Hierarchical Metadata-Aware Document Categorization under Weak Supervision

Yu Zhang[1], Xiusi Chen[2], Yu Meng[1], Jiawei Han[1]
[1]Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA
[2]Department of Computer Science, University of California, Los Angeles, CA, USA
[1]{yuz9, yumeng5, hanj}@illinois.edu, [2]xchen@cs.ucla.edu

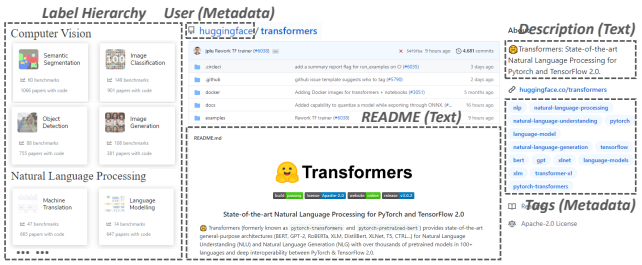
ABSTRACT

Categorizing documents into a given label hierarchy is intuitively appealing due to the ubiquity of hierarchical topic structures in massive text corpora. Although related studies have achieved satisfying performance in fully supervised hierarchical document classification, they usually require massive human-annotated training data and only utilize text information. However, in many domains, (1) annotations are quite expensive where *very few training samples* can be acquired; (2) documents are accompanied by *metadata* information. Hence, this paper studies how to integrate the label hierarchy, metadata, and text signals for document categorization under weak supervision. We develop HiMeCat, an embedding-based generative framework for our task. Specifically, we propose a novel *joint representation learning module* that allows simultaneous modeling of category dependencies, metadata information and textual semantics, and we introduce a *data augmentation module* that hierarchically synthesizes training documents to complement the original, small-scale training set. Our experiments demonstrate a consistent improvement of HiMeCat over competitive baselines and validate the contribution of our representation learning and data augmentation modules.

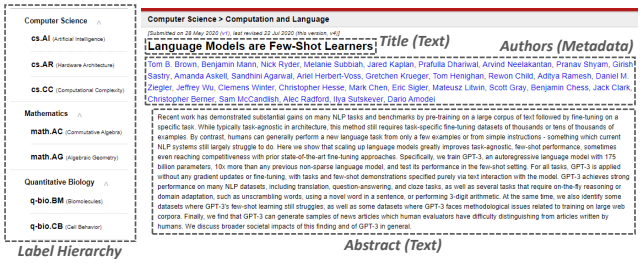
ACM Reference Format:
Yu Zhang[1], Xiusi Chen[2], Yu Meng[1], Jiawei Han[1]. 2021. Hierarchical Metadata-Aware Document Categorization under Weak Supervision. In *Proceedings of the Fourteenth ACM International Conference on Web Search and Data Mining (WSDM '21), March 8–12, 2021, Virtual Event, Israel.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3437963.3441730

1 INTRODUCTION

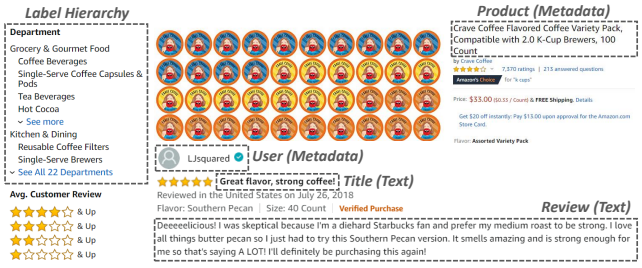
Document categorization is an important task in text mining, with many real applications such as sentiment analysis [39], location prediction [4] and scientific paper tagging [21]. Given a large text corpus, automatically inferring the category of each document not only enables effective organization of the data, but also benefits downstream tasks like document retrieval [47]. Hierarchical document categorization [7, 12] further considers relationships among categories and classifies documents into a given label hierarchy. Leveraging such hierarchical structures is shown to be effective and necessary due to their pervasiveness in web directories[1], business category lists[2] and many other domains (Figure 1).



(a) **GitHub Repository.** Label Hierarchy: PaperWithCode Task Taxonomy (https://paperswithcode.com/sota); Text: Description and README; Metadata: User and Tag.



(b) **arXiv Paper.** Label Hierarchy: arXiv Category Taxonomy (https://arxiv.org/category_taxonomy); Text: Title and Abstract; Metadata: Author.



(c) **Amazon Review.** Label Hierarchy: Amazon Product Catalog [25]; Text: Title and Review; Metadata: User and Product.

Figure 1: Three examples of documents with metadata and corresponding label hierarchies.

Although recent studies on hierarchy-aware neural models [24, 44, 55] and pre-trained language models [5, 38] greatly improve the performance of (hierarchical) text categorization, they are less concerned with two challenges in real applications:

Limited Training Data. Neural classifiers are data hungry and require a significant amount of manually labeled training documents

WSDM '21, March 8–12, 2021, Virtual Event, Israel

ACM ISBN 978-1-4503-8297-7/21/03…$15.00
https://doi.org/10.1145/3437963.3441730

[1]https://en.wikipedia.org/wiki/Yahoo!_Directory
[2]https://www.yelp.com/developers/documentation/v3/all_category_list

to achieve good performance. In some scientific domains (e.g., arXiv papers and GitHub repositories), to obtain a large training set is expensive because annotations have to be acquired from domain experts. In these scenarios, it would be favorable to perform classification without much annotation effort. To be specific, one may only expect very few (e.g., 5) training examples for each category.

Heterogeneous Signals. Documents on the Web are often accompanied by *metadata* information. We raise three examples in Figure 1: each GitHub repository has its creator and several topic tags; each arXiv paper has its author(s); each Amazon review is associated with a user and a product. These signals, as potential category indicators, should be considered together with the plain text sequence. Moreover, the *label hierarchy*, which reflects category dependencies and correlation, should also be viewed as one type of signal. In recognition of these heterogeneous signals, how can we jointly leverage them during category inference?

In this paper, to tackle the two challenges, we develop HiMeCat, an embedding-based generative framework for Hierarchical Metadata-aware document Categorization under weak supervision. HiMeCat features a hierarchical generative process in the embedding space to simultaneously model (1) *category dependencies* via a top-down generative assumption along the label hierarchy, (2) *metadata information* via a joint generative assumption that documents are dependent on both their categories and metadata, and (3) *textual semantics* via a document-word generative assumption. This hierarchical generative process guides the two important steps of our method: (1) *representation learning*: Maximizing the likelihood of the hierarchical generative process yields an embedding learning objective which jointly optimizes the representations of the label hierarchy, metadata and texts to effectively exploit the **heterogeneous signals**; (2) *data augmentation*: Following the hierarchical generative process allows us to synthesize training documents that augment the original, small-scale training set, which overcomes the **limited training data** challenge. Combining real and synthesized training data, we train a neural classifier for hierarchical text categorization by taking word representations learned from the previous step as embedding initialization.

