVE-maxF1	67.8	50.5	57.9	51.8	59.8	55.5	**64.4**	**52.9**	**58.1**	**72.7**	**67.4**	**69.9**	**61.9**	**40.3**	**48.8**	**63.7**	**54.2**	**58.1**
PAVE-maxPr	76.7	38.4	51.2	66.7	0.9	1.8	71.6	41.0	52.1	78.1	52.5	62.8	73.2	26.8	39.2	73.3	31.9	41.4
PAVE-samePr	75.8	39.9	52.3	51.8	59.8	55.5	70.5	42.8	53.3	74.3	65.2	69.5	63.8	38.5	39.4	67.2	49.2	55.7
Ablation studies.																		
PAVE ∖ {actions *1, 2*}	67.5	49.7	57.2	50.2	59.6	54.5	56.3	58.2	57.2	59.8	67.5	63.4	54.9	41.9	47.5	57.7	55.4	56.0
PAVE ∖ {action *4*}	77.9	22.0	34.3	51.5	56.7	54.0	81.0	2.0	3.9	75.1	48.8	59.2	63.8	33.4	43.8	69.9	32.6	39.0
PAVE ∖ {action *5*}	67.4	46.4	55.0	47.9	60.2	53.4	51.6	59.5	55.3	68.9	65.7	67.3	63.6	34.9	45.1	59.9	53.3	55.2
PAVE ∖ {value ranking}	71.7	45.8	55.9	52.9	57.0	54.8	54.0	59.4	56.6	73.4	61.3	66.8	65.9	29.2	40.5	63.6	50.5	54.9
PAVE ∖ {Lazy-MDP}	72.2	40.6	52.0	50.3	58.9	54.3	50.1	44.5	47.2	71.3	36.8	48.6	53.3	43.5	47.9	59.4	44.9	50.0
PAVE ∖ {reference}	68.6	46.8	55.6	51.1	58.0	54.3	50.5	57.3	53.7	71.8	62.1	66.6	66.3	22.3	33.4	61.7	49.3	52.7
PAVE ∖ {*IntReward*}	70.9	42.4	53.1	51.2	58.0	54.4	23.8	41.2	30.2	71.4	53.8	61.4	54.0	40.9	46.5	54.3	47.3	49.1
PAVE ∖ {reference, *IntReward*}	51.9	11.6	19.0	52.2	55.9	54.0	25.6	47.2	33.2	70.9	53.4	60.9	66.0	28.7	40.0	53.3	39.4	41.4

Table 2: PAVE model performance comparison with different baselines for 5 open attributes. Performance metrics with best F1 score for each attribute are boldfaced, while metrics where both precision and recall are better than baseline BERT AE models are highlighted in teal. For ablation studies, metrics that are better compared to PAVE model are underlined.

BertForSequenceClassification model to predict the correct attribute class through text classification.

5.2.2 *Rule-based ensembles.* Given a product P, we also compare PAVE model against rule-based ensemble models described below.

- First neighbor (First ensemble): We use the AE model prediction if present else P's catalog value if present else the nearest neighbor with a candidate value.
- Most confident (Confidence ensemble): We use the AE model prediction if present else P's catalog value if present else the candidate value with maximum $candidate\_confidence$ in C_P
- Majority vote (Majority ensemble): We use the AE model prediction if present else P's catalog value if present else the candidate value that occurs most frequently in C_P

5.2.3 *Model-based ensemble.* We also compare PAVE model's performance against a simple binary classifier based attribute ensemble model (*Model ensemble*). This model uses a 2 layer fully connected neural network with *ReLU* activation followed by a *sigmoid* output layer that is trained for following problem definition.

Definition 5.1. Attribute Value Ensembling Using a Binary Classifier: Given a product P, a target attribute a, a list of L candidate attribute values C_P, predict for each candidate the probability of being the correct value based on the state used to train PAVE model. For inference predict the probabilities for all L candidates and emit the candidate value with the highest predicted probability.

5.3 Results

Intermediate rewards depend on state that can be noisy, hence, higher reward does not always imply better accuracy. Therefore, we do not compare our models using rewards (as done previously [18]), instead use precision, recall and F1 score. See Table 2 for the results on open attributes (due to limited space), results on closed attributes are discussed in the following paragraph. Metrics for the final column are macro-averaged.

