OCTET: Online Catalog Taxonomy Enrichment with Self-Supervision

Yuning Mao[1], Tong Zhao[2], Andrey Kan[2], Chenwei Zhang[2],
Xin Luna Dong[2], Christos Faloutsos[3], Jiawei Han[1]
[1]University of Illinois at Urbana-Champaign [2]Amazon [3]Carnegie Mellon University
[1]{yuningm2, hanj}@illinois.edu [2]{zhaoton, avkan, cwzhang, lunadong}@amazon.com [3]christos@cs.cmu.edu

ABSTRACT

Taxonomies have found wide applications in various domains, especially online for item categorization, browsing, and search. Despite the prevalent use of online catalog taxonomies, most of them in practice are maintained by humans, which is labor-intensive and difficult to scale. While *taxonomy construction from scratch* is considerably studied in the literature, how to effectively enrich existing incomplete taxonomies remains an open yet important research question. Taxonomy enrichment not only requires the robustness to deal with emerging terms but also the consistency between existing taxonomy structure and new term attachment. In this paper, we present a self-supervised end-to-end framework, OCTET, for Online Catalog Taxonomy EnrichmenT. OCTET leverages heterogeneous information unique to online catalog taxonomies such as user queries, items, and their relations to the taxonomy nodes while requiring no other supervision than the existing taxonomies. We propose to distantly train a sequence labeling model for term extraction and employ graph neural networks (GNNs) to capture the taxonomy structure as well as the query-item-taxonomy interactions for term attachment. Extensive experiments in different online domains demonstrate the superiority of OCTET over state-of-the-art methods via both automatic and human evaluations. Notably, OCTET enriches an online catalog taxonomy in production to *2 times larger* in the open-world evaluation.

ACM Reference Format:
Yuning Mao[1], Tong Zhao[2], Andrey Kan[2], Chenwei Zhang[2], and Xin Luna Dong[2], Christos Faloutsos[3], Jiawei Han[1]. 2020. OCTET: Online Catalog Taxonomy Enrichment with Self-Supervision. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA*, 11 pages. https://doi.org/10.1145/3394486.3403274

1 INTRODUCTION

Taxonomies, the tree-structured hierarchies that represent the hypernymy (Is-A) relations, have been widely used in different domains, such as information extraction [5], question answering [35], and recommender systems [9], for the organization of concepts and instances as well as the injection of structured knowledge in downstream tasks. In particular, online catalog taxonomies serve as a building block of e-commerce websites (*e.g.*, Amazon.com) and business directory services (*e.g.*, Yelp.com) for both customer-facing and internal applications, such as query understanding, item categorization [18], browsing, recommendation [9], and search [33].

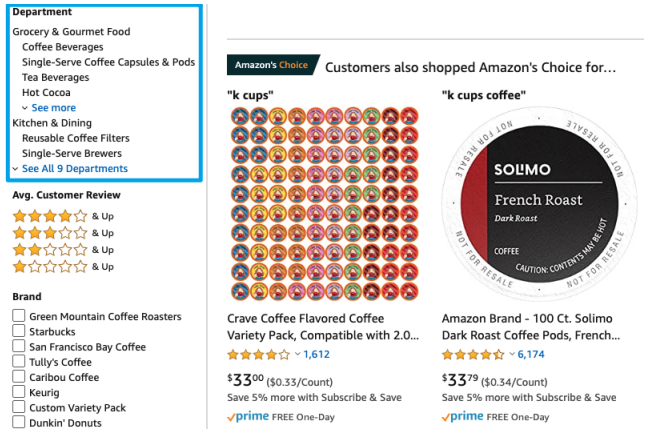

Figure 1: The most relevant taxonomy nodes are shown on the left when a user searches "k cups" on Amazon.com.

Fig. 1 shows one real-world example of how the product taxonomy at Amazon.com is used to facilitate online shopping experience: when a user searches "k cups", the most relevant nodes (types) in the taxonomy *Grocery & Gourmet Food* are shown on the left sidebar. The taxonomy here serves multiple purposes. First, the user can browse relevant nodes to refine the search space if she is looking for a more general or specific type of items (*e.g.*, "Coffee Beverages"). Second, the taxonomy benefits query understanding by identifying that "k cups" belongs to the taxonomy *Grocery & Gourmet Food* and mapping the user query "k cups" to the corresponding taxonomy node "Single-Serve Coffee Capsules & Pods". Third, the taxonomy allows query relaxation and makes more items searchable if the search results are sparse. For instance, not only "Single-Serve Coffee Capsules & Pods" but also other coffee belonging to its parent type "Coffee Beverages" can be shown in the search results.

Despite the prevalent use and benefits of online catalog taxonomies, most of them in practice are still built and maintained by human experts. Such manual practice embodies knowledge from the experts but is meanwhile labor-intensive and difficult to scale. On Amazon.com, the taxonomies usually have thousands of nodes, not necessarily enough to cover the types of billions of items: we sampled roughly 3 million items in Grocery domain on Amazon.com and found that over 70% of items do not directly mention the types in the taxonomy, implying a mismatch between knowledge organization and item search. As a result, automatic taxonomy construction has drawn significant attention.

KDD '20, August 23–27, 2020, Virtual Event, CA, USA

ACM ISBN 978-1-4503-7998-4/20/08.
https://doi.org/10.1145/3394486.3403274

Existing methods on taxonomy construction fail to work effectively on online catalog taxonomies for the following reasons. Most prior methods [1, 3, 7, 13, 17, 30] are designed for taxonomy construction from general text corpora (*e.g.*, Wikipedia), limiting their applicability to text-rich domains. The "documents" in e-commerce (*e.g.*, item titles), however, are much shorter and pose particular challenges. First, it is implausible to extract terms with heuristic approaches [13] from item titles and descriptions, since vendors can write them in arbitrary ways. Second, it is highly unlikely to find co-occurrences of hypernym pairs in the item titles due to their conciseness, making Hearst patterns [8, 21] and dependency parse-based features [17] infeasible. For instance, one may often see "US" and "Seattle" in the same document, but barely see "Beverages" and "Coffee" in the same item title. Third, blindly leveraging the co-occurrence patterns could be misleading: in an item titled "*Triple Scoop Ice Cream Mix, Premium Strawberry*", "Strawberry" and "Ice Cream" co-occur but "Strawberry" is the flavor of the "Ice Cream" rather than its hypernym. The situation worsens as online catalog taxonomies are never static. There are new items (and thus new terms) emerging every day, making *taxonomy construction from scratch* less favorable, since in practice we cannot afford to rebuild the whole taxonomy frequently and the downstream applications also require stable taxonomies to organize knowledge.

To tackle the above issues of *taxonomy construction from scratch*, we target the *taxonomy enrichment* problem, which discovers emerging concepts[1] and attaches them to the existing taxonomy (named *core taxonomy*) to precisely understand new customer interests. Different from taxonomy construction from scratch, the core taxonomies, which are usually built and maintained by experts for quality control and actively used in production, provide both valuable guidance and restrictive challenges for taxonomy enrichment. On the challenge side, the *core taxonomy* requires term attachment to follow the existing taxonomy schema instead of arbitrarily building from scratch. On the bright side, we can base our work on the core taxonomy, which usually contains high-level and qualified concepts representing the fundamental categories in a domain (such as "Beverages" and "Snacks" for Grocery) and barely needs modification (or cannot be automatically organized due to business demands), but lacks fine-grained and emerging terms (such as "Coconut Flour"). There are only a few prior works focused on taxonomy enrichment, which either employ simple rules [10] or represent taxonomy nodes by their associated items and totally neglect the lexical semantics of the concepts themselves [33]. In addition, prior studies [11, 24] require manual training data and fail to exploit the structural information of the existing taxonomy.