Inspired by related studies on spherical hierarchical clustering [13] and spherical text embedding [29] which better capture directional similarities and outperform Euclidean space models, we propose to define our hierarchical generative process in the spherical space, where conditional probabilities are described by the von Mises-Fisher (vMF) distribution [8] and Riemannian optimization [2] is adopted for embedding learning.

To summarize, our contributions are as follows:

- We explore the task of jointly leveraging limited supervision and metadata information for hierarchical text classification.
- We develop HiMeCat with a representation learning module and a data augmentation module guided by a novel hierarchical generative process in the spherical space. Our representation learning module jointly learns embeddings from heterogeneous signals, and our data augmentation module synthesizes training documents to address the supervision scarcity bottleneck.
- We conduct comprehensive experiments on three datasets from different domains and observe a consistent improvement of HiMeCat over competitive baselines. We also show that leveraging the hierarchy, leveraging metadata, and generating training samples are all beneficial to the categorization performance.

2 PRELIMINARIES

2.1 Problem Definition

The inputs of our hierarchical metadata-aware text categorization task are a collection of documents $\mathcal{D} = \{d_1, ..., d_{|\mathcal{D}|}\}$ and a label hierarchy $\mathcal{T}$. Each document $d_i \in \mathcal{D}$ has both text and metadata information. We explain these concepts one by one.

Text. In this paper, text refers to all free-text fields of a document. For example, in Figure 1, a GitHub repository has its description and README file as text information; an arXiv paper has its title and abstract. To simplify our discussion, we concatenate all these fields into one word sequence, denoted as $w_1 w_2 ... w_n$.

Metadata. Each document can have multiple types of metadata (e.g., an Amazon review has both user and product information). Given a specific metadata type, there may be an arbitrary number (could be zero) of metadata instances associated with a document (e.g., a repository may have no tags, and a paper can have more than one author). For a document d, we represent its metadata as a set of metadata instances $\mathcal{M}_d = \{m_1, ..., m_{|\mathcal{M}_d|}\}$.

Label Hierarchy. We assume the label hierarchy $\mathcal{T}$ has a tree structure. Each node $l \in \mathcal{T}$ represents a category. If, in some cases, $\mathcal{T}$ is a directed acyclic graph (DAG), we follow [55] and convert $\mathcal{T}$ to a tree by distinguishing each label node as a single-path node.

Formally, we define our task as follows:

Problem Definition. Given a collection of documents $\mathcal{D}$ and a label hierarchy $\mathcal{T}$, where each *leaf* category $l \in \mathcal{T}$ is characterized by a small set of training documents $\mathcal{D}_l \subset \mathcal{D}$, the task is to assign appropriate category labels to the remaining documents $\mathcal{D} \backslash (\cup_l \mathcal{D}_l)$. The labels of each document should form a path in $\mathcal{T}$.

2.2 The Von Mises-Fisher Distribution

The von Mises-Fisher (vMF) distribution [8, 26] will be extensively used in our proposed framework. It defines a probability density over a unit sphere and is parameterized by a mean direction vector $\boldsymbol{\mu}$ and a concentration parameter κ. Formally, let $\mathbb{S}^{p-1} = \{\boldsymbol{x} \in \mathbb{R}^p : ||\boldsymbol{x}||_2 = 1\}$ denote the p-dimensional unit sphere. The probability density function for $\boldsymbol{x} \in \mathbb{S}^{p-1}$, $||\boldsymbol{\mu}||_2 = 1$, $\kappa > 0$ is given by

$$f_{\text{vMF}}(\boldsymbol{x}|\boldsymbol{\mu}, \kappa) = c_p(\kappa) \exp(\kappa \cdot \cos(\boldsymbol{x}, \boldsymbol{\mu})). \tag{1}$$

Here, the normalization constant $c_p(\kappa)$ is

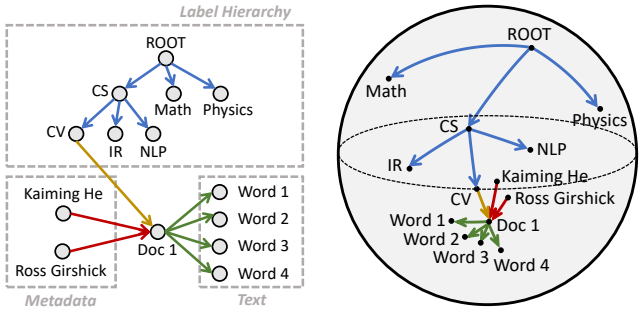
$$c_p(\kappa) = \frac{\kappa^{p/2-1}}{(2\pi)^{p/2} I_{p/2-1}(\kappa)}, \tag{2}$$

where $I_r(\cdot)$ represents the modified Bessel function of the first kind at order r [8, 26]. Intuitively, the vMF distribution is an analogue of the Gaussian distribution on a sphere. The distribution concentrates around the mean direction $\boldsymbol{\mu}$, and is more concentrated if κ is large.

3 MODEL

3.1 A Hierarchical Generative Process

Motivation. In recognition of various types of information (i.e., the label hierarchy, weak supervision, metadata, and text), we propose a probabilistic generative process to characterize these heterogeneous signals. By adopting the vMF distribution, our generative process is defined in a spherical space (i.e., $\mathbb{S}^{p-1}$) instead of a Euclidean one (i.e., $\mathbb{R}^p$). The motivation of doing so is inspired by two lines of related studies. First, the vMF distribution has demonstrated its



(a) The Hierarchical Generative Process (b) The Joint Spherical Embedding Space

Figure 2: Illustration of the proposed hierarchical generative process and the joint hierarchy-metadata-text representation learning module guided by this process.

effectiveness in several tasks including hierarchical clustering [13] and text sequence generation [19], which bears some similarities with our hierarchical representation learning and data augmentation modules, respectively. Second, normalizing word embedding vectors onto a sphere [20, 49] or directly training embeddings in a spherical space [29] can better capture directional similarities, which is proved to be beneficial to tasks like semantic similarity search and document classification [29].

As shown in Figure 2(a), the process can be divided into the following steps.

Parent Label → Child Label. Given the label hierarchy $\mathcal{T}$, the semantics of each child category l_c should largely depend on the semantics of its parent category l_p. In other words, if we use an embedding vector to represent each category, the child embedding $\boldsymbol{l}_c$ should be close to its parent embedding $\boldsymbol{l}_p$. Inspired by the softmax function used in word embedding [32] and network embedding [40], we define the generation probability as

$$p(l_c|l_p) \propto \exp(\boldsymbol{l}_c^T \boldsymbol{l}_p). \tag{3}$$

Here, one needs to notice that $p(\cdot|l_p)$ describes a probability distribution of an embedding vector $\boldsymbol{l}_c$. Since $\boldsymbol{l}_c$ can be any point in a *continuous* embedding space, $p(\cdot|l_p)$ should also be a *continuous* distribution. Therefore, we cannot fully borrow the softmax function which defines a *discrete* choice from finite candidates. Instead, the vMF distribution introduced in Section 2.2 is a proper tool here.