5.3.1 *Comparison with baselines.* BERT AE models achieve an average precision of 67.1% at 38.9% recall on open attributes without confidence thresholding. Among the rule based ensemble models, *First* and *Majority* ensembles outperform BERT AE models in overall F1 score, but lead to 24% drop in precision. *Model* ensemble has better precision than rule based ensembles and even improve F1 score on *container* attribute while reaching similar overall F1 score as BERT AE models. PAVE model consistently outperforms both BERT AE and baseline ensemble models for all attributes in terms of the F1 score. Overall, compared to BERT AE models, PAVE improves recall by 17.8% with 7.3% drop in precision resulting in 10.5% increase in macro-averaged F1 score. On closed attributes, BERT AE models operate at 60.5% precision and 75.7% recall. PAVE models outperform BERT AE models for closed attributes also, with 7.8% lift in precision and 1.5% lift in recall (see Table 4). Compared to *First* and *Majority* ensembles, PAVE achieve 1% lower recall at 16.7% better precision. PAVE models not only predict values when BERT

AE models do not, but also correct wrong predictions. To demonstrate this, we create BERT AE + PAVE ensemble model where PAVE is run only on products with missing BERT AE prediction. This ensemble model falls behind PAVE on all 3 metrics.

We break up PAVE recall of 56.7% with respect to the position of the selected correct value in the candidate sequence of length 20 (see Figure 4). 66% of the predicted correct values of PAVE come from BERT AE model or *Candidate* 1. Next majority of 19% comes from the nearest product neighbor or *Candidate* 3. While the rest is uniformly selected from *Candidate* 4 to 20. Rule based ensemble give low precision as they are not complex enough to identify where the correct value is present in this sequence. Notably, vendor values do not contribute much as vendors also mention the attribute value in the profile that gets predicted by BERT AE model. We still discuss action 2 as it may be useful in other datasets.

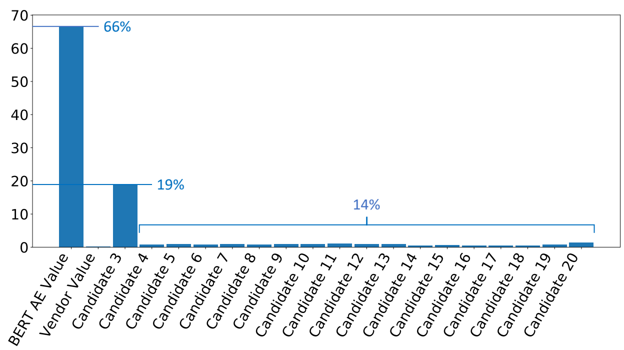

Figure 4: Breaking up recall lift from PAVE model by the position of the selected value in the candidate sequence. To plot the graph, we aggregate results for all open attributes.

5.3.2 DQN vs Policy Networks. We compare previously used DQN with policy networks optimized using PPO. We train PAVE-DQN model with the same Lazy-MDP formulation but using DQN initialized same as [18]. PAVE-DQN outperforms BERT AE models on 2 out of 5 open attributes and also overall F1 score. However, it achieves 9.1% less overall F1 score compared to PAVE. BERT AE + PAVE-DQN ensemble consistently outperforms PAVE-DQN model in F1 score implying that PAVE-DQN can be inaccurate even when BERT AE model was correct. We also compare the two algorithms on the Chinese dataset provided by Liu et. al. [18]. We train a model in the same environment and change to RLlib PPO implementation instead of DQN. PPO model consistently achieves higher overall rewards for all 3 extraction models on their dataset (see Table 3).

5.3.3 Confidence Thresholding. Confidence scores are important to control the precision-recall tradeoff, especially when operating at a high precision is a must. Like any deep RL model, PAVE does not emit confidence scores on its predictions. There is some work on obtaining the confidence scores by measuring model uncertainty, for example measuring variance in Q-values on similar states and by using auxiliary networks [5, 7]. However, well calibrated intermediate rewards can be used as confidence scores for the entire episode. This is feasible because of the Lazy-MDP formulation that makes the RL agent take an action only on important states [14]. We found that the PAVE model changes the BERT AE value (or the initial *bv*) only once in the entire episode. Hence, we use the intermediate rewards obtained on the latest action when *bv* was changed as the confidence score of the model. In case there is no change of value, we consider model confidence to be 1. Using these confidence scores, we apply thresholding to control precision-recall tradeoff. Thresholds can be tuned to optimize for F1 score (*PAVE-maxF1*) or precision (*PAVE-maxPr*). *PAVE-maxF1* model achieves 0.6% better F1 score than PAVE and *PAVE-maxPr* achieves 13.5% better precision than PAVE but suffers recall loss. We were able to tune confidence thresholds to match BERT AE model precision (*PAVE-samePr*). *PAVE-samePr* model reaches same precision on all attributes and improves overall recall and F1 score by 10.3% and 8.7% respectively compared to BERT AE models (see Table 2). We find that the metrics change smoothly as the confidence thresholds are increased for most attributes (shown for *scent* in Figure 5). However, for flavor the metrics change abruptly after a threshold due to poor features of product neighbors that yield the correct value.

