TaxoClass: Hierarchical Multi-Label Text Classification Using Only Class Names

Jiaming Shen[◇], Wenda Qiu[◇], Yu Meng[◇], Jingbo Shang[▽], Xiang Ren[△], Jiawei Han[◇]
[◇]University of Illinois at Urbana-Champaign, IL, USA, [▽]University of California, San Diego, CA, USA
[△]University of Southern California, CA, USA
[◇]{js2, qiuwenda, yumeng5, hanj}@illinois.edu [▽]jshang@ucsd.edu [△]xiangren@ucs.edu

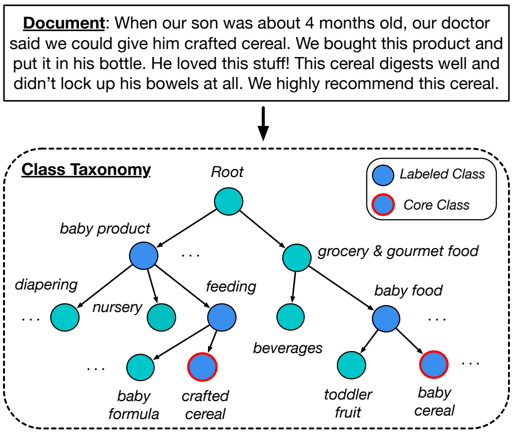
Abstract

Hierarchical multi-label text classification (HMTC) aims to tag each document with a set of classes from a class hierarchy. Most existing HMTC methods train classifiers using massive human-labeled documents, which are often too costly to obtain in real-world applications. In this paper, we explore to conduct HMTC based on only class surface names as supervision signals. We observe that to perform HMTC, human experts typically first pinpoint a few most essential classes for the document as its "core classes", and then check core classes' ancestor classes to ensure the coverage. To mimic human experts, we propose a novel HMTC framework, named TaxoClass. Specifically, TaxoClass (1) calculates document-class similarities using a textual entailment model, (2) identifies a document's core classes and utilizes confident core classes to train a taxonomy-enhanced classifier, and (3) generalizes the classifier via multi-label self-training. Our experiments on two challenging datasets show TaxoClass can achieve around 0.71 Example-F1 using only class names, outperforming the best previous method by 25%.

1 Introduction

Hierarchical multi-label text classification (HMTC) aims to assign each text document to a set of relevant classes from a class taxonomy. As a fundamental task in NLP, HMTC has many applications such as product categorization (Goumy and Mejri, 2018), semantic indexing (Li et al., 2019), and fine-grained entity typing (Xu and Barbosa, 2018).

Most existing methods address HMTC in a supervised fashion — they first ask humans to provide many labeled documents and then train a text classifier for prediction. Many classifiers have been developed with different deep learning architectures such as CNN (Kim, 2014), RNN (You et al., 2019), Attention Network (Huang et al., 2019), and achieved decent performance when trained on massive human-labeled documents. Despite such a success, people find that applying these methods to many real-world scenarios remains challenging as the human labeling process is often too time-consuming and expensive.



Document: When our son was about 4 months old, our doctor said we could give him crafted cereal. We bought this product and put it in his bottle. He loved this stuff! This cereal digests well and didn't lock up his bowels at all. We highly recommend this cereal.

Figure 1: An exemplar document tagged with five classes. Here, if we are able to pinpoint this document's most essential classes, *crafted cereal* and *baby cereal*, as core classes, we can check their ancestor classes in the taxonomy and recover all the true classes.

Recently, more studies have been developed to address text classification using smaller amount of labeled data. First, several semi-supervised methods (Gururangan et al., 2019; Berthelot et al., 2019) propose to use abundant unlabeled documents to assist model training on labeled dataset. Although mitigating the human annotation burden, these methods still require a labeled dataset that covers all classes, which could be too expensive to obtain when we have a large number of classes in HMTC. Second, some weakly-supervised models exploit class indicative keywords (Meng et al., 2018; Zeng et al., 2019; Mekala and Shang, 2020) or class surface names (Meng et al., 2020; Wang et al., 2020) to derive pseudo-labeled data for model training. Nevertheless, these models all assume each document has only one class and all class surface names (or class indicative keywords) must appear in the corpus, which are too restrictive for HMTC.

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 4239–4249
June 6–11, 2021. ©2021 Association for Computational Linguistics

In this paper, we study the problem of *weakly-supervised* hierarchical multi-label text classification where only class surface names, a class taxonomy, and an unlabeled corpus are available for model training. This setting is closer to how humans resolve the HMTC problem — we perform classification by understanding each class from its surface name rather than learning from labeled documents. We observe that when asked to assign multiple classes to a document, humans will first pinpoint most essential "core classes" and then check whether their ancestor classes in the taxonomy should also be tagged. Taking the document in Fig. 1 as an example, humans can quickly identify this review text is clearly about "*baby cereal*" and "*crafted cereal*", which are the core classes. After assigning these two most essential classes to the document, people continue to check the core classes' ancestor classes and find "*feeding*" as well as "*baby food*" should be tagged.

Motivated by the above human labeling process, we propose **TaxoClass**, a *weakly-supervised* HMTC framework including four major steps. First, we calculate the document-class similarity using a pre-trained textual entailment model (Yin et al., 2019). Second, we identify each document's core classes by (1) selecting candidate core classes that are most similar to the document at each level in a top-down fashion, and (2) choosing ⟨document, candidate core class⟩ pairs that are salient across the whole unlabeled corpus. Third, we derive training data from document core classes and use them to train a text classifier. This classifier includes a document encoder based on pre-trained BERT (Devlin et al., 2019), a class encoder capturing class taxonomy structure, and a text matching network computing the probability of a document being tagged with each class. Finally, we generalize this text classifier using multi-label self-training on all unlabeled documents.