Assume all category embedding vectors reside on a sphere $\mathbb{S}^{p-1}$. Recall Eq. (1). If we set $\kappa = 1$, then

$$f_{\text{vMF}}(\boldsymbol{x}|\boldsymbol{\mu}, 1) = c_p(1)\exp(\cos(\boldsymbol{x}, \boldsymbol{\mu})) = c_p(1)\exp(\boldsymbol{x}^T\boldsymbol{\mu}). \tag{4}$$

By comparing Eq. (4) with Eq. (3), we know that the vMF distribution meets our requirement of the generation probability (because $c_p(1)$ is a constant given the dimension p). Thus, we rewrite Eq. (3) as

$$p(l_c|l_p) = f_{\text{vMF}}(\boldsymbol{l}_c|\boldsymbol{l}_p, 1) = c_p(1)\exp(\boldsymbol{l}_c^T\boldsymbol{l}_p), \quad \forall \boldsymbol{l}_c \in \mathbb{S}^{p-1}. \tag{5}$$

Given the conditional probability, the category embedding can be generated in a top-down manner: we first determine the embedding of $\mathcal{T}$'s root, and then proceed to the next level to generate embeddings of categories at that level. The embeddings of child labels are supposed to be surrounding the embedding of their parent label, and, according to the property of the vMF distribution, we have $\mathbb{E}[\boldsymbol{l}_c|\boldsymbol{l}_p] = \boldsymbol{l}_p$. This process can be repeated until we reach the leaf categories.

Label & Metadata → Document. Our second step generates document embeddings according to their metadata and label information (if applicable). Consider how we humans write a document given a topic category (or a tag/product). We need to first have a general idea of the article, which can be represented as the document embedding $\boldsymbol{d}$. To impose the coherence between $\boldsymbol{d}$ and its category/metadata information, we assume

$$p(d|l_d, \mathcal{M}_d) \propto \exp(\boldsymbol{d}^T\boldsymbol{l}_d) \cdot \prod_{m_d \in \mathcal{M}_d} \exp(\boldsymbol{d}^T\boldsymbol{m}_d), \tag{6}$$

where l_d is the category label of d (here, we are referring to the leaf label of d); m_d is a metadata instance of d; the corresponding embedding variables are in bold. Following the derivation above, by assuming all embedding vectors reside on a sphere, we can rewrite Eq. (6) as

$$\begin{aligned} p(d|l_d, \mathcal{M}_d) &\propto \prod_{z \in \{l_d\}\cup\mathcal{M}_d} c_p(1)\exp(\boldsymbol{d}^T\boldsymbol{z}) \\ &= \prod_{z \in \{l_d\}\cup\mathcal{M}_d} f_{\text{vMF}}(\boldsymbol{d}|\boldsymbol{z}, 1), \quad \forall \boldsymbol{d} \in \mathbb{S}^{p-1}. \end{aligned} \tag{7}$$

One unique problem we need to address here is that label and metadata information can be missing. For example, due to supervision scarcity, only a small proportion of documents in $\mathcal{D}$ have label information. Besides, a GitHub repository may have no tags. In general, suppose V instances $z_1, ..., z_V$ are missing in $\{l_d\} \cup \mathcal{M}_d$. In this case, we assume their embeddings $\boldsymbol{z}_1, ..., \boldsymbol{z}_V$ can be any vector on the sphere with equal probability (i.e., $\boldsymbol{z}_1, ..., \boldsymbol{z}_V \sim U(\mathbb{S}^{p-1})$ i.i.d.). Let $O_d = \{l_d\} \cup \mathcal{M}_d \backslash \{z_1, ..., z_V\}$ be the remaining observable instances. Therefore, Eq. (7) becomes

$$\begin{aligned} &p(d|l_d, \mathcal{M}_d) \\ &\propto \mathbb{E}_{\boldsymbol{z}_1,...,\boldsymbol{z}_V \sim U(\mathbb{S}^{p-1})}\left[\prod_{z \in \{l_d\}\cup\mathcal{M}_d} c_p(1)\exp(\boldsymbol{d}^T\boldsymbol{z})\right] \\ &\propto \int_{\mathbb{S}^{p-1}} ... \int_{\mathbb{S}^{p-1}} \left(\prod_{z \in \{l_d\}\cup\mathcal{M}_d} c_p(1)\exp(\boldsymbol{d}^T\boldsymbol{z})\right) \mathrm{d}\boldsymbol{z}_1 ... \mathrm{d}\boldsymbol{z}_V \\ &= \prod_{i=1}^{V}\left(\int_{\mathbb{S}^{p-1}} c_p(1)\exp(\boldsymbol{d}^T\boldsymbol{z}_i)\,\mathrm{d}\boldsymbol{z}_i\right) \cdot \prod_{z \in O_d} c_p(1)\exp(\boldsymbol{d}^T\boldsymbol{z}) \\ &= \prod_{z \in O_d} c_p(1)\exp(\boldsymbol{d}^T\boldsymbol{z}) = \prod_{z \in O_d} f_{\text{vMF}}(\boldsymbol{d}|\boldsymbol{z}, 1). \end{aligned} \tag{8}$$

The second-to-last step holds because $\int_{\mathbb{S}^{p-1}} c_p(1)\exp(\boldsymbol{d}^T\boldsymbol{z}_i)\,\mathrm{d}\boldsymbol{z}_i \equiv 1$ (i.e., the probability density function of a vMF distribution integrates to 1 over the whole sphere). Eq. (8) tells us that, when some label/metadata information is missing, we only need to consider the remaining observable part (i.e., O_d) in the conditional probability.

Document → Word. Based on the document embedding $\boldsymbol{d}$, a sequence of words $w_1, ..., w_n$ is then generated to describe the overall semantics of d. Similar to Eqs. (3) and (6), we assume the probability of word w_i appearing in document d to be

$$p(w_i|d) \propto \exp(\boldsymbol{w}_i^T\boldsymbol{d}). \tag{9}$$

Again, we rewrite the probability using a vMF distribution:

$$p(w_i|d) = f_{\text{vMF}}(\boldsymbol{w}_i|\boldsymbol{d}, 1) = c_p(1)\exp(\boldsymbol{w}_i^T\boldsymbol{d}), \quad \forall \boldsymbol{w}_i \in \mathbb{S}^{p-1}. \tag{10}$$

Eqs. (5), (8) and (10) together fully specify our proposed hierarchical generative process.

3.2 Joint Representation Learning

Guided by the hierarchical generative process, we aim to jointly learn the embeddings of all labels, metadata instances, documents and words. Our learning objective can be divided into two parts: one for modeling the label hierarchy information (corresponding to the first step in our generative process), and the other for modeling corpus statistics and metadata information (corresponding to the second and third steps).