Despite the challenges, a unique opportunity for online catalog taxonomy enrichment is the availability of rich user behavior logs: vendors often carefully choose words to describe the type of their items and associate the items with appropriate taxonomy nodes to get more exposure; customers often include the item type in their queries and the majority of the clicked (purchased) items are instances of the type they are looking for. Such interactions among queries, items, and taxonomy nodes offer distinctive signals for hypernymy detection, which is unavailable in general-purpose text corpora. For instance, if a query mentioning "hibiscus tea" leads to the clicks of items associated with taxonomy node "herbal tea" or "tea", we can safely infer strong connections among "hibiscus tea", "herbal tea", and "tea". Existing works [15, 33], however, only utilize the user behavior heuristically to extract new terms or reduce the prediction space of hypernymy detection.

In this paper, we present a self-supervised end-to-end framework, Octet, for online catalog taxonomy enrichment. Octet is novel in three aspects. First, Octet identifies new terms from item titles and user queries; it employs a sequence labeling model that is shown to significantly outperform typical term extraction methods. Second, to tackle the lack of text corpora, Octet leverages heterogeneous sources of signals; it captures the *lexical semantics* of the terms and employs Graph Neural Networks (GNNs) to model the *structure of the core taxonomy* as well as the *query-item-taxonomy interactions* in user behavior. Third, Octet requires no human effort for generating training labels as it uses the core taxonomy for self-supervision during both term extraction and term attachment. We conduct extensive experiments on real-world online catalog taxonomies to verify the effectiveness of Octet via automatic, expert, and crowdsourcing evaluations. Experimental results show that Octet outperforms state-of-the-art methods by 46.2% for term extraction and 11.5% for term attachment on average. Notably, Octet doubles the size (2,163 to 4,355 terms) of an online catalog taxonomy in production with 0.881 precision.

Contributions. (1) We introduce a self-supervised end-to-end framework, Octet, for online catalog taxonomy enrichment; Octet automatically extracts emerging terms and attaches them to the core taxonomy of a target domain with no human effort. (2) We propose a GNN-based model that leverages heterogeneous sources of information, especially the structure of the core taxonomy and query-item-taxonomy interactions, for term attachment. (3) Our extensive experiments show that Octet significantly improves over state-of-the-art methods under automatic and human evaluations.

2 TASK FORMULATION

Notations. We define a taxonomy $T = (V, R)$ as a tree-structured hierarchy with term set V and edge set R. A term $v \in V$ can be either single-word or multi-word (*e.g.*, "Yogurt" and "Herbal Tea"). The edge set R indicates the Is-A relationship between V (hypernym pairs such as "Coffee" -> "Ground Coffee"). The online catalog taxonomies, which can be found in almost all online shopping websites such as Amazon.com and eBay, and business directory services like Yelp.com, maintain the hypernym relationship of their items (*e.g.*, products or businesses). We define a *core taxonomy* as a pre-given partial taxonomy that is usually manually curated and stores the high-level concepts in the target domain. We denote user behavior logs as $B = (Q, I)$, which record the user queries Q in a search engine and corresponding clicked items I. The items I are represented by item profiles such as titles and descriptions. I is associated with (assigned to) nodes V according to their types by item categorization (done by vendors or algorithms).

Problem Definition. Let $T = (V, R)$ be a core taxonomy and $B = (Q, I)$ be user behavior logs, the *taxonomy enrichment* problem extends T to $\bar{T} = (\bar{V}, \bar{R})$ with $\bar{V} = V \cup V'$, $\bar{R} = R \cup R'$, where V' contains new terms extracted from Q and I, and R' contains pairs $(v, v'), v \in V, v' \in V'$, representing that v is a hypernym of v'.

[1] We use "concept", "term", "type", "category", and "node" interchangeably.

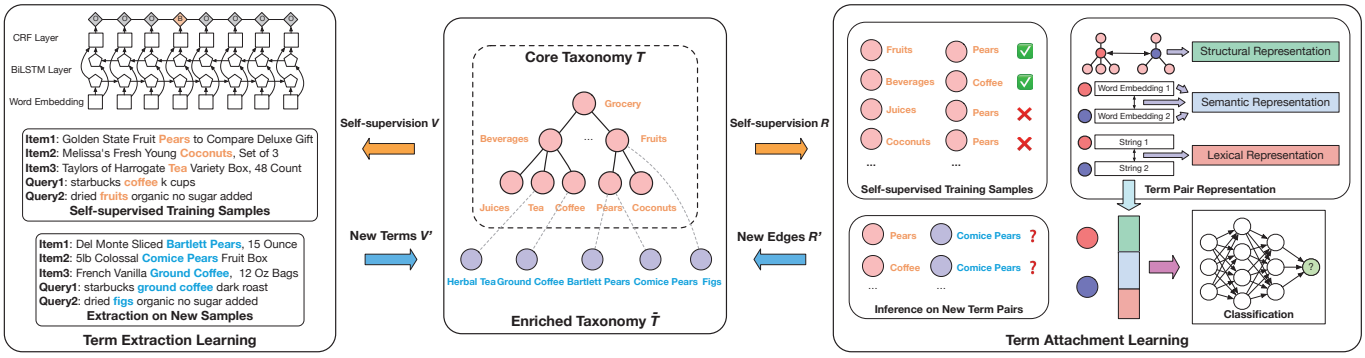

Figure 2: An overview of the proposed framework Octet. Item profiles and user queries in the target domain serve as framework input and the core taxonomy is used as self-supervision. New terms are automatically extracted via sequence labeling. Heterogeneous sources of signals, including the structure of the core taxonomy, the query-item-taxonomy interactions, and the lexical semantics of the term pairs are leveraged for hypernymy detection during term attachment.

Non-Goals. Although Octet works for terms regardless of their granularity, we keep T unchanged as in [11, 33] since we would like to keep the high-level expert-curated hypernym pairs intact and focus on discovering fine-grained terms. Following convention [11], we do not identify the hypernym relationship between newly discovered types ($(v'_1, v'_2), v'_1, v'_2 \in V'$). Octet already makes an important first step to solve the problem as our analysis shows that over 80% terms are leaf nodes in the core taxonomy.

3 THE OCTET FRAMEWORK

In this section, we first give a framework overview of the learning goal of term extraction and term attachment. We then elaborate on how to employ self-supervision of the core taxonomy to conduct term extraction via sequence labeling (Sec. 3.2) and term attachment via GNN-based heterogeneous representation (Sec. 3.3).

3.1 Framework Overview

Octet consists of two inter-dependent stages: **term extraction** and **term attachment**, which, in a nutshell, solves the problem of "which terms to attach" and "where to attach", respectively. Formally, the term extraction stage extracts the new terms V' (with the guidance from V) that are to be used for enriching T to $\bar{T}$. The term attachment stage takes T and V' as well as the sources of V' (*i.e.*, Q and I) as input and attaches each new term $v' \in V'$ to a term $v \in V$ in T, forming the new edge set R'. Octet is readily deployable to different domains with no human effort, *i.e.*, with no additional resources other than T, Q, I, and their interactions.

Fig. 2 shows an overview of Octet. At a high level, we regard term extraction as a sequence labeling task and employ a sequence labeling model with distant supervision from V to extract new terms V'. We show in Sec. 4.1 that such a formulation is beneficial for the term extraction in online catalog taxonomy enrichment. For term attachment, existing hypernym pairs R on T are used as self-supervision. Structure of the core taxonomy as well as interactions among queries, items, and taxonomy nodes are captured via graph neural networks (GNNs) for structural representation learning. Meanwhile, the lexical semantics of the terms is employed and provides complementary signals for hypernymy detection (Sec. 4.4.1). Each new term $v' \in V'$ is attached to one existing term $v \in V$ on T based on the term pair representation.

3.2 Term Extraction Learning

Term extraction extends $T = (V, R)$ with new terms V'. Extracting new terms from item profiles I and user queries Q has two benefits. First, they are closely related to the dynamics of customer needs, which is essential for the enrichment of user-oriented online catalog taxonomies and their deployment in production. Second, extraction from I and Q naturally connects type terms to user behavior, preparing rich signals required for generating structural term representations during term attachment.