Contributions. To summarize, our major contributions are as follows: (1) We propose a weakly-supervised framework TaxoClass that only requires class surface names to perform hierarchical multi-label text classification. To the best of our knowledge, TaxoClass is the first weakly-supervised HMTC method. (2) We develop an unsupervised method to identify document core classes based on which a text classifier can be learned. (3) We conduct extensive experiments to verify the effectiveness of TaxoClass on two real-world datasets.

2 Problem Formulation

In this section, we introduce the notations and present our task definition.

Notations. A corpus $\mathcal{D} = \{D_1, \dots, D_N\}$ is a text collection where each document $D_i \in \mathcal{D}$ is a sequence of words. A class taxonomy $\mathcal{T} = (\mathcal{C}, \mathcal{R})$ is a directed acyclic graph where each node represents a class c_j and each directed edge $\langle c_m, c_n \rangle \in \mathcal{R}$ indicates that parent class c_m is more general than the child class c_n. In this work, we assume each class c_j has a surface name s_j (either a word or a phrase) that serves as the weak supervision signal.

Task Definition. Given an unlabeled corpus $\mathcal{D}$, a class hierarchy $\mathcal{T} = (\mathcal{C}, \mathcal{R})$, and class surface names $\mathcal{S} = \{s_j\}_{j=1}^{|\mathcal{C}|}$, our task is to learn a text classifier $f(\cdot)$ that maps a new document D_{new} to its target $\mathbf{y} = [y_1, \dots, y_{|\mathcal{C}|}] \in \mathcal{Y} = \{0, 1\}^{|\mathcal{C}|}$ where y_j equals to 1 if this document is categorized with class c_j and 0 otherwise.

Discussion. When the number of classes $|\mathcal{C}|$ is large (as it is in many HMTC applications), we can no longer assume all class surface names in $\mathcal{S}$ will explicitly appear in the given corpus $\mathcal{D}$ as done in most previous studies (Meng et al., 2019; Li et al., 2019; Wang et al., 2020). This is because many class names are actually summarizing phrases provided by humans (*e.g.*, "*grocery & gourmet food*" in Fig. 1). As a result, we need to design a method that works under such a scenario.

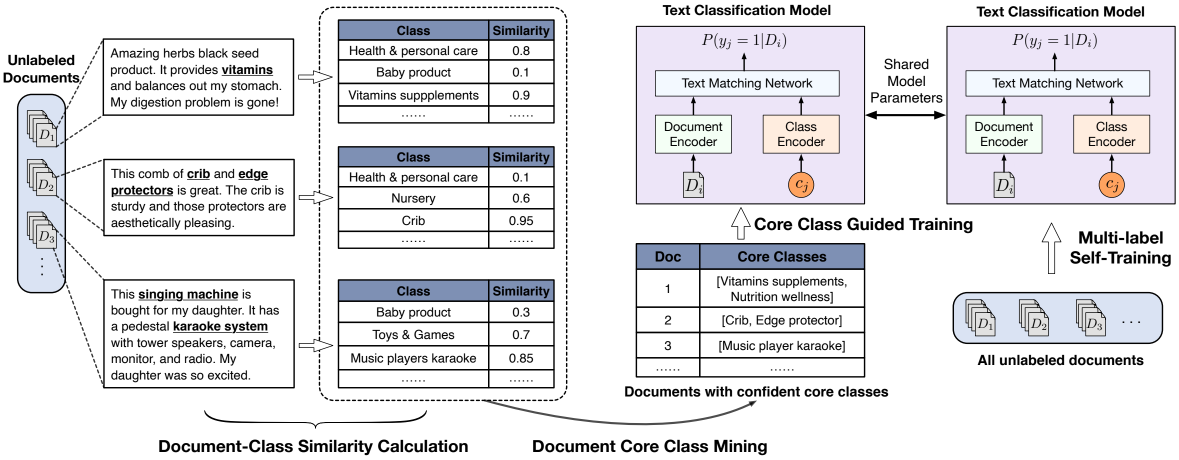
3 Our TaxoClass Framework

Our TaxoClass framework consists of four major steps: (1) document-class similarity calculation, (2) document core class mining, (3) core class guided classifier training, and (4) multi-label self-training. Fig. 2 shows our framework overview and below sections discuss each step in more details.

3.1 Document-Class Similarity Calculation

We take a textual entailment approach (Yin et al., 2019) to calculate the semantic similarity between each ⟨document, class⟩ pair. This approach imitates how humans determine whether a document is similar to a class or not — we read this document, create a hypothesis by filling the class name into a template (*e.g.*, "*this document is about* __"), and ask ourselves to what extent this hypothesis is correct, given the context document.

In this work, we adopt a pre-trained textual entailment model that inputs a document D_i as the


Unlabeled Documents
D_1
D_2
D_3
Text Classification Model
$P(y_j = 1|D_i)$
Text Matching Network
Document Encoder
Class Encoder
D_i
c_j
Shared Model Parameters
Core Class Guided Training
Multi-label Self-Training
All unlabeled documents
Documents with confident core classes
Document-Class Similarity Calculation
Document Core Class Mining

Amazing herbs black seed product. It provides **vitamins** and balances out my stomach. My digestion problem is gone!

Class	Similarity
Health & personal care	0.8
Baby product	0.1
Vitamins suppplements	0.9
……	……

This comb of **crib** and **edge protectors** is great. The crib is sturdy and those protectors are aesthetically pleasing.

Class	Similarity
Health & personal care	0.1
Nursery	0.6
Crib	0.95
……	……

This **singing machine** is bought for my daughter. It has a pedestal **karaoke system** with tower speakers, camera, monitor, and radio. My daughter was so excited.

Class	Similarity
Baby product	0.3
Toys & Games	0.7
Music players karaoke	0.85
……	……

Doc	Core Classes
1	[Vitamins supplements, Nutrition wellness]
2	[Crib, Edge protector]
3	[Music player karaoke]
……	……

Figure 2: Our TaxoClass framework overview. We first calculate document-class similarities using a textual entailment model (Sect. 3.1). Then, we identify document core classes (Sect. 3.2) and train a taxonomy-enhanced text classifier (Sect. 3.3). Finally, we generalize the classifier via multi-label self-training (Sect. 3.4). The "shared model parameters" indicates that we do self-training on the same model learned using our identified core classes.