Objective for Label Hierarchy. Given a parent-child label pair (l_p, l_c), our goal is to maximize the log-likelihood $\log p(l_c|l_p)$ during embedding learning. Inspired by studies on knowledge graph embedding [3, 42], we adopt the following margin-based ranking objective:

$$\min\Big(0, \log p(l_c|l_p) - \log p(l'_c|l'_p) - \gamma\Big), \tag{11}$$

where $\gamma > 0$ is a margin hyperparameter and (l'_p, l'_c) is a negative training sample. Given the positive pair (l_p, l_c), we generate the negative pair by replacing the parent label with any other label in $\mathcal{T}$ while keeping the child label. According to this corruption strategy, our objective can be defined as

$$\mathcal{J}_L = \sum_{l_c \in \mathcal{T}\setminus\{\text{ROOT}\}} \sum_{l'_p \in \mathcal{T}\setminus\{l_p, l_c\}} \min\Big(0, \log p(l_c|l_p) - \log p(l_c|l'_p) - \gamma_L\Big). \tag{12}$$

Based on the definition of $p(l_c|l_p)$ in Eq. (5), we have

$$\begin{aligned} &\log p(l_c|l_p) - \log p(l_c|l'_p) \\ =\ &\log\Big(c_p(1)\exp(\boldsymbol{l}_c^T \boldsymbol{l}_p)\Big) - \log\Big(c_p(1)\exp(\boldsymbol{l}_c^T \boldsymbol{l}'_p)\Big) \\ =\ &\boldsymbol{l}_c^T \boldsymbol{l}_p - \boldsymbol{l}_c^T \boldsymbol{l}'_p. \end{aligned} \tag{13}$$

Therefore,

$$\mathcal{J}_L = \sum_{l_c \in \mathcal{T}\setminus\{\text{ROOT}\}} \sum_{l'_p \in \mathcal{T}\setminus\{l_p, l_c\}} \min\Big(0, \boldsymbol{l}_c^T \boldsymbol{l}_p - \boldsymbol{l}_c^T \boldsymbol{l}'_p - \gamma_L\Big). \tag{14}$$

Objective for Metadata and Corpus. Given the label and metadata information of a document d, the log-likelihood of observing the document and its word sequence is

$$\log p(d, w_{1:n}|l_d, \mathcal{M}_d) = \log\Big(p(d|l_d, \mathcal{M}_d) \prod_{i=1}^{n} p(w_i|d)\Big). \tag{15}$$

According to Eqs. (8) and (10), it can be written as

$$\sum_{z \in O_d} \log f_{\text{vMF}}(\boldsymbol{d}|\boldsymbol{z}, 1) + \sum_{i=1}^{n} \log f_{\text{vMF}}(\boldsymbol{w}_i|\boldsymbol{d}, 1) + \text{const}. \tag{16}$$

Again, we adopt the margin-based ranking objective to push the negative pairs distant enough from the positive pairs (z, d) and (d, w_i). Formally,

$$\begin{aligned} \mathcal{J}_{MC} = \sum_{d \in \mathcal{D}} \sum_{d' \in \mathcal{D}\setminus\{d\}} \Bigg(&\sum_{z \in O_d} \min\Big(0, \log f_{\text{vMF}}(\boldsymbol{d}|\boldsymbol{z}, 1) - \log f_{\text{vMF}}(\boldsymbol{d'}|\boldsymbol{z}, 1) - \gamma_M\Big) + \\ &\sum_{i=1}^{n} \min\Big(0, \log f_{\text{vMF}}(\boldsymbol{w}_i|\boldsymbol{d}, 1) - \log f_{\text{vMF}}(\boldsymbol{w}_i|\boldsymbol{d'}, 1) - \gamma_C\Big)\Bigg). \end{aligned} \tag{17}$$

Following the derivation in Eq. (13), we have

$$\begin{aligned} \mathcal{J}_{MC} = \sum_{d \in \mathcal{D}} \sum_{d' \in \mathcal{D}\setminus\{d\}} \Bigg(&\sum_{z \in O_d} \min\Big(0, \boldsymbol{d}^T \boldsymbol{z} - \boldsymbol{d'}^T \boldsymbol{z} - \gamma_M\Big) + \\ &\sum_{i=1}^{n} \min\Big(0, \boldsymbol{w}_i^T \boldsymbol{d} - \boldsymbol{w}_i^T \boldsymbol{d'} - \gamma_C\Big)\Bigg). \end{aligned} \tag{18}$$

To summarize, our joint representation learning step can be formulated as the following optimization problem:

$$\begin{aligned} &\max \mathcal{J} = \mathcal{J}_L + \mathcal{J}_{MC}, \\ &\text{s.t.} \quad ||\boldsymbol{\theta}||_2 = 1, \quad \forall \theta \in \mathcal{T} \cup \mathcal{M} \cup \mathcal{D} \cup \mathcal{V}. \end{aligned} \tag{19}$$

Here, $\mathcal{M}$ is the set of metadata instances appearing in the dataset, and $\mathcal{V}$ is the vocabulary (i.e., the set of words).

Optimization. Given the constraint that all embedding vectors live on a sphere, directly performing stochastic gradient descent in the Euclidean space is not an effective way to optimize the objective. Instead, following recent studies on hyperbolic embedding [33, 34] and spherical embedding [29, 45], we apply the Riemannian gradient method [2] to learn the embeddings.

According to [29], on a unit sphere, the relationship between the Riemannian gradient ∇^R and the Euclidean gradient ∇^E is

$$\nabla^R \mathcal{J}(\boldsymbol{\theta}) = (\boldsymbol{I} - \boldsymbol{\theta}\boldsymbol{\theta}^T)\nabla^E \mathcal{J}(\boldsymbol{\theta}). \tag{20}$$

Based on Eqs. (14) and (18), the Euclidean gradient of each embedding vector is easy to compute. For example, given a parent-child label pair (l_p, l_c), we have

$$\begin{aligned} \nabla^E \mathcal{J}_L(\boldsymbol{l}_c) &= \sum_{l'_p \in \mathcal{T}\setminus\{l_p, l_c\}} \mathbf{1}(\boldsymbol{l}_c^T \boldsymbol{l}_p - \boldsymbol{l}_c^T \boldsymbol{l}'_p < \gamma_L) \cdot (\boldsymbol{l}_p - \boldsymbol{l}'_p), \\ \nabla^E \mathcal{J}_L(\boldsymbol{l}_p) &= \sum_{l'_p \in \mathcal{T}\setminus\{l_p, l_c\}} \mathbf{1}(\boldsymbol{l}_c^T \boldsymbol{l}_p - \boldsymbol{l}_c^T \boldsymbol{l}'_p < \gamma_L) \cdot \boldsymbol{l}_c, \\ \nabla^E \mathcal{J}_L(\boldsymbol{l}'_p) &= \mathbf{1}(\boldsymbol{l}_c^T \boldsymbol{l}_p - \boldsymbol{l}_c^T \boldsymbol{l}'_p < \gamma_L) \cdot (-\boldsymbol{l}_c), \quad \forall \boldsymbol{l}'_p \in \mathcal{T}\setminus\{l_p, l_c\}, \end{aligned} \tag{21}$$

where $\mathbf{1}(\cdot)$ is the indicator function. Then, following Eq. (20), we can obtain the Riemannian gradient $\nabla^R \mathcal{J}(\boldsymbol{l}_c)$, $\nabla^R \mathcal{J}(\boldsymbol{l}_p)$ and $\nabla^R \mathcal{J}(\boldsymbol{l}'_p)$. The gradient of other embedding vectors can be calculated in a similar way. After knowing the Riemannian gradient, we update the parameters in the following form [2, 29]:

$$\boldsymbol{\theta}_{t+1} \leftarrow \frac{\boldsymbol{\theta}_t + \alpha_t \nabla^R \mathcal{J}(\boldsymbol{\theta}_t)}{||\boldsymbol{\theta}_t + \alpha_t \nabla^R \mathcal{J}(\boldsymbol{\theta}_t)||_2}, \tag{22}$$

where α_t is the learning rate at step t. Intuitively, this update can be viewed as an addition along the Riemannian gradient direction followed by a projection onto the unit sphere.