	Evaluating Dataset				
	GPU	Games	Movie	Phone	All
BiDAF with DQN & PPO methods and Oracle strategy.					
DQN (BiDAF)	**0.786**	0.692	**0.686**	0.739	0.726
PPO (BiDAF)	0.765	**0.739**	0.669	**0.769**	**0.736**
Oracle (BiDAF)	0.902	0.793	0.846	0.812	0.838
QANet with DQN & PPO methods and Oracle strategy.					
DQN (QANet)	0.786	0.687	**0.731**	0.790	0.749
PPO (QANet)	**0.806**	**781**	0.691	**0.821**	**0.775**
Oracle (QANet)	0.932	0.840	0.878	0.868	0.880
BERT with DQN & PPO methods and Oracle strategy.					
DQN (BERT)	0.817	0.637	**0.777**	0.837	0.767
PPO (BERT)	**0.819**	**0.781**	0.727	**0.872**	**0.800**
Oracle (BERT)	0.925	0.857	0.887	0.909	0.895

Table 3: Performance comparison between PPO and DQN models on the dataset provided by Liu et. al. [18]. Higher rewards are boldfaced.

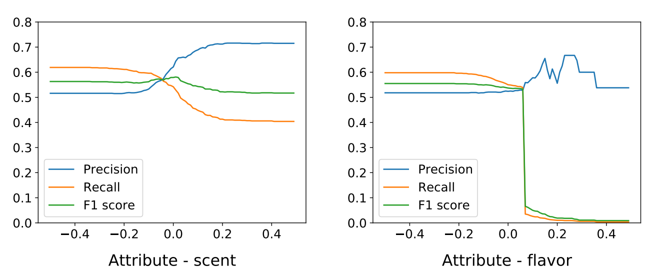

Figure 5: Impact of confidence thresholding on performance metrics for *scent* and *flavor* attributes.

5.3.4 Ablation Studies. We perform several ablation studies to test the various components we use in our PAVE models (see Table 2). In principal, PAVE model can emit av or vv without actions $1(av)$ or $2(vv)$. However, removing these actions lead to some drop in performance. With actions 1, 2 the agent switches back to a correct value if it makes a wrong choice in its initial learning stage, that expands the policy search space by not ending in a bad reward when the correct value does not re-occur later in the sequence. We also train without actions 4 ($stop$) and actions 5 ($blank$). Removing action 4 results in much lower F1 score and recall as the model predicts more $blank$ values. Without action 5, model precision drops on 4 out of 5 attributes since it makes a prediction even if the the correct value is absent from the candidate sequence. We also test the usefulness of *value ranking* by using only the $candidate\_confidence$ to rank. Due to unpredictable value sequences, the model learns unintuitive policies like takes action 3 on last candidate or takes action 3 on first non-blank candidate and then takes action 4. It takes more steps and still identifies good candidates and hence better precision than PAVE but fails on other metrics.

PAVE model trained without Lazy-MDP dimensions result in lower performance on all 3 metrics. The model learns a policy that maximizes intermediate rewards by selecting a good value repeatedly (taking actions 1 and 3 alternatively), sometimes at the cost of the final reward. This also reduces interpretability and the same confidence scores cannot be used now. Removing reference values result in lower recall as the model fails to identify common values in the product category that seldom occur in the candidate sequence. Removing intermediate rewards result in both low precision and recall as it fails to correlate final reward with the state dimensions. Removing both reference and intermediate rewards drop performance of all metrics even further.