"premise", a template filled with a class name s_j as the "hypothesis", and outputs a probability of how likely this premise can entail the hypothesis. We treat this probability $\mathbf{P}(D_i \rightarrow c_j)$ as the document-class similarity $sim(D_i, c_j)$. More specifically, we use `Roberta-Large-MNLI`[1] as our textual entailment model which utilizes the pre-trained `Roberta-Large` as its backbone and is fine-tuned on the MNLI dataset.

3.2 Document Core Class Mining

When asked to tag a document with a set of classes from a class taxonomy, humans will first pinpoint a few classes that are most essential to this document. We refer to those most essential classes as the "core classes" and identify them in below two steps.

3.2.1 Core Class Candidate Selection

We observe that on average each document is tagged with a small set of classes from the entire class taxonomy. Therefore, we first reduce the search space of core classes using a top-down approach (c.f. Fig. 3). Given a document D, we start with the "*Root*" class at level $l = 0$, find its two children classes that have the highest similarity with D, and add them into a queue. Then, for each class at level l in the queue, we select $l + 2$ classes from its children classes that are most similar to D. After all level l classes are processed, we aggregate all selected children classes and choose $(l+1)^2$ classes (at level $l + 1$) with the highest path score (ps) defined below:

$$\begin{aligned} ps(Root) &= 1, \\ ps(c_j) &= \max_{c_k \in Par(c_j)} \{ps(c_k) \cdot sim(c_j, D)\}, \end{aligned} \tag{1}$$

where $Par(c_j)$ is class c_j's parent class set. All chosen classes (at level $l + 1$) will be pushed into the queue and we stop this process when no class in the queue has further children. Finally, all classes that have entered the queue, except for the "*Root*" class, consist of the candidate core class set. We use $\mathbb{C}_i^{cand}$ to denote the candidate core class set of document D_i.

3.2.2 Confident Core Class Identification

For each document, we identify its core classes from the above selected candidate set based on two observations. First, a document usually has higher similarity with its core class c than with the parent and sibling classes of c. Take the document D_2 in Fig. 2 as an example, the similarity between D_2 and its core class "*crib*" is 0.95, much higher than the similarity between D_2 and core class's parent class "*nursery*" (0.6) as well as core class's sibling classes. Based on this observation, we define the "confidence score" of a candidate core class c for a document D as below:

$$conf(D, c) = sim(D, c) - \max_{c' \in Par(c) \cup Sib(c)} \{sim(D, c')\}, \tag{2}$$

where $Sib(c)$ represents the sibling class set of c.

Our second observation is that the similarity between a document D and its core class c is salient from a *corpus-wise* perspective. Namely, if a class c is a document D's core class, the confidence score

[1] https://huggingface.co/roberta-large-mnli

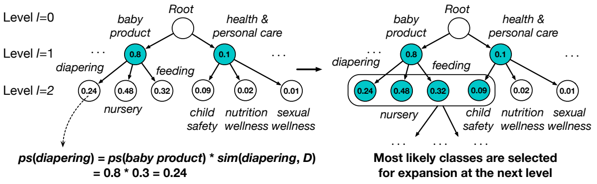


Figure 3: Top-down core class candidate selection.

$conf(D, c)$ is higher than the median confidence score[2] between class c and all documents tagged with c (denoted as $\mathcal{D}(c)$). Formally, we have:

$$conf(D, c) \geq median\{conf(D', c) | D' \in \mathcal{D}(c)\}. \quad (3)$$

According to this observation, we check each class in document D_i's candidate core set $\mathbb{C}_i^{cand}$ and add classes that satisfy the above criteria into the final core class set $\mathbb{C}_i$. Note here this core class set $\mathbb{C}_i$ could be empty when document D_i does not have any confident core class.

3.3 Core Class Guided Classifier Training

Based on identified document core classes, we train one classifier for hierarchical multi-label text classification. Below we first introduce our classifier architecture and then present our training method.

3.3.1 Text Classifier Architecture

We design our classifier to have a dual-encoder architecture: one document encoder maps document D_i to its representation $\mathbf{D}_i$, one class encoder learns class c_j's representation $\mathbf{c}_j$, and one matching network returns the probability of document D_i being tagged with class c_j.

Document Encoder. In this work, we instantiate our document encoder $g_{\text{doc}}(\cdot)$ to be a pre-trained `BERT-base-uncased` model (Devlin et al., 2019) and follow previous work (Chang et al., 2019; Meng et al., 2020) to use the `[CLS]` token representation as the document representation.

Class Encoder. For class encoder $g_{\text{class}}(\cdot)$, we follow (Shen et al., 2020) and use a graph neural network (GNN) (Kipf and Welling, 2017) to model the class taxonomy structure. This taxonomy-enhanced class encoder can capture both the textual information from class surface names and structural information from the class taxonomy.

Given a class c_j, we first obtain its ego network that includes its parent and children classes in the class taxonomy, as shown in Fig. 4. Then, we input this ego network to a GNN that propagates node features over the network structure. The node features are initialized with the pre-trained word embeddings of class surface names[3]. The propagation mechanism updates the feature of a node u by iteratively aggregating representations of its neighbors and itself. Formally, we define a GNN with L-layers as follows:

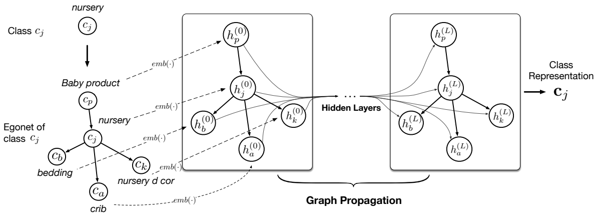


Figure 4: Taxonomy-enhanced class encoder.

$$h_u^{(l)} = \text{ReLU}\left(\sum_{v \in N(u)} \alpha_{uv}^{(l-1)} \mathbf{W}^{(l-1)} h_v^{(l-1)}\right), \quad (4)$$

where $l \in \{1, \ldots, L\}$, $N(u)$ includes node u's neighbors and itself, $\alpha_{uv}^{(l-1)} = \frac{1}{\sqrt{|N(u)||N(v)|}}$ is a normalization constant (same for all layers), and $\mathbf{W}^{(l-1)}$ are learnable parameters.