Since computing the gradient requires enumerating all labels or documents, in our actual computation, we adopt the negative sampling strategy [3, 32] to accelerate this process. To be specific, for each positive pair (l_p, l_c), (z, d) or (d, w_i), we sample 5 negative pairs to estimate the gradient.

3.3 Hierarchical Data Augmentation

Now we proceed to tackle the limited training data challenge. After representation learning, we have obtained the embedding vectors of all observed labels, metadata instances, documents and words. Using these embeddings, we can generate *synthesized* training documents for each category. Formally, given a category l, we aim to create a document set $\mathcal{D}_l^*$ to augment its original training set

Algorithm 1 Hierarchical Data Augmentation

Input: embedding vectors ($\boldsymbol{l}$, $\boldsymbol{d}$, $\boldsymbol{w}$); # synthesized docs per category (β)
Generate a bottom-up traverse order of $\mathcal{T}$
for l in the traverse order **do**
  if l is a leaf category **then**
    $\mathcal{D}_l^* = \emptyset$
    for $i = 1$ to β **do**
      Generate $\boldsymbol{d}^*$ according to Eq. (24)
      Generate $\boldsymbol{w}_1^* \boldsymbol{w}_2^* ... \boldsymbol{w}_n^*$ according to Eq. (25)
      $\mathcal{D}_l^* = \mathcal{D}_l^* \cup \{d^*\}$
    end for
  else
    for $x \in \texttt{Child}(l)$ **do**
      $\mathcal{D}_{x,sample}^* \leftarrow$ sample $\frac{\beta}{|\texttt{Child}(l)|}$ docs from $\mathcal{D}_x^*$
    end for
    $\mathcal{D}_l^* = \cup_{x \in \texttt{Child}(l)} \mathcal{D}_{x,sample}^*$
  end if
end for
Output: $\{\mathcal{D}_l^* : l \in \mathcal{T}\}$

$\mathcal{D}_l$. To implement this idea, we seek guidance from our proposed generative process once again.

Data Augmentation for a Leaf Category. Recall Figure 2(a). If l is a leaf category, it will directly participate in the generation of document embeddings according to Eq. (8). If we denote a synthesized document as d^* (to distinguish it from a "real" document d), the generation probability will become

$$p(d^*|l, \mathcal{M}_{d^*}) \propto \prod_{z \in O_{d^*}} f_{\text{vMF}}(\boldsymbol{d}^*|\boldsymbol{z}, \kappa). \tag{23}$$

Note that we do not have any metadata information of the synthesized document. Therefore, the only observable variable here in $\{l\} \cup \mathcal{M}_{d^*}$ is the label l. Also, it is not necessary to generate documents associated with a certain metadata instance because our goal is to predict the label, instead of the metadata information, of a document. Substituting $O_{d^*} = \{l\}$ into Eq. (23), we have

$$p(d^*|l, \mathcal{M}_{d^*}) = f_{\text{vMF}}(\boldsymbol{d}^*|\boldsymbol{l}, \kappa). \tag{24}$$

After getting the document embedding $\boldsymbol{d}^*$, we continue our generative process to create a sequence of words describing d^*. In principle, we should follow Eq. (10) and sample words from $f_{\text{vMF}}(\cdot|\boldsymbol{d}^*, 1)$. However, in practice, we expect that the generated word embedding $\boldsymbol{w}^*$ can be mapped back to a concrete word that has occurred in our vocabulary $\mathcal{V}$. In [19], Kumar and Tsvetkov propose a way that first samples an vector $\boldsymbol{e}$ from a continuous distribution, and then find the nearest neighbor of $\boldsymbol{e}$ in the embedding space among all $w \in \mathcal{V}$. In contrast, we directly require the generated word to reside in the neighborhood of $\boldsymbol{d}^*$ in our joint embedding space. In this way, Eq. (10) degenerates to a discrete softmax function:

$$p(w^*|d^*) = \frac{f_{\text{vMF}}(\boldsymbol{w}^*|\boldsymbol{d}^*, 1)}{\sum_{w \in \mathcal{V}_N(d^*)} f_{\text{vMF}}(\boldsymbol{w}|\boldsymbol{d}^*, 1)} = \frac{\exp(\boldsymbol{w}^{*T}\boldsymbol{d}^*)}{\sum_{w \in \mathcal{V}_N(d^*)} \exp(\boldsymbol{w}^T\boldsymbol{d}^*)}, \tag{25}$$

where $\mathcal{V}_N(d)$ is the set of top-N nearest neighbors of $\boldsymbol{d}^*$ in the embedding space among all $w \in \mathcal{V}$. Using Eq. (25) repeatedly, we can obtain a sequence of words $w_1^* w_2^* ... w_n^*$ as the content of d^*.

Data Augmentation for a Non-Leaf Category. In our hierarchical generative process, a non-leaf label does not directly generate documents. Instead, it determines the embeddings of its children, and the child categories further determine its lower-level descendants. Formally, given a non-leaf label l, let $\mathcal{T}_l \subseteq \mathcal{T}$ be the subtree with root l and $\texttt{Leaf}(\mathcal{T}_l)$ be the leaf descendants of l. Then,

$$p(\mathcal{D}_l^*, \mathcal{T}_l|l) = p(\mathcal{T}_l|l) \cdot \prod_{x \in \texttt{Leaf}(\mathcal{T}_l)} p(\mathcal{D}_x^*|x). \tag{26}$$

After the joint representation learning step, all label embeddings have been fixed. Therefore, for any parent-child pair (l_p, l_c), the probability $p(l_c|l_p) = c_p(1)\exp(\boldsymbol{l}_c^T\boldsymbol{l}_p)$ is fixed. Thus,

$$p(\mathcal{T}_l|l) = p(\texttt{Child}(l)|l) \cdot \prod_{x \in \texttt{Child}(l)} p(\texttt{Child}(x)|x) \cdot ... = \text{const}. \tag{27}$$

Here, $\texttt{Child}(l)$ is the set of l's children. Putting Eqs. (26) and (27) together, we know that the generation of $\mathcal{D}_l^*$ solely depends on $\texttt{Leaf}(\mathcal{T}_l)$ in the current data augmentation step. That being said, for a leaf category $x \in \texttt{Leaf}(\mathcal{T}_l)$, the documents $\mathcal{D}_x^*$ sampled from $p(\mathcal{D}_x^*|x)$ can also be viewed as the documents $\mathcal{D}_l^*$ sampled from $p(\mathcal{D}_l^*, \mathcal{T}_l|l)$. This is intuitive because training samples of a leaf category are naturally training samples of its ancestor categories. We have discussed how to generate synthesized documents for a leaf category. Therefore, we can directly have

$$\mathcal{D}_l^* = \cup_{x \in \texttt{Leaf}(\mathcal{T}_l)} \mathcal{D}_x^*. \tag{28}$$