We propose to treat *term extraction for online catalog taxonomy enrichment* as a sequence labeling task. Tokens in the text are labeled with the BIOE schema where each tag represents the beginning, inside, outside, and ending of a chunk, respectively. Instead of collecting expensive human annotations for training [39], we propose to adopt distant supervision by using the existing terms V as self-supervised labels. Specifically, we find in Q and I the mentions of V and label those mentions as the desired terms to be extracted. For example, item "Golden State Fruit Pears to Compare Deluxe Gift" with associated taxonomy node "Pears" will be labeled as "O O O B O O O O" for model learning. This approach has several advantages. First, unlike unsupervised term extraction methods [13], we train a sequence labeling model to ensure that only the terms in the same domain as the existing terms V are extracted. Second, sequence labeling is more likely to extract terms of the desired type while filtering terms of undesired types (such as the brand or flavor of items) by inspecting the sentence context, which typical context-agnostic term extraction methods [16, 26] fail to do.

We train a BiLSTM-CRF model [20] to fulfill term extraction in Octet. Briefly speaking, each token in the text is represented by its word embedding, fed into a bidirectional LSTM layer for contextualized representation. Then, tag prediction is conducted by a conditional random field (CRF) layer operating on the representation of the current token and the previous tag. A similar model architecture can be found in Zheng et al. [39].

Note that distant supervision may inevitably introduce noises in the labels. Nevertheless, we show in Sec. 4.1 that Octet obtains

superior performance in term extraction (precision@100=0.91). In addition, Octet is likely to have low confidence in attaching incorrectly extracted terms, allowing further filtering with threshold setting during term attachment (Sec. 4.4.3).

3.3 Term Attachment Learning

Term attachment extends taxonomy ($\bar{V} = V \cup V', R$) with new edges R' between terms in V and V'. Following common practice [1, 33], we consider a taxonomy $\bar{T}$ as a Bayesian network where each node $v \in \bar{V}$ is a random variable. The joint probability distribution of nodes $\bar{V}$ can be then formulated as follows.

$$P(\bar{V} \mid \bar{T}, \Theta) = P(v_0) \prod_{v \in \bar{V} \setminus v_0} P(v \mid p(v), \Theta),$$

where v_0 denotes the root node, $p(v)$ denotes the direct hypernym (parent) of node v, and Θ denotes model parameters. Maximizing the likelihood with respect to the taxonomy structure $\bar{T}$ gives the optimal taxonomy $\bar{T}^*$. As we do not modify the structure of core taxonomy T, the formulation can be simplified as follows.

$$\begin{aligned}\bar{T}^* = \arg\max_{\bar{T}} P(\bar{V} \mid \bar{T}, \Theta) &= \arg\max_{\bar{T}} \prod_{v \in \bar{V} \setminus v_0} P(v \mid p(v), \Theta) \\ &= \arg\max_{\bar{T}} \prod_{v' \in V'} P(v' \mid p(v'), \Theta).\end{aligned}$$

The problem definition further ensures that $p(v') \in V$ always holds true and thus no inter-dependencies exist between the new terms $v' \in V'$. Therefore, we can naturally regard term attachment as a *multi-class classification problem* according to $P(v' \mid p(v'), \Theta)$ where each $p(v') = v \in V$ is a class.

One unique challenge for online catalog taxonomy enrichment is the lack of conventional corpora. For example, it is rare to find co-occurrences of multiple item types in a single query (<1% in Grocery domain on Amazon.com), let alone the "such as"-style patterns in Hearst-based methods [13]. Instead of using the limited text directly, we introduce signals from user behavior logs and the structure of the core taxonomy by modeling them in a graph (Sec. 3.3.1). The lexical semantics of the term pairs are further considered to better identify hypernymy relations (Sec. 3.3.2 & 3.3.3).

3.3.1 *Structural Representation.*

There is rich structural information for online catalog taxonomy enrichment which comes from two sources. First, the neighborhood (*e.g.*, parent and siblings) of a node $v \in V$ on T can serve as a meaningful supplement for the semantics of v. For example, one may not have enough knowledge of node v = "Makgeolli" (Korean rice wine); but if she perceives that "Sake" (Japanese rice wine) is v's sibling and "Wine" is v's parent, she would have more confidence in considering "Makgeolli" as one type of "Alcoholic Beverages". Second, there exist abundant user behavior data, providing even richer structural information than that offered by the core taxonomy T. Specifically, items I are associated with the existing taxonomy nodes V. New terms V' are related to items I and user queries Q since V' are extracted from the two sources. Furthermore, I and Q are also connected via clicks. Based on the observations above, we propose to learn a structural term pair representation to capture the structure in the core taxonomy and the query-item-taxonomy interactions as follows.

Graph Construction. We construct a graph $\mathcal{G}$ where the nodes consist of the existing terms $v \in V$ and new terms $v' \in V'$. There are two sets of edges: one set is the same as R in the core taxonomy T, which captures the ontological structure in T. The other set leverages the query-item-taxonomy interactions: for each new term $v' \in V'$, we find the user queries $Q_{v'}$ that mention v' and collect the clicked items $I_{Q_{v'}}$ in the queries $Q_{v'}$. Then, we find the taxonomy nodes $\{v_i\}$ that $I_{Q_{v'}}$ is associated with. Finally, we add an edge between each (v_i, v') pair. For instance, when determining the parent of a new term "Figs", we find that some queries mentioning "Figs" lead to clicked items associated with the taxonomy node "Fruits", evincing strong relations between the two terms.

Graph Embedding. We leverage graph neural networks (GNNs) to aggregate the neighborhood information in $\mathcal{G}$ when measuring the relationship between a term pair. Specifically, we take the rationales in relational graph convolutional networks (RGCNs) [12, 25]. Let $\mathcal{R}$ denote the set of relations, including (r_1) The neighbors of v on the core taxonomy T. The neighbors can be the (grand)parents or children of v and we compare different design choices in Sec. 4.4.2; (r_2) the interactions between $v \in V$ and $v' \in V'$ discovered in user behavior. We confine the interactions to be unidirectional (from v to v') since the terms $v \in V$ are already augmented with r_1 while there might be noise in the user behavior; and (r_3) the self-loop of node v. The self-loop of v ensures that the information of v itself is preserved and those isolated nodes without any connections can still be updated using its own representation.

Let $N(v, r)$ denote the neighbors of node v with relation r, and $\boldsymbol{h}_v^0$ the initial input representation of node v. The hidden representation of v at layer (hop) l is updated as follows: $\boldsymbol{h}_v^{l+1} = \text{ReLU}(\sum_{r \in \mathcal{R}} \sum_{i \in \{N(v,r)\}} \frac{1}{c_{v,r}} \boldsymbol{W}_r^l \boldsymbol{h}_i^l)$, where $\boldsymbol{W}_r^l$ is the matrix at layer l for linear projection of relation r and $c_{v,r}$ is a normalization constant. We take the final hidden representation of each node $\boldsymbol{h}_v^L$ (*denoted as* $\boldsymbol{g}_v$) as the graph embedding.

Relationship Measure. One straightforward way of utilizing the structural representation is to use the graph embeddings $\boldsymbol{g}_v$ and $\boldsymbol{g}_{v'}$ as the representation of term v and v'. Instead, we propose to measure the relationship between a term pair explicitly by cosine similarity and norm distance (more details in App. A). We denote the relationship measure between two embeddings as $s(\boldsymbol{v}, \boldsymbol{v'})$. One benefit of using $s(\boldsymbol{g}_v, \boldsymbol{g}_{v'})$ is that empirically we observe $s(\boldsymbol{g}_v, \boldsymbol{g}_{v'})$ alleviates overfitting compared with directly using $\boldsymbol{g}_v$ and $\boldsymbol{g}'_v$. Also, using $s(\boldsymbol{g}_v, \boldsymbol{g}_{v'})$ makes the final output size much smaller and reduces the number of parameters in the following layer significantly.