After obtaining individual node features, we combine them into a vector representing the whole ego network G as follows:

$$h_G = \frac{1}{|G|} \sum_{u \in G} h_u^{(L)}. \quad (5)$$

As this ego network is centered on class c_j and encodes its both textual and structural information, we treat this final graph representation as the class representation $\mathbf{c}_j$.

Text Matching Network. Based on the document representation $\mathbf{D}_i$ and the class representation $\mathbf{c}_j$, we use a log-bilinear text matching model to compute the probability of document D_i being tagged with class c_j as follows:

$$p_{ij} = \mathbf{P}(y_j = 1 | D_i) = \sigma(\exp(\mathbf{c}_j^T \mathbf{B} \mathbf{D}_i)), \quad (6)$$

where $\sigma(\cdot)$ is the sigmoid function and $\mathbf{B}$ is a learnable interaction matrix.

3.3.2 Text Classifier Training

We use our discovered document confident core classes to train a text classifier. One intuitive strategy is to treat each document's core classes as positive classes and all the remaining classes as negative classes. However, this strategy has a high false

[2] We have also tried using "average" but empirically found that using "median" is better and more robust to outliers.

[3] For multi-gram class names, we use their averaged word embeddings.

Algorithm 1: TaxoClass Framework.

```
Input: An unlabeled corpus $\mathcal{D}$, a class taxonomy
       $\mathcal{T}$ with class names $\mathcal{S}$, an entailment
       model $\mathcal{M}$, total number of batches $B$.
Output: A trained classifier $f(\cdot)$.
Use model $\mathcal{M}$ to compute document-class similarity
  (c.f. Sect. 3.1);
Obtain document core classes $\{(D_i, \mathbb{C}_i) \mid D_i \in \mathcal{D}\}$
  (c.f. Sect. 3.2);
Train classifier $f(\cdot)$ with Eq. (8);
for $i$ from 1 to $B$ do
    if $i$ mod 25 = 0 then
        Update $Q$ with Eq. (10);
    Train classifier $f(\cdot)$ with Eq. (9);
Return $f(\cdot)$;
```

negative rate because some non-core classes could still be relevant to the document (c.f. Fig. 1).

We observe a document's multiple labeled classes usually have some ancestor-descendent relations in the class hierarchy $\mathcal{T} = (\mathcal{C}, \mathcal{R})$. This implies that given a document's core class, its parent class and some of its children classes are also likely to be tagged with this document. Therefore, we introduce all core classes' parent classes into the positive class set and exclude their children classes from the negative class set. Formally, given a document D_i with its core class set $\mathbb{C}_i$, we define its positive and negative class set as follows:

$$\begin{aligned} \mathbb{C}_i^{pos} &= \left(\bigcup_{c_j \in \mathbb{C}_i} Par(c_j) \right) \cup \mathbb{C}_i, \\ \mathbb{C}_i^{neg} &= \mathcal{C} - \mathbb{C}_i^{pos} - \bigcup_{c_j \in \mathbb{C}_i} Chd(c_j), \end{aligned} \tag{7}$$

where $Chd(c_j)$ is class c_j's children class set. Finally, we train our classification model using the below binary cross entropy (BCE) loss:

$$\mathcal{L} = -\sum_{\substack{i=1 \\ \mathbb{C}_i \neq \emptyset}}^{|\mathcal{D}|} \Big(\sum_{c_j \in \mathbb{C}_i^{pos}} \log p_{ij} + \sum_{c_j \in \mathbb{C}_i^{neg}} \log(1 - p_{ij}) \Big), \tag{8}$$

where "$\emptyset$" indicates an empty set and we exclude the documents without any confident core class from the loss calculation.

Dataset	# Train	# Test	# Classes
Amazon-531	29,487	19,685	531
DBPedia-298	196,665	49,167	298

Table 1: Dataset statistics. Supervised methods are trained on the entire training set. Weakly-supervised methods are trained by treating the training set as unlabeled data. All methods are evaluated on the test set.

3.4 Multi-label Self-Training

After training the text classifier based on document core classes, we propose to further refine the model via self-training on the entire unlabeled corpus $\mathcal{D}$ for better generalization. The idea of self-training (ST) (Xie et al., 2016) is to iteratively use the model's current prediction P to compute a target distribution Q which guides the model for refinement. In general, the ST objective is expressed with the KL divergence loss as below:

$$\mathcal{L}_{ST} = \mathrm{KL}(Q||P) = \sum_{i=1}^{|\mathcal{D}|} \sum_{j=1}^{|\mathcal{C}|} q_{ij} \log \frac{q_{ij}}{p_{ij}}. \tag{9}$$

The target distribution Q is constructed by enhancing high-confidence predictions while down-weighting low-confidence ones:

$$q_{ij} = \frac{p_{ij}^2 / (\sum_i p_{ij})}{p_{ij}^2 / (\sum_i p_{ij}) + (1 - p_{ij})^2 / (\sum_i (1 - p_{ij}))}. \tag{10}$$

Different from the previous studies (Meng et al., 2018; Yu et al., 2020), our target distribution Q can be applied to multi-label classification problem as it normalizes the current predictions P for each individual class. Intuitively, this equation can enhance high-confidence predictions while down-weighting low-confidence predictions. This is because if example i is more confidently labeled with class j than other examples, we will have a large p_{ij} that dominates the $\sum_i p_{ij}$ term. Consequently, Eq 10 computes a large q_{ij}, which further pushes the model to predict class j for example i.