If we adopt a bottom-up order to generate $\mathcal{D}_l^*$, this can also be written in a recursive way:

$$\mathcal{D}_l^* = \cup_{x \in \texttt{Child}(l)} \mathcal{D}_x^*. \tag{29}$$

In practice, some non-leaf categories have more children than their siblings have. Thus, if we directly apply Eq. (29), they will have more synthesized training data. To avoid class imbalance, we fix $|\mathcal{D}_l^*| = \beta = \text{const}$ for any $l \in \mathcal{T}$. When applying Eq. (29), we sample $\frac{\beta}{|\texttt{Child}(l)|}$ documents from $\mathcal{D}_x^*$ and put them into $\mathcal{D}_l^*$. Algorithm 1 completely summarizes our data augmentation process.

3.4 Hierarchical Classifier Training

After hierarchical data augmentation, every (leaf or non-leaf) category l has a set of synthesized training documents $\mathcal{D}_l^*$ and a set of real training data $\mathcal{D}_l$. (If l is a non-leaf category, we follow the idea of creating $\mathcal{D}_l^*$ and let $\mathcal{D}_l = \cup_{x \in \texttt{Leaf}(\mathcal{T}_l)} \mathcal{D}_x$.) Each training document $d \in \mathcal{D}_l \cup \mathcal{D}_l^*$ has a sequence of words, whose embedding vectors are already learned in joint representation learning.

Now, based on the augmented training data, we need to train a hierarchical text classifier. Hierarchical text classification has been extensively studied for decades with many effective models proposed (see Section 5 for a detailed discussion). The main goal of this paper is to tackle signal heterogeneity and supervision scarcity instead of proposing a novel neural architecture for hierarchical classification. Therefore, we simply adopt the top-down training strategy [7, 18, 22] that classifies documents at the top layer and then propagates the results to the next layer until the leaves. For each non-leaf category $l \in \mathcal{T}$, we need to train a flat classifier that assigns documents to its child classes $\texttt{Child}(l)$ for more fine-grained predictions. We adopt Kim-CNN [17], a popular convolutional architecture, as our flat text classifier. The embeddings $\{\boldsymbol{w} : w \in \mathcal{V}\}$ trained in the joint representation learning step are used as initialization embeddings of the input layer.

Table 1: Dataset Statistics.

Dataset	#Docs	#Classes	#Leaves	#Training	#Testing
GitHub	1,596	18	14	70	1,526
ArXiv	26,400	94	88	440	25,960
Amazon	147,000	166	147	735	146,265

The model adopted here can be easily improved by replacing Kim-CNN with a more advanced flat classifier (e.g., [41, 51]) or by adding a hierarchical regularization (e.g., [12, 14, 35]). However, since we would like to demonstrate the contribution of our embedding and augmentation modules when comparing with baselines, we keep the classifier as simple as possible.

4 EXPERIMENTS

4.1 Experimental Setup

Datasets. We use three datasets from different domains.[3] The statistics of the datasets can be found in Table 1.

- **GitHub [54].** This dataset is collected by paperswithcode.com. It contains 1,596 GitHub repositories implementing state-of-the-art machine learning algorithms for different tasks (e.g., object detection, question answering, speech synthesis, etc.). The tasks form a taxonomy which is viewed as the label hierarchy.
- **ArXiv.** This dataset is crawled from arXiv.org. There are 5 level-1 categories (i.e., cs, math, physics, q-bio, and q-fin) and 88 level-2 categories. For each category, we randomly sample 300 papers to constitute the dataset.
- **Amazon [27].** This dataset is a large collection of Amazon product reviews. The label of each review is its product category (e.g., automotive, car care, beauty, skin care, etc.). These categories are organized into a catalog taxonomy [25]. We select 147 large leaf categories and sample 1,000 reviews for each of them.

According to our weakly supervised setting, we use 5 documents per leaf category for training and all the other documents for testing.

Baseline Methods. We evaluate the performance of HiMeCat against the following classification/embedding algorithms:

- **HierSVM [7]** is a supervised hierarchical classification method. It adopts a top-down classification strategy and trains an SVM for each non-leaf node to distinguish its child categories.
- **WeSHClass [30]** is a weakly supervised hierarchical classification method. It adopts LSTM to generate data for pre-training and iteratively refines results based on the hierarchy.
- **PCEM [48]** is a weakly supervised hierarchical classification method. It proposes a path-cost sensitive hierarchical classifier and applies an EM technique to utilize the unlabeled data.
- **HiGitClass [54]** is a weakly supervised hierarchical classification method. It uses heterogeneous network embedding to encode relationships between labels, text and metadata and then leverages WeSHClass to train a hierarchical classifier.
- **MetaCat [52]** is a weakly supervised flat classification method. It leverages metadata information but cannot utilize the label hierarchy. We use it to directly classify documents to the leaf layer and then infer higher-level labels using the hierarchy.
- **Metapath2vec [6]** is a metadata-aware embedding method. It models the proximity between heterogeneous elements through meta-path guided random walks.
- **Poincaré [33]** is a hierarchy-aware embedding method. It preserves the tree structure of embedded elements by putting them into a hyperbolic space.
- **Pretrained BERT [5]** is a benchmark language model that provides contextualized word representations. For text classification, following [5], we add a classification layer upon the [CLS] token of the last transformer layer and fine-tune the model using labeled documents. After predicting the leaf category of a document, we infer its higher-level labels using the hierarchy.

For Metapath2vec and Poincaré, inspired by our hierarchical generative process, we construct a graph with edges (l_p, l_c), (l, d), (m, d) and (d, w) and perform node embedding on the graph. (We use four meta-paths *l-l*, *d-l-d*, *d-m-d* and *d-w-d* for Metapath2vec.) Then, to apply these two baselines in weakly supervised hierarchical classification, we use the learned embeddings to train a WeSHClass classifier (by replacing its word2vec embedding module). Since some baselines only utilize text information, for each document, we append its metadata instances to the end of its text sequence so that they can exploit these signals.

Parameters. Embedding dimension $p = 100$ for all compared methods except BERT (whose base model is 768-dimensional). Margin hyperparameter $\gamma_L = \gamma_M = \gamma_C = 0.2$. Document-specific vocabulary size $N = 50$. Number of synthesized documents per category $\beta = 500$. For the Kim-CNN classifier, we use one convolutional layer whose filter widths are 2, 3, 4, 5 with 20 feature maps each.