5.3.5 Scaling PAVE Model. We train joint PAVE models by combining different types of attributes together. We found that single PAVE models trained on all open attributes (*PAVE-open*) still outperforms individual BERT AE models by 7.6% in F1 score. *PAVE-open* is more conservative in choosing new values thereby having a better precision at a lower recall than PAVE models. Upon running inference on unseen closed attributes, *PAVE-open* outperforms both BERT AE and generalizes better than PAVE models (see Table 4). Similarly, we train a *PAVE-closed* model on all closed attributes. This model performs at par on unseen open attributes compared to individual BERT AE models while surpassing on closed attributes. These models too fall short compared to PAVE models. We also train *PAVE-all* model combining all the dataset together. We find that this model

	All (open)			**All (closed)**		
	Pr%	Re%	F1%	Pr%	Re%	F1%
BERT AE	67.1	38.9	47.0	60.5	75.7	67.2
PAVE	59.8	56.7	57.5	68.3	77.2	72.5
PAVE-open	60.7	51.8	54.6	71.5	77.3	73.9
PAVE-closed	67.5	38.9	47.1	62.2	75.3	67.9
PAVE-all	51.4	56.7	53.0	62.8	78.6	70.0

Table 4: Joint PAVE models performance compared to baseline BERT AE models and individual PAVE models.

also outperforms BERT AE models in terms of F1 score for both type of attributes. We also train *PAVE-open3* model on *color, flavor* and *scent* and test on *material* and *container* attributes. This model surpasses BERT AE models by 14% in average F1 score but has 6.1% less F1 score compared to PAVE models. These joint models demonstrate the scalability and generalizability of our approach where a single model can work on unseen attributes as well, bringing down the operational burden of maintaining multiple ML models in production. However, such models come at the cost of slightly lower F1 score compared to individual PAVE models that can fit well on a particular attribute.

6 CONCLUSION AND FUTURE WORK

We presented PAVE - a Lazy-MDP based ensemble model to improve recall of existing *AE* models in an e-commerce setting. Our model is robust to noisy product neighbors and allows dynamic neighbor length for ensembling. We report 10.3% improvement in recall on open attributes compared to BERT-based *AE* models with no drop in precision. Our method outperforms simple ensembling techniques and also performs well on closed attributes. We propose novel intermediate rewards that coupled with Lazy-MDP formulation increases PAVE's performance and interpretability as it identifies the important states to take an action. We also propose confidence thresholds to control precision-recall tradeoff that makes our model easy to adopt. Our model is scalable at an e-commerce scale and generalizes well, where a single PAVE model can be deployed on several attributes to obtain recall lift even on unseen attributes compared to BERT-based AE models. In future, we wish to experiment with offline RL methods that may offer better generalization by stitching good parts to suboptimal parts in trajectories of online PAVE model while operating at a lower latency.

REFERENCES

[1] Oron Anschel, Nir Baram, and Nahum Shimkin. 2016. Averaged-DQN: Variance Reduction and Stabilization for Deep Reinforcement Learning. https://doi.org/10.48550/ARXIV.1611.01929

[2] Tarik Arici, Kushal Kumar, Hayreddin Çeker, Anoop S V K K Saladi, and Ismail Tutar. 2021. Solving Price Per Unit Problem Around the World: Formulating Fact Extraction as Question Answering. arXiv:2204.05555 [cs.CL]

[3] Christopher Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning* 11 (01 2010).

[4] Luoxin Chen, Xinyue Liu, Weitong Ruan, and Jianhua Lu. 2020. Enhance Robustness of Sequence Labelling with Masked Adversarial Training. In *Findings of the Association for Computational Linguistics: EMNLP 2020.* Association for Computational Linguistics, Online, 297–302. https://doi.org/10.18653/v1/2020.findings-emnlp.28

[5] Si-An Chen, Voot Tangkaratt, Hsuan-Tien Lin, and Masashi Sugiyama. 2019. Active deep Q-learning with demonstration. *Machine Learning* 109, 9-10 (nov 2019), 1699–1725. https://doi.org/10.1007/s10994-019-05849-4

[6] Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc V. Le. 2018. Semi-Supervised Sequence Modeling with Cross-View Training. https://doi.org/10.48550/ARXIV.1809.08370

[7] William R. Clements, Bastien Van Delft, Benoît-Marie Robaglia, Reda Bahi Slaoui, and Sébastien Toth. 2019. Estimating Risk and Uncertainty in Deep Reinforcement Learning. https://doi.org/10.48550/ARXIV.1905.09638

[8] Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. 2021. Template-Based Named Entity Recognition Using BART. https://doi.org/10.48550/ARXIV.2106.01760