In practice, instead of updating the target distribution Q for every training example, we update it every 25 batches[4] and train the model with Eq. (9), which makes the self-training process more efficient and robust. We summarize our TaxoClass framework in Algorithm 1.

4 Experiments

4.1 Datasets

We use two public datasets from different domains to evaluate our method: (1) **Amazon-531** (McAuley and Leskovec, 2013) contains 49,145 product reviews and a three-level class taxonomy consisting of 531 classes; and (2) **DBPedia-298** (Lehmann et al., 2015) includes 245,832

[4] This hyper-parameter controls the update frequency. Empirically, we find our model is insensitive to this hyper-parameter (in the typical value range of 10-100).

Wikipedia articles and a three-level class taxonomy with 298 classes. Documents in both datasets are lower-cased and truncated to has maximum 500 tokens. We list the data statistics in Table 1.

4.2 Compared Methods

To the best of our knowledge, we are the first to study *weakly-supervised* HMTC problem and there is no directly comparable baseline under the exact same setting as ours. Therefore, we choose a wide range of representative methods that are most related to TaxoClass and adapt them to our problem setting, described as follows.

- **Hier-doc2vec** (Le and Mikolov, 2014)[5]: This *weakly-supervised* method first embeds documents and classes into a shared semantic space, and then recursively selects the class of the highest embedding similarity with the document in a top-down fashion. We set the embedding dimensionality to be 100 and use the default value for all other hyper-parameters.[6]
- **WeSHClass** (Meng et al., 2019)[7]: Another *weakly-supervised* method that generates pseudo documents to pre-train a text classifier and bootstraps the pre-trained classifier on unlabeled documents with self-training. The class surface names are treated as the "class-related keywords" in this method. For the pseudo document generation step, we use its internal LSTM language model. We treat all classes in its returned class path as the output classes.
- **SS-PCEM** (Xiao et al., 2019)[8]: This *semi-supervised* method uses a generative model to generate documents based on a class path sampled from the class taxonomy. Both labeled and unlabeled documents are used to fit this generative model via the EM algorithm. Finally, it uses the posterior probability of a test document to predict its labeled classes. Among different base classifiers, we choose their author reported best variant PCEM in this study. We use 30% of labeled training documents for this method.
- **Hier-0Shot-TC** (Yin et al., 2019)[9]: This *zero-shot* method uses a pre-trained textual entailment model to predict to what extent a document (as the premise text) can entail a template filled with the class name (as the hypothesis text). Similar to **Hier-doc2vec**, we select the class with the highest entailment score at each level in a top-down recursive fashion. For fair comparison, we change its internal `BERT-base-uncased` model to `RoBERTa-large-mnli` model as is used in our method.
- **TaxoClass**[10]: Our proposed *weakly-supervised* framework that identifies document core classes, leverages core classes to train a taxonomy-enhanced text classifier, and generalizes the classifier using multi-label self-training. We also evaluate two ablations: **TaxoClass-NoST** which removes the multi-label self-training step, and **TaxoClass-NoGNN** which replaces the GNN-based class encoder with a simple embedding layer initialized with pre-trained word embeddings (c.f. Sect. 3.3.1).

4.3 Evaluation Metrics

We follow previous studies (Partalas et al., 2015; Prabhu et al., 2018) and evaluate the multi-label classification results from different aspects using various metrics. The first metric is **Example-F1**[11] which calculates the average F1 scores for all documents as follows:

$$\text{Example-F1} = \frac{1}{N}\sum_{i=1}^{N}\frac{2|\mathbb{C}_i^{true} \cap \mathbb{C}_i^{pred}|}{|\mathbb{C}_i^{true}| + |\mathbb{C}_i^{pred}|},$$

where $\mathbb{C}_i^{true}$ ($\mathbb{C}_i^{pred}$) is the true (model predicted) class set of document D_i.

Moreover, as many applications formalize the HMTC as a class ranking problem (Jain et al., 2016; Guo et al., 2019), we convert predicted class set $\mathbb{C}_i^{pred}$ into a rank list $\mathbb{R}_i^{pred}$ based on each class's model predicted probability and calculate **Precision at k ($P@k$)** as follows:

$$P@k = \frac{1}{N}\sum_{i=1}^{N}\frac{|\mathbb{C}_i^{true} \cap \mathbb{R}_{i,1:k}^{pred}|}{min(k, |\mathbb{C}_i^{true}|)},$$

[5] https://radimrehurek.com/gensim/models/doc2vec.html

[6] We also test the Flat-doc2vec variant which directly ranks all classes in the taxonomy and returns top ranked classes. Its performance is significantly worse than Hier-doc2vec and thus we only report Hier-doc2vec results.

[7] https://github.com/yumeng5/WeSHClass

[8] https://github.com/HKUST-KnowComp/PathPredictionForTextClassification

[9] https://github.com/yinwenpeng/BenchmarkingZeroShot

[10] https://github.com/mickeystroller/TaxoClass

[11] This metric is also called "micro-Dice coefficient".

Method	Amazon-531				DBPedia-298			
	Example-F1	P@1	P@3	MRR	Example-F1	P@1	P@3	MRR
Hier-doc2vec (Le and Mikolov, 2014)	0.3157	0.5805	0.3115	N/A	0.1443	0.2635	0.1443	N/A
WeSHClass (Meng et al., 2019)	0.2458	0.5773	0.2517	N/A	0.3047	0.5359	0.3048	N/A
TaxoClass-NoST	0.5431	0.7918	0.5414	0.5911	0.7712	0.8621	0.7712	0.8221
TaxoClass-NoGNN	0.5271	0.7642	0.5213	0.5621	0.7241	0.8154	0.7241	0.7692
TaxoClass	**0.5934**	**0.8120**	**0.5894**	**0.6332**	**0.8156**	**0.8942**	**0.8156**	**0.8762**
SS-PCEM (Xiao et al., 2019)	0.2921	0.5369	0.2948	0.3004	0.3845	0.7424	0.3845	0.4032
Hier-0Shot-TC (Yin et al., 2019)	0.4742	0.7144	0.4610	N/A	0.6765	0.7871	0.6765	N/A