3.3.2 *Semantic Representation.*

We use the word embedding $\boldsymbol{w}_v$ of each term v to capture its semantics. For grouping nodes consisting of several item types (*e.g.*, "Fresh Flowers & Live Indoor Plants") and multi-word terms with qualifiers (*e.g.*, "Fresh Cut Bell Pepper"), we employ dependency parsing to find the noun chunks ("Fresh Flowers", "Live Indoor Plants", and "Fresh Cut Bell Pepper") and respective head words ("Flowers", "Plants", and "Pepper"). Intuitively, (v, v') tends to be related as long as one head word of v is similar to that of v'. We thus use the relationship measure $s(\boldsymbol{v}, \boldsymbol{v'})$ defined in Sec. 3.3.1 to measure the semantic relationship of the head words: $\boldsymbol{H}(v, v') = \max_{i,j} s(\boldsymbol{w}_{\text{head}^i(v)}, \boldsymbol{w}_{\text{head}^j(v')})$, where $\boldsymbol{w}_{\text{head}^i(v)}$ denotes the word embedding of the i-th head word in the term v. Finally,

the overall semantic representation $\boldsymbol{S}(v, v')$ is defined as $\boldsymbol{S}(v, v') = [\boldsymbol{H}(v, v'), \boldsymbol{w}_v, \boldsymbol{w}_{v'}]$, where "," denotes the concatenation operation.

3.3.3 Lexical Representation.

String-level measures prove to be very effective in hypernymy detection [1, 3, 7, 17]. For online catalog taxonomies, we also find many cases where lexical similarity provides strong signals for hypernymy identification. For example, "Black Tea" is a hyponym of "Tea" and "Fresh Packaged Green Peppers" is a sibling of "Fresh Packaged Orange Peppers". Therefore, we take the following lexical features [17] to measure the lexical relationship between term pairs: *Ends with*, *Contains*, *Suffix match*, *Longest common substring*, *Length difference*, and *Edit distance*. Values in each feature are binned by range and each bin is mapped to a randomly initialized embedding, which would be updated during model training. We denote the set of lexical features by $\mathcal{M}$ and compute lexical representation as the concatenation of the lexical features: $\boldsymbol{L}(v, v') = [\boldsymbol{L}_i(v, v')]_{i \in \mathcal{M}}$.

3.3.4 Heterogeneous Representation.

For each term pair (v, v'), we generate a heterogeneous term pair representation $\boldsymbol{R}(v, v')$ by combining the representations detailed above. $\boldsymbol{R}(v, v')$ captures several orthogonal aspects of the term pair relationship, which contribute to hypernymy detection in a complementary manner (Sec. 4.4.1). To summarize, the structural representation models the core taxonomy structure as well as underlying query-item-taxonomy interactions, whilst the semantic and lexical representations capture the distributional and surface information of the term pairs, respectively. We further calculate $s(\boldsymbol{v}, \boldsymbol{v'})$ between the graph embedding $\boldsymbol{g}_v$ ($\boldsymbol{g}'_v$) of one term and the word embedding $\boldsymbol{w}_{v'}$ ($\boldsymbol{w}_v$) of the other term in the term pair, which measures the relationship of the term pair in different forms and manifests improved performance. Formally, the heterogeneous term pair representation $\boldsymbol{R}(v, v')$ is defined as follows.

$$\boldsymbol{R}(v, v') = [s(\boldsymbol{g}_v, \boldsymbol{g}_{v'}), s(\boldsymbol{w}_v, \boldsymbol{g}_{v'}), s(\boldsymbol{g}_v, \boldsymbol{w}_{v'}), \boldsymbol{S}(v, v'), \boldsymbol{L}(v, v')].$$

3.3.5 Model Training and Inference.

Similar to prior works [17, 30], we feed $\boldsymbol{R}(v, v')$ into a two-layer feed-forward network and use the output after the sigmoid function as the probability of hypernym relationship between v and v'. To train the term attachment module, we permute all the term pairs (v_i, v_j) in V as training samples and utilize the existing hypernym pairs R on T for self-supervision – the pairs in R are regarded as positive samples and other pairs are negative. The heterogeneous term pair representation, including the structural representation, is learned in an end-to-end fashion. We use binary cross-entropy loss as the objective function due to the classification formulation. An alternative formulation is to treat term attachment as a hierarchical classification problem where the positive labels are all the ancestors of the term (instead of only its parent). We found, however, that hierarchical classification does not outperform standard classification and thus opt for the simpler formulation following [1, 33]. For inference, we choose $v_i \in V$ with the highest probability among all the permuted term pairs (v_i, v') as the predicted hypernym of v'.

4 EXPERIMENTS

In this section, we examine the effectiveness of Octet via both automatic and human evaluations. We first conduct experiments on term extraction (Sec. 4.1) and term attachment (Sec. 4.2) individually, and then perform an end-to-end open-world evaluation for Octet (Sec. 4.3). Finally, we carefully analyze the performance of Octet via framework ablation and case studies (Sec. 4.4).

4.1 Evaluation on Term Extraction

4.1.1 Evaluation Setup.

We take the *Grocery & Gourmet Food* taxonomy (2,163 terms) on Amazon.com as the major testbed for term extraction. We design three different evaluation setups with closed-world or open-world assumption as follows.

Closed-world Evaluation. We first conduct a closed-world evaluation that holds out a number of terms on the core taxonomy T as the virtual V'. In this way, we can ensure that the test set follows a similar distribution as the training set and the evaluation can be done automatically. Specifically, we match the terms on T with the titles of active items belonging to *Grocery & Gourmet Food* (2,995,345 in total).[2] 948 of the 2,163 terms are mentioned in at least one item title and 948,897 (item title, term) pairs are collected. We split the pairs by the 948 matched terms into training set V and test set V' with a ratio of 80% / 20% and evaluate term recall on the unseen set V'. Splitting the pairs by terms (instead of items) ensures that V and V' have no overlap, which is much more challenging than typical named entity recognition (NER) tasks but resembles the real-world use cases for online catalog taxonomy enrichment.

Open-world Evaluation. Open-world evaluation tests on new terms that do not currently exist in T, which is preferable since it evaluates term extraction methods in the same scenario as in production. The downside is that it requires manual annotations as there are no labels for the newly extracted terms. Therefore, we ask experts and workers on Amazon Mechanical Turk (MTurk) to verify the quality of the new terms. As we would like to take new terms with high confidence for term attachment, we ask our taxonomists to measure the precision of top-ranked terms that each method is most confident at. The terms that are already on the core taxonomy T are excluded and the top 100 terms of each compared method are carefully verified. To evaluate from the average customers' perspective, we sample 1,000 items and ask MTurk workers to extract the item types in the item titles. Different from expert evaluation, items are used as the unit for evaluation rather than terms. Precision and recall, weighted by the votes of the workers, are measured. More details of crowdsourcing are provided in App. C.

Baseline Methods. For the evaluation on term extraction, we compare with two approaches widely used for taxonomy construction, namely noun phrase (NP) chunking and AutoPhrase [26]. Pattern-based methods [15] and classification-based methods with simple n-gram and click count features [33] perform poorly in our preliminary experiments and are thus excluded. More details and discussions of the baselines are provided in App. B.

4.1.2 Evaluation Results.

For closed-world automatic evaluation, we calculate recall@K and show the comparison results in Fig. 3 *(Left)*. We observe that Octet consistently achieves the highest recall@K on the held-out test set. The overall recall of all compared methods, however, is relatively

[2] We also observed positive results on term extraction from user queries at Amazon.com but omit the results in the interest of space.

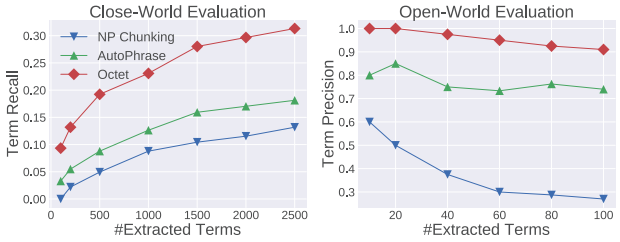

Figure 3: Closed-world automatic evaluation on term recall and open-world expert evaluation on term precision.

Table 1: Examples of top-ranked terms extracted by each approach. Valid terms for item types are marked in bold.

NP Chunking	AutoPhrase [26]	Octet
dark chocolate, whole bean, penzeys spices, **extra virgin olive oil**, net wt, a hint, no sugar	**almond butter**, honey roasted, **hot cocoa**, **tonic water**, **brown sugar**, **curry paste**, american flag	**coconut flour**, **ground cinnamon**, **red tea**, **ground ginger**, **green peas**, sweet leaf, **coconut syrup**

low. Nevertheless, we argue that the low recall is mainly due to the wording difference between the terms in the core taxonomy T and item titles. As we will demonstrate in the open-world evaluation below, many extracted terms are valid but not on T, which also confirms that T is very sparse and incomplete.