[9] Aloïs De la Comble, Anuvabh Dutt, Pablo Montalvo, and Aghiles Salah. 2022. Multi-Modal Attribute Extraction for E-Commerce. https://doi.org/10.48550/ARXIV.2203.03441

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. https://doi.org/10.48550/ARXIV.1810.04805

[11] H. ElMaraghy, G. Schuh, W. ElMaraghy, F. Piller, P. Schönsleben, M. Tseng, and A. Bernard. 2013. Product variety management. *CIRP Annals* 62, 2 (2013), 629–652. https://doi.org/10.1016/j.cirp.2013.05.007

[12] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. https://doi.org/10.48550/ARXIV.1703.03400

[13] Rayid Ghani, Katharina Probst, Yan Liu, Marko Krema, and Andrew Fano. 2006. Text Mining for Product Attribute Extraction. *SIGKDD Explor. Newsl.* 8, 1 (jun 2006), 41–48. https://doi.org/10.1145/1147234.1147241

[14] Alexis Jacq, Johan Ferret, Olivier Pietquin, and Matthieu Geist. 2022. Lazy-MDPs: Towards Interpretable Reinforcement Learning by Learning When to Act. (2022). https://doi.org/10.48550/ARXIV.2203.08542

[15] Giannis Karamanolakis, Jun Ma, and Xin Luna Dong. 2020. TXtract: Taxonomy-Aware Knowledge Extraction for Thousands of Product Categories. https://doi.org/10.48550/ARXIV.2004.13852

[16] Jasper Kuperus, Cor Veenman, and Maurice Van Keulen. 2013. Increasing NER Recall with Minimal Precision Loss. 106–111. https://doi.org/10.1109/EISIC.2013.23

[17] Jing Li, Shuo Shang, and Ling Shao. 2020. *MetaNER: Named Entity Recognition with Meta-Learning.* Association for Computing Machinery, New York, NY, USA, 429–440. https://doi.org/10.1145/3366423.3380127

[18] Ye Liu, Sheng Zhang, Rui Song, Suo Feng, and Yanghua Xiao. 2020. Knowledge-guided Open Attribute Value Extraction with Reinforcement Learning. https://doi.org/10.48550/ARXIV.2010.09189

[19] Robert L. Logan, Samuel Humeau, and Sameer Singh. 2017. Multimodal Attribute Extraction. https://doi.org/10.48550/ARXIV.1711.11118

[20] Karin Mauge, Khash Rohanimanesh, and Jean-David Ruvini. 2012. Structuring E-Commerce Inventory. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Jeju Island, Korea, 805–814. https://aclanthology.org/P12-1085

[21] Kartik Mehta, Ioana Oprea, and Nikhil Rasiwasia. 2021. LaTeX-Numeric: Language-agnostic Text attribute eXtraction for E-commerce Numeric Attributes. https://doi.org/10.48550/ARXIV.2104.09576

[22] Yu Meng, Yunyi Zhang, Jiaxin Huang, Xuan Wang, Yu Zhang, Heng Ji, and Jiawei Han. 2021. Distantly-Supervised Named Entity Recognition with Noise-Robust Learning and Language Model Augmented Self-Training. https://doi.org/10.48550/ARXIV.2109.05003

[23] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. https://doi.org/10.48550/ARXIV.1312.5602

[24] Karthik Narasimhan, Adam Yala, and Regina Barzilay. 2016. Improving Information Extraction by Acquiring External Evidence with Reinforcement Learning. https://doi.org/10.48550/ARXIV.1603.07954

[25] Cristian Padurariu and Mihaela Elena Breaban. 2019. Dealing with Data Imbalance in Text Classification. *Procedia Computer Science* 159 (2019), 736–745. https://doi.org/10.1016/j.procs.2019.09.229 Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 23rd International Conference KES2019.

[26] Petar Petrovski and Christian Bizer. 2017. Extracting Attribute-Value Pairs from Product Specifications on the Web. In *Proceedings of the International Conference on Web Intelligence* (Leipzig, Germany) *(WI '17)*. Association for Computing Machinery, New York, NY, USA, 558–565. https://doi.org/10.1145/3106426.3106449

[27] Disheng Qiu, Luciano Barbosa, Xin Luna Dong, Yanyan Shen, and Divesh Srivastava. 2015. Dexter: Large-Scale Discovery and Extraction of Product Specifications on the Web. *Proc. VLDB Endow.* 8, 13 (sep 2015), 2194–2205. https://doi.org/10.14778/2831360.2831372