Table 2: Evaluation of all compared methods on two datasets. For some methods predicting a class path in a top-down fashion rather than returning all classes' probabilities, we cannot compute their MRR scores and indicate this using "N/A".

where $\mathbb{R}^{pred}_{i,1:k}$ is each method predicted top k most likely classes for D_i. Finally, for methods able to return the probability of a document being tagged with each class in the taxonomy, we calculate their **Mean Reciprocal Rank (MRR)** as follows:

$$\text{MRR} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|\mathbb{C}_i^{true}|}\sum_{c_j\in\mathbb{C}_i^{true}}\frac{1}{R_{ij}},$$

where R_{ij} is the "rank" of document D_j's true class c_j in model predicted rank list (over all classes).

4.4 Experiment Settings

For all baseline methods except Hier-doc2vec, we use the public implementations from their authors and leave the hyper-parameters unchanged. For both Hier-0Shot-TC and our method, we adopt the same public `Roberta-Large-MNLI` model as the textual entailment model and use the same hypothesis template: "*this product is about* __." for Amazon-531 dataset and "*this example is* __." for DBPedia-298 dataset. We use AdamW optimizer to train our model with batch size 64, learning rate 5e-5 for all parameters in BERT document encoder and learning rate 4e-3 for all remaining parameters. During the multi-label self-training stage (c.f. Sect. 3.4), we use learning rate 1e-6 for all parameters in the BERT document encoder and 5e-4 for all remaining parameters.

We run all experiments on a single cluster with 80 CPU cores and a Quadro RTX 8000 GPU. All deep learning models are moved to the GPU for faster inference speed. With batch size 64, the TaxoClass framework consumes about 10GB GPU memory. In principle, all methods should be runnable on CPU.

Core Class Mining Method	Example-F1	P@1	P@3	MRR
Explicit Mention	0.1611	0.2168	0.1564	0.2045
0Shot	0.4793	0.7361	0.4782	N/A
Ours	0.5431	0.7918	0.5414	0.5911
Ours-NoCS	0.3812	0.6254	0.3831	0.4366
Ours-NoConf	0.2603	0.4431	0.2521	0.3014

Table 3: Evaluation of core class mining algorithms on Amazon-531 dataset. We train the classifier using different training sets derived from different core class mining algorithm outputs. Please refer to Section 4.6 for detailed descriptions of each method.

4.5 Overall Performance Comparison

Table 2 presents the overall results of all compared methods. First, we find most weakly-supervised and zero-shot method can outperform the semi-supervised method SS-PCEM even the later has access to 30% of labeled documents. Second, we can see that TaxoClass has the overall best performance across all the metrics and defeats the second best method by a large margin. Comparing TaxoClass with TaxoClass-NoGNN, we show the importance of incorporating taxonomy structure into the class encoder. Moreover, the improvement of TaxoClass over TaxoClass-NoST demonstrates the effectiveness of our multi-label self-training.

4.6 Effectiveness of Core Class Mining

We evaluate the effectiveness of our core class mining method as follows. First, we define a set of rival methods and use them to generate various sets of "core classes". Then, we derive pseudo-training data for each generated core class set and use it to learn a text classifier with the same architecture as the one in TaxoClass. Finally, we report each model's performance on the test set. Note here we skip the self-training step to ensure the "core class based pseudo-training data" is the only variable.

Method	Example-F1	P@1	P@3	MRR
fastText	0.4472	0.7515	0.4521	0.4587
TextCNN	0.4787	0.7694	0.4771	0.4827
TaxoClass-NoGNN	0.5271	0.7642	0.5213	0.5621
TaxoClass	0.5934	0.8120	0.5894	0.6332

Table 4: Performance of different classifiers on Amazon-531 dataset. All methods use the same training set derived from our identified document core classes.

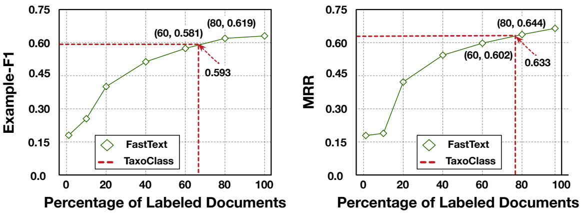


Figure 5: Comparison between TaxoClass and supervised fastText method on Amazon-531 dataset. We train the fastText model using on different percentages of labeled training documents.

Table 3 lists all the results. First, we find that the "Explicit Mention" method, which treats all classes with names explicitly appear in the corpus as the core classes, does not perform well for our HMTC problem. One reason could be many class names are human-curated summarizing phrases that do not appear in the corpus naturally. Second, the "0Shot" method views the output classes of baseline method Hier-0Shot-TC as the core classes and trains a new classifier. Interestingly, this new classifier performs better than the original Hier-0Shot-TC classifier, which shows that transferring knowledge from a general zero-shot classifier to a domain-specific classifier is a possible and promising direction. Finally, we compare variants of our own methods. The "Ours-NoCS" method removes the candidate core class selection step (c.f. Sect. 3.2.1) and treats all classes with high confidence scores as core classes. The "Ours-NoConf" method skips the confident core class identification step (c.f. Sect. 3.2.2) and views all candidate core classes as the final output core classes. We can see a significant performance drop on both ablations, which shows the importance of our two core class mining steps.

4.7 Analysis of Classifier Architecture

We study whether we can use the identified document core classes to train other text classifiers with different architectures such as fastText (Joulin et al., 2016) and TextCNN (Kim, 2014). As shown in Table 4, both methods achieve reasonable performance. We can also see that TaxoClass with and without GNN-enhanced class encoder can outperform both methods. This shows the effectiveness of our dual-encoder style classifier architecture.