For open-world expert evaluation, we examine the terms each method is most confident at. In AutoPhrase [26], each extracted term is associated with a confidence score. In NP chunking and Octet, we use the frequency of extracted terms (instead of the raw frequency via string match) as their confidence score. As shown in Fig. 3 *(Right)*, Octet achieves very high precision@K (0.91 when K=100), which indicates that the newly extracted terms not found on the core taxonomy are of high quality according to human experts and can be readily used for term attachment. The compared methods, however, perform much worse on the top-ranked terms. In particular, the performance of NP chunking degenerates very quickly and only 27 of its top 100 extracted terms are valid.

We further show examples of the extracted terms in Table 1. As one can see, NP chunking extracts many terms that are either not item types (*e.g.*, "Penzeys Spices" is a company and "No Sugar" is an attribute) or less specific (*e.g.*, "Whole Bean" Coffee). AutoPhrase extracts some terms that are not of the desired type. For example, "Honey Roasted" and "American Flag" are indeed high-quality phrases that appear in the item titles but not valid item types. In contrast, Octet achieves very high precision on the top-ranked terms while ensuring that the extracted terms are of the same type as existing terms on the core taxonomy, which empirically verifies the superiority of formulating term extraction for online catalog taxonomy enrichment as sequence labeling.

Finally, we show the results of open-world crowdsourcing evaluation in Table 2. Octet again achieves much higher precision and F1 score than the baseline methods. AutoPhrase obtains higher recall as we found that it tends to extract multiple terms in each sample, whereas there is usually one item type. The recall of all the compared methods is still relatively low, which is possibly due to the conciseness and noise in the item titles (recall examples in Fig. 2) and leaves much space for future work.

Table 2: Performance comparison in the open-world crowdsourcing evaluation on 1,000 sampled items. Octet achieves significantly higher precision and best F1 score.

Method	Precision	Recall	F1
NP Chunking	12.3	20.4	15.4
AutoPhrase [26]	20.9	**41.3**	27.8
Octet	**87.5**	24.9	**38.8**

4.2 Evaluation on Term Attachment

4.2.1 Evaluation Setup.

For term attachment, we also conduct both closed-world and open-world evaluations, which involves ablating the core taxonomy T and attaching newly extracted terms, respectively. In contrast, most of prior studies [1, 11, 17] only perform closed-world evaluation.

Closed-world Evaluation. We take four taxonomies actively used in production as the datasets: *Grocery & Gourmet Food*, *Home & Kitchen*, and *Beauty & Personal Care* taxonomies at Amazon.com, and the *Business Categories* at Yelp.com.[3] *Amazon Grocery & Gourmet Food* is used for framework analysis unless otherwise stated. As the taxonomies are used in different domains and constructed according to their own business needs, they exhibit a wide spectrum of characteristics and almost have no overlap. Considering the real-world use cases where fine-grained terms are missing, we hold out the leaf nodes as the new terms V' to be attached as in [11]. We split V' into training, development, and test sets with a ratio of 64% / 16% / 20%, respectively. Detailed statistics of the datasets can be found in Table 3. Note that if we regard term attachment as a classification problem, each class would have very few training samples (*e.g.*, 1193 / 298 $\approx$ 4 in *Amazon Grocery & Gourmet Food*), which calls for a significant level of model generalizability.

Table 3: Taxonomy statistics for term attachment.

Taxonomy	$\lvert V\rvert$	$\lvert V'\rvert$	$\lvert V'_{\text{Train}}\rvert$	$\lvert V'_{\text{Dev}}\rvert$	$\lvert V'_{\text{Test}}\rvert$
Amazon Grocery & Gourmet Food	298	1,865	1,193	299	373
Amazon Home & Kitchen	338	1,410	902	226	282
Amazon Beauty & Personal Care	109	454	290	73	91
Yelp Business Categories	84	920	588	148	184

Open-world Evaluation. We conduct an open-world evaluation for term attachment on *Amazon Grocery & Gourmet Food*. Specifically, we ask the taxonomists to identify valid new terms among those discovered in the term extraction stage with frequency greater than 20. 106 terms are thus labeled as valid. We then ask the taxonomists to attach the 106 terms manually to the core taxonomy as ground truth and evaluate systems on these new terms with the same criteria as in the closed-world evaluation.

Evaluation Metrics. We use Edge-F1 and Ancestor-F1 [17] to measure the performance of term attachment. **Edge-F1** compares the predicted edge (v, v') with the gold edge $(p(v'), v')$, *i.e.*, whether $v = p(v')$. We use P_{Edge}, R_{Edge}, and $F1_{\text{Edge}}$ to denote the precision,

[3]The taxonomies are available online. We use five-month user behavior logs on Amazon.com for structural representation and do not leverage user behavior on Yelp.com due to accessibility. See App. A for more details on data availability.

Table 4: Comparison results of closed-world evaluation on *Amazon Grocery & Gourmet Food*.

Method	Dev		Test	
	Edge-F1	Ancestor-F1	Edge-F1	Ancestor-F1
Random [1, 11]	0.3	30.3	0.5	31.2
Root	0.3	41.1	0	41.8
I2T	10.0	50.5	9.9	50.7
Substr [3]	8.4	49.9	10.7	52.9
HiDir [33]	42.5	66.8	40.5	66.4
MSejrKu [24]	58.9	80.6	53.1	76.7
Octet	**64.9**	**85.2**	**62.5**	**84.2**

Table 5: Closed-world evaluation in different domains. Only competitive baseline methods that perform well on *Amazon Grocery & Gourmet Food* are listed.

Dataset	Method	Edge-F1	Ancestor-F1
Amazon Home & Kitchen	HiDir [33]	46.5	76.9
	MSejrKu [24]	46.0	74.1
	Octet	**54.4**	**78.5**
Amazon Beauty & Personal Care	HiDir [33]	34.1	71.8
	MSejrKu [24]	49.5	75.0
	Octet	**50.6**	**77.1**
Yelp Business Categories	HiDir [33]	19.6	35.8
	MSejrKu [24]	29.9	35.7
	Octet	**32.6**	**43.5**

recall, and F1-score, respectively. In particular, $P_{\text{Edge}} = R_{\text{Edge}} = F1_{\text{Edge}}$ if the number of predicted edges is the same as gold edges, *i.e.*, when all the terms in V' are attached. **Ancestor-F1** is more relaxed than Edge-F1 as it compares all the term pairs (v_{sys}, v') with (v_{gold}, v'), where v_{sys} represents the terms along the system predicted path (ancestors) to the root node, and v_{gold} denotes those terms on the gold path. Similarly, we denote the ancesor-based metrics as $P_{\text{Ancestor}} = \frac{|v_{\text{sys}} \wedge v_{\text{gold}}|}{|v_{\text{sys}}|}$ and $R_{\text{Ancestor}} = \frac{|v_{\text{sys}} \wedge v_{\text{gold}}|}{|v_{\text{gold}}|}$.

Baseline Methods. As discussed earlier, there are few existing methods for taxonomy enrichment. MSejrKu [24] is the winning method in the SemEval-2016 Task 14 [11]. HiDir [33] is the state-of-the-art method for online catalog taxonomy enrichment. In addition, we compare with the substring method Substr [3], and I2T that finds one's hypernym by examining where its related items are assigned to. Two naïve baselines are also tested to better understand the difficulty of the task following convention [1, 11], where *Random* attaches $v' \in V'$ randomly to T and *Root* attaches every term to the root node of T. More details of the baselines are in App. B.

4.2.2 Evaluation Results.

We first evaluate different methods under the closed-world assumption (Tables 4 & 5). We observe that the Edge-F1 of two naïve baselines is very low since there are hundreds of $v \in V$ as candidates. The performance of I2T is similar to Substr but still far from satisfactory, implying that there might be noise in the matching between V' and I, and the associations between I and V. HiDir [33] and MSejrKu [24] achieve better performance than other baselines, especially in Edge-F1, while Octet outperforms all the compared methods by a large margin on both development and test sets across all of the four domains.