Evaluation Metrics. We use Micro and Macro F1 scores as evaluation metrics. Denote TP_l, FP_l and, FN_l as the instance numbers of true positive, false positive and false negative for a category $l \in \mathcal{T}$. The **Leaf Micro F1** is defined as $\frac{2PR}{P+R}$, where $P = \frac{\sum_{l \in \texttt{Leaf}(\mathcal{T})} TP_l}{\sum_{l \in \texttt{Leaf}(\mathcal{T})} (TP_l + FP_l)}$ and $R = \frac{\sum_{l \in \texttt{Leaf}(\mathcal{T})} TP_l}{\sum_{l \in \texttt{Leaf}(\mathcal{T})} (TP_l + FN_l)}$. The **Leaf Macro F1** is defined as $\frac{1}{|\texttt{Leaf}(\mathcal{T})|} \sum_{l \in \texttt{Leaf}(\mathcal{T})} \frac{2P_l R_l}{P_l + R_l}$, where $P_l = \frac{TP_l}{TP_l + FP_l}$ and $R_l = \frac{TP_l}{TP_l + FN_l}$. Following previous works [30, 48], we also calculate the **Overall Micro/Macro F1**, which is defined accordingly on $\mathcal{T} \backslash \{\text{ROOT}\}$ instead of $\texttt{Leaf}(\mathcal{T})$.

4.2 Performance Comparison

Table 2 shows the categorization performance of compared algorithms on the three datasets. We repeat each experiment five times and report the average F1 scores. To measure statistical significance, we conduct a two-tailed paired t-test to compare HiMeCat and each baseline approach. The significance level ("p-value < 0.01" or "p-value < 0.05") of each result is also marked in Table 2.

We observe that: (1) HiMeCat consistently outperforms the compared baseline methods on all three datasets. In most cases, the performance gap is statistically significant. The improvement of HiMeCat over the best baseline is 8.3%, 13.0% and 47.5% on GitHub, ArXiv and Amazon, respectively. (2) For the three metadata-aware text categorization methods, HiMeCat performs better than HiGitClass and MetaCat, indicating the importance of considering label hierarchy and incorporating it into representation learning. (3) Although Metapath2vec and Poincaré are assumed to be strong tools for metadata and hierarchy embedding *respectively*, they do not achieve competitive performance when we need to *jointly* embed these signals. This observation also emphasizes the contribution of our joint representation learning module. (4) Despite the great success of BERT in many supervised NLP tasks, it does not perform

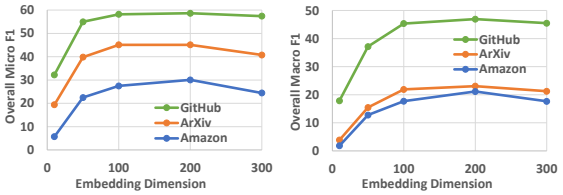
[3] Our code and datasets are available at https://github.com/yuzhimanhua/HIMECat.

Table 2: {Leaf, Overall}×{Micro, Macro} F1 scores of compared algorithms on the three datasets. *: significantly worse than HiMeCat (p-value < 0.05). **: significantly worse than HiMeCat (p-value < 0.01).

	GitHub				ArXiv				Amazon			
	Leaf		Overall		Leaf		Overall		Leaf		Overall	
	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro
HierSVM [7]	0.1861**	0.1388**	0.4862**	0.2457**	0.0538**	0.0460**	0.4066**	0.0750**	0.0248**	0.0217**	0.2218**	0.0494**
WeSHClass [30]	0.1727**	0.1559**	0.3332**	0.1924**	0.0604**	0.0602**	0.3077**	0.0797**	0.0483**	0.0500**	0.1234**	0.0640**
PCEM [48]	0.2519**	0.1234**	0.5299*	0.1786**	0.1090**	0.0717**	0.4440	0.0963**	0.0675**	0.0439**	0.2189**	0.0659**
HiGitClass [54]	0.3984	0.3902*	0.5073**	0.4084**	0.1738**	0.1656**	0.3928**	0.1880**	0.0903**	0.0876**	0.1677**	0.1040**
MetaCat [52]	0.3762**	0.3403**	0.5411*	0.3863**	0.0790**	0.0768**	0.3071**	0.0935**	0.1008**	0.0994**	0.1703**	0.1083**
Metapath2vec [6]	0.2814**	0.2805**	0.4592**	0.3212**	0.1360**	0.1344**	0.3419**	0.1534**	0.0669**	0.0666**	0.1334**	0.0800**
Poincaré [33]	0.2750**	0.1980**	0.4302**	0.2218**	0.1336**	0.1296**	0.2995**	0.1454**	0.0645**	0.0607**	0.1202**	0.0739**
BERT [5]	0.2889**	0.2561**	0.4675**	0.3007**	0.1316**	0.0812**	0.4203**	0.1100**	0.0891**	0.0520**	0.2361**	0.0771**
HiMeCat	**0.4254**	**0.4209**	**0.5820**	**0.4535**	**0.2038**	**0.1938**	**0.4509**	**0.2191**	**0.1552**	**0.1553**	**0.2748**	**0.1770**

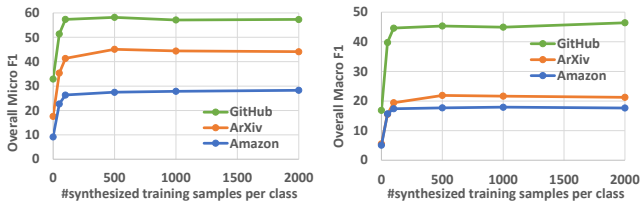
Table 3: Ablation analysis of the joint embedding module. –: the dataset does not have such metadata. * and **: the same as Table 2.

	GitHub, Overall		ArXiv, Overall		Amazon, Overall	
	Micro	Macro	Micro	Macro	Micro	Macro
HiMeCat	**0.5820**	**0.4535**	**0.4509**	**0.2191**	**0.2748**	**0.1770**
No-Hierarchy	0.5649*	0.4385	0.4353*	0.2007*	0.2640**	0.1658**
No-Metadata	0.5262**	0.3949**	0.4208**	0.1951**	0.2196**	0.1117**
No-User/Author	0.5626*	0.4238**	0.4208**	0.1951**	0.2578**	0.1520**
No-Tag	0.5649*	0.4263**	–	–	–	–
No-Product	–	–	–	–	0.2304**	0.1134**



(a) Overall Micro F1 (b) Overall Macro F1

Figure 3: Performance of HiMeCat with different embedding dimensions (p).



(a) Overall Micro F1 (b) Overall Macro F1

Figure 4: Performance of HiMeCat with different numbers of synthesized training samples per class (β).

well in our problem settings. There may be two possible reasons. First, directly using BERT for text classification does not consider metadata and hierarchy information. Second, the small set of training data may not be enough to fine-tune BERT.

4.3 Ablation Study of Representation Learning

Two key steps, representation learning and data augmentation, are proposed in our HiMeCat framework. Now we validate their contribution to the categorization performance through ablation analysis. We start from the representation learning module.

Hierarchy and Metadata. In our joint hierarchy-metadata-text representation learning module, to check whether the label hierarchy and various types of metadata are useful, we create the following ablation versions of HiMeCat.