4.8 Supervision Signals in Class Names

We vary the percentage of labeled documents on Amazon-531 dataset for training a supervised fastText classifier and present its corresponding performance in Fig. 5. We can see the performance of our TaxoClass framework is equivalent to that of supervised fastText learned on roughly 70% of labeled documents in the training set (i.e., about 20,000 labeled documents).

5 Related Work

Weakly-supervised Text Classification. There exist some previous studies that leverage a few labeled documents or class-indicative keywords as weak supervision signals for text classification. A pioneering method is dataless classification (Chang et al., 2008; Song and Roth, 2014) which embeds documents and classes into the same semantic space of Wikipedia concepts and performs classification using the embedding similarity. Li et al. (2018, 2019) extend this idea by mining concepts directly from the corpus rather than using the external Wikipedia. Along another line, Chen et al. (2015) and Li et al. (2016) propose to apply a seed-guided topic model to infer class-specific topics from class-indicative keywords and to predict document classes from posterior class-topic assignments. Compared with these methods, our TaxoClass framework neither restricts document and class embeddings to live in the same semantic space nor imposes strong statistical assumptions.

Recently, neural models are applied to weakly-supervised text classification. Meng et al. (2018, 2019) propose a pretrain-and-refine paradigm which first generates pseudo documents to pretrain a neural classifier and then refine this classifier via self-training. Mekala and Shang (2020); Meng et al. (2020); Wang et al. (2020) improve the above methods by introducing contextualized weak supervision and using a pre-trained language model to obtain better text representations. While achieving inspiring performance, these methods all assume each document has only one class and all class names (or class-indicative keywords) must appear in the corpus for pseudo training data generation. In this paper, we relax these assumptions and develop a new method for weakly-supervised hierarchical multi-label text classification task.

Zero-shot Text Classification. Zero-shot text classification learns a text classifier based on training documents belonging to *seen* classes and applies the learned classifier to predict testing documents belonging to *unseen* classes (Wang et al., 2019). Nam et al. (2016) jointly embed documents and classes into a shared semantic space where knowledge from seen classes can be transferred to unseen classes. Such an idea is further developed in (Rios and Kavuluru, 2018; Srivastava et al., 2018; Yin et al., 2019; Chu et al., 2020) where external resources (*e.g.*, knowledge graphs, natural language explanations of unseen classes, and open domain data) are introduced to help learn a better shared semantic space. Comparing with these methods, our TaxoClass framework does not require labeled data for a set of seen classes.

Hierarchical Text Classification. Hierarchical text classification leverages a class hierarchy to improve the standard text classification performance. Typical methods can be divided into two categories: (1) *local approaches* which learn a text classifier per class (Banerjee et al., 2019), per parent class (Liu et al., 2005), or per level (Wehrmann et al., 2018), and (2) *global approaches* which incorporate taxonomy structure information into one single classifier through recursive regularization (Gopal and Yang, 2013) or graph neural network (GNN) based encoder (Peng et al., 2018; Huang et al., 2019; Zhou et al., 2020). Our TaxoClass framework adopts the second global approach and uses a GNN-based encoder to obtain each class's representation.

6 Conclusions & Future Work

This paper studies the hierarchical multi-label text classification problem when only class surface names, instead of massive labeled documents, are given. We propose a novel TaxoClass framework which leverages the class taxonomy structure to derive document core classes and learns taxonomy-enhanced text classifier for prediction. Extensive experiments demonstrate the effectiveness of TaxoClass on two real-world datasets from different domains. In the future, we plan to explore how TaxoClass framework can be integrated with semi-supervised methods and data augmentation methods, when some class surface names are too ambiguous to indicate class semantics. Moreover, we consider extending our multi-label self-training method to other related NLP tasks such as fine-grained entity typing.

Discussion of Ethics

As text classification is a standard task in NLP, we do not see any significant ethical concerns. The expected usage of our work is to classify documents such as news articles, scientific literature, and etc.

Acknowledgements

Research was sponsored in part by US DARPA SocialSim Program No. W911NF-17-C0099, NSF IIS 16-18481, IIS 17-04532, and IIS 17-41317, and DTRA HDTRA11810026. Any opinions, findings or recommendations expressed herein are those of the authors and should not be interpreted as necessarily representing the views, either expressed or implied, of DARPA or the U.S. Government. We thank anonymous reviewers for valuable and insightful feedback.

References

S. Banerjee, Cem Akkaya, Francisco Perez-Sorrosal, and K. Tsioutsiouliklis. 2019. Hierarchical transfer learning for multi-label text classification. In *ACL*.

David Berthelot, Nicholas Carlini, I. Goodfellow, Nicolas Papernot, A. Oliver, and Colin Raffel. 2019. Mixmatch: A holistic approach to semi-supervised learning. *ArXiv*, abs/1905.02249.

Ming-Wei Chang, Lev-Arie Ratinov, Dan Roth, and Vivek Srikumar. 2008. Importance of semantic representation: Dataless classification. In *AAAI*.

Wei-Cheng Chang, Hsiang-Fu Yu, Kai Zhong, Yiming Yang, and Inderjit S. Dhillon. 2019. X-bert: extreme multi-label text classification with bert. In *arXiv*.

Xingyuan Chen, Yunqing Xia, Peng Jin, and John A. Carroll. 2015. Dataless text classification with descriptive lda. In *AAAI*.

Zewei Chu, K. Stratos, and Kevin Gimpel. 2020. Natcat: Weakly supervised text classification with naturally annotated datasets. *ArXiv*, abs/2009.14335.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

S. Gopal and Yiming Yang. 2013. Recursive regularization for large-scale classification with hierarchical and graphical dependencies. In *KDD*.

Sylvain Goumy and Mohamed-Amine Mejri. 2018. Ecommerce product title classification. In *SIGIR*.

Chuan Fei Guo, Alireza Mousavi, Xiang Wu, Daniel N. Holtmann-Rice, Satyen Kale, Sashank J. Reddi, and Sanjiv Kumar. 2019. Breaking the glass ceiling for embedding-based classifiers for large output spaces. In *NeurIPS*.