Table 6: Open-world expert evaluation. Note that the seemingly low *absolute* performance is comparable to results on other datasets [1, 17] due to the difficulty of the task.

Method	Edge-F1	Ancestor-F1
HiDir [33]	28.3	59.2
MSejrKu [24]	29.3	61.2
Octet	**30.2**	**67.5**

For the open-world evaluation, we compare Octet with the best performing baselines MSejrKu [24] and HiDir [33] on the expert-labeled data (Table 6). Perhaps unsurprisingly, the performance of each method is lower than that under the closed-world evaluation, since the distributions of the existing terms on T and the newly extracted terms are largely different (shown in Sec. 4.1). Octet again achieves the best performance thanks to its better generalizability.

4.3 End-to-End Evaluation

Besides the individual evaluations on term extraction and term attachment, we perform a novel *end-to-end open-world evaluation* that helps us better understand the quality of the enriched taxonomy by examining errors throughout the entire framework: whether (A) the extracted term is invalid or (B) the term is valid but the attachment is inaccurate. To our knowledge, such an end-to-end evaluation has not been conducted in prior studies.

We evaluate Octet on *Amazon Grocery & Gourmet Food* using Amazon MTurk. The details of crowdsourcing can be found in App. C. In total, Octet extracts and attaches 2,192 new terms from the item titles described in Sec. 4.1.1, doubling the size of the existing taxonomy (2,163 terms). As listed in Table 7, only 6.5% of extracted terms are considered invalid by average customers (MTurk workers). The top-1 edge precision and ancestor precision are relatively low, but they are comparable to the state-of-the-art performance on similar datasets with *clean* term vocabulary [17]. The neighbor precision, which considers the siblings on the predicted path as correct, is very high (88.1). One can further improve the precision by filtering low-confidence terms or allowing top-k prediction (Sec. 4.4.3).

Table 7: Open-world end-to-end evaluation for Octet.

Error A%	Error B%	Edge Prec	Ancestor Prec	Neighbor Prec
6.5	11.9	22.1	40.9	88.1

4.4 Framework Analysis

4.4.1 Feature Ablation. We analyze the contribution of each representation for the term pair (Table 8). As one can see, only using word embedding (W) does not suffice to identify hypernymy relations while adding the semantics of head words (H) explicitly boosts the performance significantly. Lexical (L) features are very effective and combining lexical representation with semantic representation (L + W + H) brings 17.2 absolute improvement in Edge-F1. The structural information is very useful in that even if we use word embedding (W) as the input of the structural representation (G), the Edge-F1 improves by 4.8x and the Ancestor-F1 improves by 59.3% upon W. The full model that incorporates various representations performs the best, indicating that they capture different aspects of the term pair relationship and are complementary to each other.

Table 8: Ablation study of word embedding (W), head word semantics (H), lexical representation (L), and structural graph-based representation (G).

Representation	Dev		Test	
	Edge-F1	Ancestor-F1	Edge-F1	Ancestor-F1
W	12.0	50.4	11.0	49.7
H	30.8	62.9	29.2	64.4
W + H	41.8	68.8	39.7	67.8
L	48.2	74.7	42.6	70.3
L + W	57.9	79.7	52.6	76.9
G	57.5	80.3	50.9	78.6
L + W + G	63.6	82.6	56.0	79.9
L + W + H	62.5	83.6	59.8	81.6
L + W + H + G	**64.9**	**85.2**	**62.5**	**84.2**

Table 9: Comparison of various design choices in the structural representation. C and P denote Child and Parent. r_1, r_2, r_3 denote structure in the core taxonomy, query-item-taxonomy interactions, and self-loop, respectively.

	Design Choice	Edge-F1	Ancestor-F1
L	One-hop neighborhood	50.4	75.9
	Two-hop neighborhood	**60.1**	**83.0**
$N(v, r_1)$	C->P	**60.1**	83.0
	C<->P	59.8	83.2
	P->C	59.3	**83.6**
$\mathcal{R}$	$\{r_1, r_3\}$	60.1	83.0
	$\{r_1, r_2, r_3\}$	**62.5**	**84.2**

4.4.2 Graph Ablation. We analyze different variants regarding the design choices in the structural representation. As shown in Table 9, considering multi-hop relations (*e.g.*, grandparents and siblings) is better than only considering immediate family (*i.e.*, parent and children). The directionality of edges does not have a huge effect on the model performance, although the information of the ancestors tends to be more benefitial for Ancestor-F1 and descendants for Edge-F1. Adding the query-item-taxonomy interactions in the user behavior (r_2) in addition to the structure in the core taxonomy further improves model performance, showing the benefits of leveraging user behavior for term attachment.

4.4.3 Performance Trade-Off. Precision recall trade-off answers the practical question in production "how many terms can be attached if a specific precision of term attachment is required", by filtering predictions with $\max_{v \in V} P(v' \mid v, \Theta) < c$, where $c \in [0, 1]$ is a thresholding constant. As depicted in Fig 4 *(Left)*, more than 15% terms can be recalled and attached perfectly. Over 60% terms can be recalled when an edge precision of 80% is achieved.

We also analyze the performance changes if we relax term attachment to top-k predictions rather than top-1 prediction (as measured in Edge-F1). We observe that more than 95% gold hypernyms of the query terms are in the top-50 predictions (Fig 4 *(Right)*), which indicates that Octet generally ranks the gold hypernyms higher than other candidates even if its top-1 prediction is incorrect. Note that the results in all the previous experiments are regarding term recall equal to 1 (all extracted terms are attached) and Hit@1 (whether the top-1 prediction is correct).

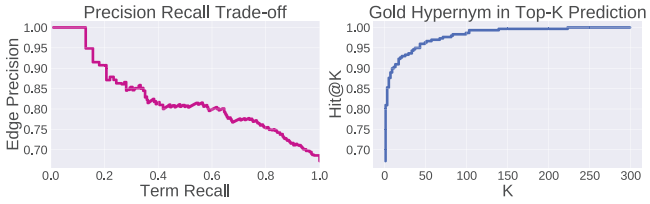

Figure 4: The precision recall trade-off *(Left)* and performance of term attachment in Hit@K *(Right)*.

4.4.4 Case Studies. We inspect correct and incorrect term attachments to better understand model behavior and the contribution of each type of representation. As shown in Table 10, Octet successfully predicts "Fresh Cut Flowers" as the hypernym of "Fresh Cut Carnations", where lexical representation possibly helps with the matching of "Fresh Cut" and semantic representation identifies "Carnations" is a type of "Flowers". When lexical clues are lacking, Octet can still use semantic representation to recognize that "Tilapia" is a type of "Fresh Fish". Without structural representation, Octet detects that "Bock Beer" is closely related to "Beer" and "Ales". By adding the structural representation Octet could narrow down its prediction to the more specific type "Lager & Pilsner Beers". For the wrongly attached terms, Octet is unconfident in distinguishing "Russet Potatoes" from "Fingerling & Baby Potatoes", which is undoubtfully a harder case and requires deeper semantic understanding. Octet also detects a potential error in the core taxonomy itself as we found the siblings of "Pinto Beans" all start with "Dried", which might have confused Octet during training.

Table 10: Case studies of term attachment. Correct and incorrect cases are marked in green and red, respectively.