[28] Alexander Ratner, Stephen H. Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré . 2017. Snorkel. *Proceedings of the VLDB Endowment* 11, 3 (nov 2017), 269–282. https://doi.org/10.14778/3157794.3157797

[29] Martin Rezk, Laura Alonso Alemany, Lasguido Nio, and Ted Zhang. 2019. Accurate Product Attribute Extraction on the Field. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. 1862–1873. https://doi.org/10.1109/ICDE.2019.00202

[30] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. https://doi.org/10.48550/ARXIV.1707.06347

[31] Shengjie Sun, Dong Yang, Hongchun Zhang, Yanxu Chen, Chao Wei, Xiaonan Meng, and Yi Hu. 2018. Important Attribute Identification in Knowledge Graph. https://doi.org/10.48550/ARXIV.1810.05320

[32] Ben Taskar, Ming-fai Wong, Pieter Abbeel, and Daphne Koller. 2003. Link Prediction in Relational Data. In *Advances in Neural Information Processing Systems*, S. Thrun, L. Saul, and B. Schölkopf (Eds.), Vol. 16. MIT Press. https://proceedings.neurips.cc/paper/2003/file/1e0a84051e6a4a7381473328f43c4884-Paper.pdf

[33] Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*. 142–147. https://aclanthology.org/W03-0419

[34] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. 2016. Matching Networks for One Shot Learning. https://doi.org/10.48550/ARXIV.1606.04080

[35] 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* (Virtual Event, CA, USA) *(KDD '20)*. Association for Computing Machinery, New York, NY, USA, 47–55. https://doi.org/10.1145/3394486.3403047

[36] Shuhe Wang, Xiaoya Li, Yuxian Meng, Tianwei Zhang, Rongbin Ouyang, Jiwei Li, and Guoyin Wang. 2022. *k*NN-NER: Named Entity Recognition with Nearest Neighbor Search. https://doi.org/10.48550/ARXIV.2203.17103

[37] Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2020. Automated Concatenation of Embeddings for Structured Prediction. https://doi.org/10.48550/ARXIV.2010.05006

[38] Yaqing Wang, Subhabrata Mukherjee, Haoda Chu, Yuancheng Tu, Ming Wu, Jing Gao, and Ahmed Hassan Awadallah. 2021. Meta Self-Training for Few-Shot Neural Sequence Labeling. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (Virtual Event, Singapore) *(KDD '21)*. Association for Computing Machinery, New York, NY, USA, 1737–1747. https://doi.org/10.1145/3447548.3467235

[39] Yaqing Wang, Yifan Ethan Xu, Xian Li, Xin Luna Dong, and Jing Gao. 2020. Automatic Validation of Textual Attribute Values in E-commerce Catalog by Learning with Limited Labeled Data. https://doi.org/10.48550/ARXIV.2006.08779

[40] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. https://doi.org/10.48550/ARXIV.1910.03771

[41] Jun Yan, Nasser Zalmout, Yan Liang, Christan Grant, Xiang Ren, and Xin Luna Dong. 2021. AdaTag: Multi-Attribute Value Extraction from Product Profiles with Adaptive Decoding. https://doi.org/10.48550/ARXIV.2106.02318

[42] Yi Yang and Arzoo Katiyar. 2020. Simple and Effective Few-Shot Named Entity Recognition with Structured Nearest Neighbor Learning. https://doi.org/10.48550/ARXIV.2010.02405

[43] Dong Zhang, Suzhong Wei, Shoushan Li, Hanqian Wu, Qiaoming Zhu, and Guodong Zhou. 2021. Multi-modal Graph Fusion for Named Entity Recognition with Targeted Visual Guidance. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 16 (May 2021), 14347–14355. https://ojs.aaai.org/index.php/AAAI/article/view/17687

[44] Guineng Zheng, Subhabrata Mukherjee, Xin Luna Dong, and Feifei Li. 2018. OpenTag. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM. https://doi.org/10.1145/3219819.3219839

[45] Wenxuan Zhou and Muhao Chen. 2021. Learning from Noisy Labels for Entity-Centric Information Extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.emnlp-main.437

[46] Tiangang Zhu, Yue Wang, Haoran Li, Youzheng Wu, Xiaodong He, and Bowen Zhou. 2020. Multimodal Joint Attribute Prediction and Value Extraction for E-commerce Product. https://doi.org/10.48550/ARXIV.2009.07162