Suchin Gururangan, Tam Dang, Dallas Card, and Noah A. Smith. 2019. Variational pretraining for semi-supervised text classification. In *ACL*.

Wei Huang, Enhong Chen, Qi Liu, Yuying Chen, Zai Huang, Yang Liu, Zhou Zhao, Dan Zhang, and Shijin Wang. 2019. Hierarchical multi-label text classification: An attention-based recurrent network approach. In *CIKM*.

Himanshu Jain, Yashoteja Prabhu, and Manik Varma. 2016. Extreme multi-label loss functions for recommendation, tagging, ranking, and other missing label applications. In *KDD*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Thomas Kipf and M. Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*.

Jens Lehmann, Robert Isele, Max Jakob, A. Jentzsch, D. Kontokostas, Pablo N. Mendes, S. Hellmann, M. Morsey, Patrick van Kleef, S. Auer, and C. Bizer. 2015. Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6:167–195.

Chenliang Li, Jian Xing, Aixin Sun, and Zongyang Ma. 2016. Effective document labeling with very few seed words: A topic model approach. In *CIKM*.

Keqian Li, Shiyang Li, Semih Yavuz, Hanwen Zha, Yu Su, and Xifeng Yan. 2019. Hiercon: Hierarchical organization of technical documents based on concepts. In *ICDM*.

Keqian Li, Hanwen Zha, Yu Su, and Xifeng Yan. 2018. Unsupervised neural categorization for scientific publications. In *SDM*.

T. Liu, Yiming Yang, H. Wan, H. Zeng, Z. Chen, and W. Ma. 2005. Support vector machines classification with a very large-scale taxonomy. *SIGKDD*, 7:36–43.

Julian J. McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*.

Dheeraj Mekala and Jingbo Shang. 2020. Contextualized weak supervision for text classification. In *ACL*.

Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. 2018. Weakly-supervised neural text classification. In *CIKM*.

Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. 2019. Weakly-supervised hierarchical text classification. In *AAAI*.

Yu Meng, Yunyi Zhang, Jiaxin Huang, Chenyan Xiong, Heng Ji, Chao Zhang, and Jiawei Han. 2020. Text classification using label names only: A language model self-training approach. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*.

Jinseok Nam, Eneldo Loza Mencía, and Johannes Fürnkranz. 2016. All-in text: Learning document, label, and word representations jointly. In *AAAI*.

Ioannis Partalas, A. Kosmopoulos, Nicolas Baskiotis, T. Artières, G. Paliouras, Éric Gaussier, Ion Androutsopoulos, M. Amini, and P. Gallinari. 2015. Lshtc: A benchmark for large-scale text classification. *ArXiv*, abs/1503.08581.

Hao Peng, Jianxin Li, Y. He, Yaopeng Liu, Mengjiao Bao, L. Wang, Y. Song, and Qiang Yang. 2018. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. *Proceedings of the 2018 World Wide Web Conference*.

Yashoteja Prabhu, Anil Kag, Shrutendra Harsola, Rahul Agrawal, and Manik Varma. 2018. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *WWW*.

Anthony Rios and Ramakanth Kavuluru. 2018. Few-shot and zero-shot multi-label learning for structured label spaces. In *EMNLP*.

J. Shen, Zhihong Shen, Chenyan Xiong, Chunxin Wang, Kuansan Wang, and Jiawei Han. 2020. Taxoexpan: Self-supervised taxonomy expansion with position-enhanced graph neural network. *Proceedings of The Web Conference 2020*.

Yangqiu Song and Dan Roth. 2014. On dataless hierarchical text classification. In *AAAI*.

Shashank Srivastava, Igor Labutov, and Tom M. Mitchell. 2018. Zero-shot learning of classifiers from natural language quantification. In *ACL*.

Wei Wang, Vincent Wenchen Zheng, Han Yu, and Chunyan Miao. 2019. A survey of zero-shot learning: Settings, methods, and applications. In *ACM TIST*.

Zihan Wang, Dheeraj Mekala, and Jingbo Shang. 2020. X-class: Text classification with extremely weak supervision. *ArXiv*, abs/2010.12794.

Jonatas Wehrmann, R. Cerri, and Rodrigo C. Barros. 2018. Hierarchical multi-label classification networks. In *ICML*.

Huiru Xiao, Xin Liu, and Y. Song. 2019. Efficient path prediction for semi-supervised and weakly supervised hierarchical text classification. *The World Wide Web Conference*.

Junyuan Xie, Ross B. Girshick, and Ali Farhadi. 2016. Unsupervised deep embedding for clustering analysis. In *ICML*.

Peng Xu and Denilson Barbosa. 2018. Neural fine-grained entity type classification with hierarchy-aware loss. In *The 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2018)*. ACL.

Wenpeng Yin, Jamaal Hay, and Dan Roth. 2019. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. In *EMNLP/IJCNLP*.

Ronghui You, Suyang Dai, Zihan Zhang, Hiroshi Mamitsuka, and Shanfeng Zhu. 2019. Attentionxml: Extreme multi-label text classification with multi-label attention based recurrent neural networks. In *NeurIPS*.

Yue Yu, Simiao Zuo, Haoming Jiang, W. Ren, Tuo Zhao, and C. Zhang. 2020. Fine-tuning pre-trained language model with weak supervision: A contrastive-regularized self-training approach. *ArXiv*, abs/2010.07835.

Ziqian Zeng, Wenxuan Zhou, Xin Liu, and Yangqiu Song. 2019. A variational approach to weakly supervised document-level multi-aspect sentiment classification. In *NAACL-HLT*.

Jie Zhou, Chunping Ma, Dingkun Long, Guangwei Xu, Ning Ding, Haoyu Zhang, Pengjun Xie, and G. Liu. 2020. Hierarchy-aware global model for hierarchical text classification. In *ACL*.