Query Term	Gold Hypernym	Top-3 Predictions
fresh cut carnations	fresh cut flowers	fresh cut flowers, fresh cut root vegetables, fresh cut & packaged fruits
tilapia	fresh fish	fresh fish, liquor & spirits, fresh shellfish
bock beers	lager & pilsner beers	**W/O structural representation:** ales, beer, tea beverages **Full Model:** lager & pilsner beers, porter & stout beers, tea beverages
fresh russet potatoes	fresh potatoes & yams	fresh fingerlings & baby potatoes, fresh root vegetables, fresh herbs
pinto beans	dried beans	canned beans, fresh peas & beans, single herbs & spices

5 RELATED WORK

Taxonomy Construction. [29] proposed a graph-based approach to attach new concepts to Wikipedia categories. [10] enriched WordNet by finding terms from Wiktionary and attaching them via rule-based patterns. A concurrent study [27] also leverages GNNs for taxonomy enrichment. However, the features used in prior methods [10, 15, 28, 29] are designed either for specific taxonomies or for text-rich domains whereas Octet is robust to short texts and generally applicable to various online domains. For online catalog taxonomies, [33] extracted terms from search queries and formulated taxonomy enrichment as a hierarchical Dirichlet process, where each node is represented by a uni-gram language model of its associated items. [15] used patterns to extract new concepts in the queries and online news, and built a topic-concept-instance

taxonomy with human-labeled training set. In contrast, Octet is self-supervised and utilizes heterogeneous sources of information.

Term Extraction. [13] used Hearst patterns [8] to extract new terms from the web pages. [28, 37] used AutoPhrase [26] to extract keyphrases in general-purpose corpora. These methods, however, are inapplicable or ineffective for short texts like item titles. On the other hand, many prior studies [1, 3, 11, 17, 21] made a somewhat unrealistic assumption that one clean vocabulary is given as input and focused primarily on hypernymy detection. Another plausible alternative is to treat entire user queries as terms rather than perform term extraction [33], which results in a low recall. In contrast, Octet employs a sequence labeling model designed for term extraction from online domains with self-supervision.

Hypernymy Detection. Pattern-based hypernymy detection methods [8, 13, 19, 21, 31] consider the lexico-syntactic patterns between the co-occurrences of term pairs. They achieve high precision in text-rich domains but suffer from low recall and also generate false positives in domains like e-commerce. Distributional methods [6, 23, 30, 32, 36] utilize the contexts of each term for semantic representation learning. Unsupervised measures such as symmetric similarity [14] and distributional inclusion hypothesis [4, 34] are also proposed, but not directly applicable to online catalog taxonomy enrichment as there are *grouping nodes* in the online catalog taxonomies like "Dairy, Cheese & Eggs" and "Flowers & Gifts" curated by taxonomists for business purposes, which might not exist in any form of text corpora. In Octet, heterogeneous sources of signals are leveraged to tackle the lack of text corpora and grouping nodes are captured by head word semantics.

6 CONCLUSION

In this paper, we present a self-supervised end-to-end framework for online catalog taxonomy enrichment that considers heterogeneous sources of representation and does not involve additional human effort other than the existing core taxonomies to be enriched. We conduct extensive experiments on real-world taxonomies used in production and show that the proposed framework consistently outperforms state-of-the-art methods by a large margin under both automatic and human evaluations. In the future, we will explore the feasibility of joint learning of online catalog taxonomy enrichment and downstream applications such as recommendation and search.

ACKNOWLEDGMENT

We thank Jiaming Shen, Jun Ma, Haw-Shiuan Chang, Giannis Karamanolakis, Colin Lockard, Qi Zhu, and Johanna Umana for the valuable discussions and feedback.

REFERENCES

[1] Mohit Bansal, David Burkett, Gerard De Melo, and Dan Klein. 2014. Structured Learning for Taxonomy Induction with Belief Propagation.. In *ACL*.
[2] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *TACL* (2017).
[3] Georgeta Bordea, Els Lefever, and Paul Buitelaar. 2016. Semeval-2016 task 13: Taxonomy extraction evaluation (texeval-2). In *SemEval-2016*. ACL.
[4] Haw-Shiuan Chang, ZiYun Wang, Luke Vilnis, and Andrew McCallum. 2017. Unsupervised Hypernym Detection by Distributional Inclusion Vector Embedding. *arXiv preprint arXiv:1710.00880* (2017).
[5] Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. 2016. Lifted Rule Injection for Relation Embeddings. In *EMNLP*.
[6] Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning semantic hierarchies via word embeddings. In *ACL*.
[7] Amit Gupta, Rémi Lebret, Hamza Harkous, and Karl Aberer. 2017. Taxonomy Induction using Hypernym Subsequences. In *CIKM*. ACM, 1329–1338.
[8] Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *ACL*.
[9] Jin Huang, Zhaochun Ren, Wayne Xin Zhao, Gaole He, Ji-Rong Wen, and Daxiang Dong. 2019. Taxonomy-aware multi-hop reasoning networks for sequential recommendation. In *WSDM*. ACM, 573–581.
[10] David Jurgens and Mohammad Taher Pilehvar. 2015. Reserating the awesometastic: An automatic extension of the WordNet taxonomy for novel terms. In *NAACL*.
[11] David Jurgens and Mohammad Taher Pilehvar. 2016. Semeval-2016 task 14: Semantic taxonomy enrichment. In *SemEval-2016*.
[12] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
[13] Zornitsa Kozareva and Eduard Hovy. 2010. A semi-supervised method to learn and construct taxonomies using the web. In *EMNLP*. ACL, 1110–1118.
[14] Dekang Lin et al. 1998. An information-theoretic definition of similarity.. In *Icml*, Vol. 98. Citeseer, 296–304.
[15] Bang Liu, Weidong Guo, Di Niu, Chaoyue Wang, Shunnan Xu, Jinghong Lin, Kunfeng Lai, and Yu Xu. 2019. A User-Centered Concept Mining System for Query and Document Understanding at Tencent. In *KDD*.
[16] Jialu Liu, Jingbo Shang, Chi Wang, Xiang Ren, and Jiawei Han. 2015. Mining quality phrases from massive text corpora. In *SIGMOD*. ACM, 1729–1744.
[17] Yuning Mao, Xiang Ren, Jiaming Shen, Xiaotao Gu, and Jiawei Han. 2018. End-to-End Reinforcement Learning for Automatic Taxonomy Induction. In *ACL*.
[18] Yuning Mao, Jingjing Tian, Jiawei Han, and Xiang Ren. 2019. Hierarchical Text Classification with Reinforced Label Assignment. In *EMNLP*.
[19] Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. PATTY: a taxonomy of relational patterns with semantic types. In *EMNLP*. 1135–1145.
[20] Rrubaa Panchendrarajan and Aravindh Amaresan. 2018. Bidirectional LSTM-CRF for Named Entity Recognition. In *ACL*.
[21] Alexander Panchenko, Stefano Faralli, Eugen Ruppert, Steffen Remus, Hubert Naets, Cedrick Fairon, Simone Paolo Ponzetto, and Chris Biemann. 2016. TAXI at SemEval-2016 Task 13: a Taxonomy Induction Method based on Lexico-Syntactic Patterns, Substrings and Focused Crawling. In *SemEval*. San Diego, CA, USA.
[22] Xiang Ren, Ahmed El-Kishky, Chi Wang, Fangbo Tao, Clare R Voss, and Jiawei Han. 2015. Clustype: Effective entity recognition and typing by relation phrase-based clustering. In *KDD*. ACM, 995–1004.
[23] Laura Rimell. 2014. Distributional lexical entailment by topic coherence. In *EACL*.
[24] Michael Schlichtkrull and Héctor Martínez Alonso. 2016. Msejrku at semeval-2016 task 14: Taxonomy enrichment by evidence ranking. In *SemEval-2016*.
[25] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*. Springer, 593–607.
[26] Jingbo Shang, Jialu Liu, Meng Jiang, Xiang Ren, Clare R Voss, and Jiawei Han. 2018. Automated phrase mining from massive text corpora. *TKDE* (2018).
[27] Jiaming Shen, Zhihong Shen, Chenyan Xiong, Chi Wang, Kuansan Wang, and Jiawei Han. 2020. TaxoExpan: Self-supervised Taxonomy Expansion with Position-Enhanced Graph Neural Network. In *WWW*. 486–497.
[28] Jiaming Shen, Zeqiu Wu, Dongming Lei, Chao Zhang, Xiang Ren, Michelle T Vanni, Brian M Sadler, and Jiawei Han. 2018. HiExpan: Task-guided taxonomy construction by hierarchical tree expansion. In *KDD*. ACM, 2180–2189.
[29] Wei Shen, Jianyong Wang, Ping Luo, and Min Wang. 2012. A graph-based approach for ontology population with named entities. In *CIKM*. ACM, 345–354.
[30] Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving Hypernymy Detection with an Integrated Path-based and Distributional Method. In *ACL*.
[31] Rion Snow, Daniel Jurafsky, and Andrew Y Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In *NIPS*. 1297–1304.
[32] Luu A Tuan, Yi Tay, Siu C Hui, and See K Ng. 2016. Learning Term Embeddings for Taxonomic Relation Identification Using Dynamic Weighting Neural Network. In *EMNLP*.
[33] Jingjing Wang, Changsung Kang, Yi Chang, and Jiawei Han. 2014. A hierarchical dirichlet model for taxonomy expansion for search engines. In *WWW*. 961–970.
[34] Julie Weeds, David Weir, and Diana McCarthy. 2004. Characterising measures of lexical distributional similarity. In *COLING*.
[35] Shuo Yang, Lei Zou, Zhongyuan Wang, Jun Yan, and Ji-Rong Wen. 2017. Efficiently Answering Technical Questions - A Knowledge Graph Approach. In *AAAI*.
[36] Zheng Yu, Haixun Wang, Xuemin Lin, and Min Wang. 2015. Learning Term Embeddings for Hypernymy Identification.. In *IJCAI*. 1390–1397.
[37] Chao Zhang, Fangbo Tao, Xiusi Chen, Jiaming Shen, Meng Jiang, Brian Sadler, Michelle Vanni, and Jiawei Han. 2018. Taxogen: Unsupervised topic taxonomy construction by adaptive term embedding and clustering. In *KDD*.
[38] Hao Zhang, Zhiting Hu, Yuntian Deng, Mrinmaya Sachan, Zhicheng Yan, and Eric Xing. 2016. Learning Concept Taxonomies from Multi-modal Data. In *ACL*.
[39] Guineng Zheng, Subhabrata Mukherjee, Xin Luna Dong, and Feifei Li. 2018. Opentag: Open attribute value extraction from product profiles. In *KDD*.

A REPRODUCIBILITY

A.1 Implementation Details

We use fastText [2] as the word embedding $\boldsymbol{w}_v$ in order to capture sub-word information. $\boldsymbol{w}_v$ is fixed to avoid semantic shift for the inference of unseen terms V'. All the terms v are lowercased and concatenated with underscores if there are multiple words.

For structural representation, we use the Deep Graph Library[4] for the implementation of GNN-based models. The number of layers L is set to 2 (two-hop neighbors are considered) and the normalization constant $c_{v,r}$ is 1. We sample $N = 5$ neighbors instead of using all the neighbors $N(v)$ for information aggregation. Node embedding $\boldsymbol{h}_v^l$ is of size 300, also initialized by the fastText embedding. $\boldsymbol{W}^l$ is of size 300×300. All of the lexical string-level features $\boldsymbol{L}_i(v, v')$ are randomly initialized and have an embedding size of 10. $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ are of size 1×100 and $100 \times |\boldsymbol{R}(v, v')|$, respectively.

For *relationship measure* involving the output of the GNNs, $s(\boldsymbol{v}, \boldsymbol{v}')$ measures the L^1 Norm, L^2 Norm, and cosine similarity between the embeddings of a term pair. For that between head words, $s(\boldsymbol{v}, \boldsymbol{v}')$ measures the cosine similarity of their corresponding word embeddings. We use Adam as the optimizer with initial learning rate 1e-4 and choose the best performing model according to the development set.

A.2 Data Availability

The taxonomies used in the experiments are available online. The taxonomies on Amazon.com can be obtained by scraping public webpages (refer to Fig. 1) or registering as a vendor. The Yelp taxonomy is accessible at www.yelp.com/developers/documentation/v3/all_category_list. The user behavior logs are mainly used for the GNN-based structural representation learning. We use five-month user behavior logs in the target domain on Amazon.com and do not leverage user behavior on Yelp.com due to accessibility. All of the experiments and framework analysis without the structural representation can be reproduced.

While each taxonomy used in the experiments is seemingly "small", it is *real* and quite big (2K+ nodes) for a single domain (Grocery). Octet is easily applicable to other domains (Home, Electronics, ...), which also contain thousands of categories; they collectively form a taxonomy of 30K+ nodes. There are also datasets with similar or smaller size (*e.g.*, 187 to 1386 nodes in [38], and 50 to 100 nodes in [1, 17]). Our setup is more general than SemEval-2016 Task 14 [11] and can be simplified to it if we ignore user behavior.

B BASELINE

B.1 Baseline in Term Extraction

NP chunking is one of the popular choices for general-purpose entity extraction [22]. We conduct NP chunking via spaCy[5], which performs dependency parsing on the sentences and finds noun phrases in them. No task-specific input (*e.g.*, a list of positive terms) is required or supported for the training of NP chunking. Post-processing, including removing the terms containing punctuation or digits, is used to filter clearly invalid terms from the results.

AutoPhrase [26] is the state-of-the-art phrase mining method widely used in previous taxonomy construction approaches [28, 37]. We replace the built-in Wikipedia phrase list that AutoPhrase uses for distant supervision with the terms V on the core taxonomy T, as we find that it performs better than appending V to the Wikipedia phrase list. Note that AutoPhrase uses exactly the same resources as Octet for distant supervision. In terms of methodology, AutoPhrase [26] focuses more on the corpus-level statistics and performs phrasal segmentation instead of sequence labeling.

Pattern-based methods [15] are ineffective in our scenario due to the lack of clear patterns in the item profiles and user queries. The classification-based method in [33] only works for term extraction from queries and also performs poorly, possibly because Wang et al. [33] treats the whole query as one term, which results in low recall.

B.2 Baseline in Term Attachment

HiDir [33] conducts hypernymy detection with the assumption that the representation of a taxonomy node is the same as its parent node, where one node is represented by its associated items.

Since we do not have *description sentences* as in SemEval-2016 Task 14 [11], most of MSejrKu's features are inapplicable and we thus replace its features with those used in Octet except for the structural representation. we found that such changes improve MSejrKu's performance.

Substr [3] is the sub-string baseline used in SemEval-2016 Task 13 [3]. It is shown that Substr, which regards A as B's hypernym if A is a substring of B (*e.g.*, "Coffee" and "Ground Coffee"), outperforms most of the systems in the SemEval-2016 competition on automatic taxonomy construction [3].

I2T matches v' with item titles and finds the taxonomy nodes these items $I_{v'}$ are associated with. The final prediction of I2T is made by majority voting of the associated nodes where the items $I_{v'}$ are assigned to.

C CROWDSOURCING

Crowdsourcing is a reasonable choice in our scenario because our terms are common words (*e.g.*, "coffee" and "black coffee") without complicated relations that require domain expertise (*e.g.*, "sennenhunde" and "entlebucher" in WordNet).

C.1 Crowdsourcing for Open-world Term Extraction Evaluation

For crowdsourcing evaluation in term extraction, each item is assigned to 5 workers and only item types labeled by at least 2 workers are considered valid. Links to the corresponding pages on Amazon.com are also provided to the workers.

C.2 Crowdsourcing for End-to-End Evaluation

We use crowdsourcing on MTurk for the end-to-end evaluation since expert evaluation is difficult to scale. We assign each term to 4 MTurk workers for evaluation. One critical problem of using crowdsourcing for end-to-end evaluation is that we can not ask MTurk workers to find the hypernym of each new term directly, as there are thousands of terms V on the core taxonomy T while the workers are unfamiliar with the taxonomy structure. Alternatively, we ask the workers to verify the precision of Octet: we provide

[4] https://www.dgl.ai/
[5] https://spacy.io/

the predicted hypernym of Octet and the ancestors along the predicted path to the root as the candidates. We also include in the candidates one sibling term at each level along the predicted path. The workers are required to select the hypernym(s) of the query term v' among the provided candidates. In this way, we can estimate how accurate the term attachment is from the average customers' perspective. Two other options, "*The query term is not a valid type*" and "*None of above are hypernyms of the query term*", are also provided, which corresponds to error types (A) and (B), respectively (refer to Sec. 4.3).