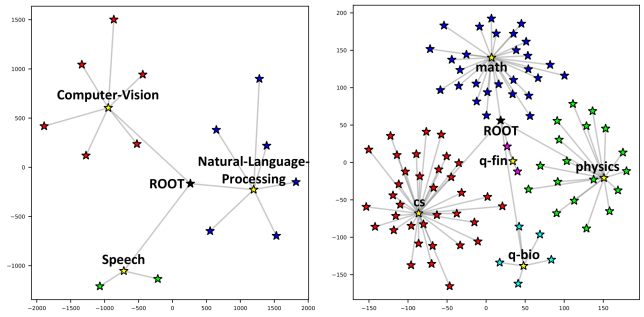
- **No-Hierarchy** does not consider parent-child relationships in representation learning. Formally, it ignores $\mathcal{J}_L$ in Eq. (19) and only maximizes $\mathcal{J}_{MC}$.
- **No-Metadata** does not consider metadata-document relationships in representation learning. Formally, in Eq. (18), we will have $O_d = \{l_d\}$ if d is a training document and $O_d = \emptyset$ otherwise. Moreover, instead of ignoring all metadata instances, we can overlook one specific type of metadata. This will yield ablations including **No-User/Author**, **No-Tag** and **No-Product**.

Table 3 depicts the comparison between the full HiMeCat model and its ablations on the three datasets. (Due to space limitations, we only show Overall F1 scores. Similar observations can be drawn from Leaf F1 scores.) We find that: (1) The full model outperforms No-Hierarchy, which validates our claim that modeling label dependencies and correlation is beneficial to hierarchical categorization. Moreover, when the dataset has a larger label hierarchy, the gap between HiMeCat and No-Hierarchy is more statistically significant. The average improvement of HiMeCat over No-Hierarchy on the six metrics is 5.0%. (2) The full model outperforms No-Metadata, which validates our claim that incorporating metadata signals is helpful. When studying different types of metadata separately, we find all of them play a positive role on our datasets. The contribution of metadata is larger than that of label hierarchy, possibly because metadata information is *denser* than hierarchy information. That being said, each document has its own metadata, while the whole corpus only has one label hierarchy.

Dimension. The embedding dimension p is an important parameter of the representation learning module. To check the sensitivity of p, we plot Overall F1 scores of HiMeCat with $p = 10, 50, 100, 200$ and 300 in Figure 3. We can observe that: if p is too small, the embedding vectors cannot sufficiently capture heterogeneous signals in the dataset; if p is too large, we may face overfitting problems, especially under weak supervision. According to the results, setting p between 100 and 200 is reasonable on our datasets.

4.4 Ablation Study of Data Augmentation

Now we proceed to the hierarchical data augmentation module. We need to answer two questions here: (1) Can data augmentation mitigate the label scarcity problem? (2) How many synthesized



(a) GitHub (b) ArXiv

Figure 5: Tree Embedding Visualization.

training samples should we create for each category? To investigate these two questions, we show the performance of HiMeCat with $\beta = 0, 50, 100, 500, 1000$ and 2000 in Figure 4.

We observe that, when comparing $\beta = 500$ with $\beta = 0$ (i.e., we do not generate any synthesized training data and use real training data only), the performance boost is quite evident. For example, the average absolute improvement of Overall F1 scores on the three datasets is 21.4%. Thus, we validate our claim that the hierarchical data augmentation module can help improve the classification accuracy under weak supervision. Meanwhile, the performance gap between $\beta = 2000$ with $\beta = 500$ is quite subtle. In fact, the average absolute gap of Overall F1 scores on the three datasets becomes 0.1%. Also, too many synthesized training samples will make the training process inefficient. Seeking a balance, we believe setting $\beta = 500$ is an appropriate choice (and we do set $\beta = 500$ in previous experiments).

4.5 Visualization of Tree Embedding

In order to show how the label hierarchy is modeled and how the labels are distributed in our joint embedding space, we plot label embeddings in Figure 5 after reducing their dimensions using t-SNE [23]. Labels are denoted as stars, and names of all level-1 categories and the root are shown. We have two observations: (1) The tree structure of the label hierarchy is well preserved in our embedding space. Child categories center around their parent categories. (2) In the ArXiv embedding space, "q-fin" (Quantitative Finance) is embedded near the root, and its children are embedded closer to "math" and "physics" than they are to "cs" and "q-bio" (Quantitative Biology). This is reasonable because quantitative finance is a quite interdisciplinary area and has many overlap with mathematics and physics. This observation shows that our joint representation learning module not only considers the label hierarchy but also captures semantics from text and/or metadata information.

5 RELATED WORK

Hierarchical Text Classification. Many efforts have been put on how to utilizing the label hierarchy to improve text classification. For example, Dumais and Chen [7] and Liu et al. [22] adopt a top-down training strategy and use SVMs to distinguish child categories of the same parent. In contrast, bottom-up classification [1] backpropagates the labels from the leaves to the top layer. Gopal and Yang [12, 14] propose a recursive regularization framework encouraging the similarity between the child classifiers and the parent classifier. Peng et al. [35] extend this regularization to deep graph-CNN models. Wehrmann et al. [44] and Huang et al. [15] combine the ideas of training a local classifier per level and optimizing the global classification results to mitigate exposure bias. The global structure of hierarchies is also used in many other models, such as meta-learning [46], reinforcement learning [24] and tree/graph based neural networks [55]. However, these fully supervised methods rely heavily on the amount of human-labeled training data. Under weakly supervised settings, hierarchical dataless classification [36] jointly embeds class labels and documents using Explicit Semantic Analysis; WeSHClass [30] models topic semantics in the word2vec embedding space and applies a self-training scheme; PCEM [48] introduces a path-cost sensitive classifier for semi-supervised hierarchical classification; HierCon [21] projects documents and taxonomy categories into a common concept space and calculates their fine-grained similarity. However, these studies are not concerned with metadata information.

Metadata-Aware Text Classification. Several previous studies try to incorporate metadata information into a text classifier. For example, Ghani et al. [11] use HTML meta tags to help hypertext classification; Steyvers et al. [37] leverage author information in topic classification; Tang et al. [39] learn user and product representations for sentiment analysis; Zhang et al. [53] employ user biography data for tweet localization. Chen et al. [4] encode user mobility data and social network information for joint time and location prediction; Kim et al. [16] propose a general framework to inject categorical metadata signal into a deep text classifier. However, these models assume a fully supervised setting. Zhang et al. [54] present a weakly supervised hierarchical classification approach to classify GitHub repositories. Later, they also propose a flat metadata-aware text categorization framework [52]. Mekala et al. [28] explore to incorporate metadata as additional supervision for text classification with seed words only. However, in these studies, the label hierarchy is not leveraged in the embedding space.

Tree and Metadata Embedding. Recent studies on non-Euclidean embedding models, such as Poincaré [33], Lorentz [34], hyperbolic cones [10] and spherical tree embedding [31], consider to preserve a tree structure in the embedding space. Along another line of work, heterogeneous network embedding algorithms [6, 9, 50] are widely used to learn representations of metadata. HHNE [43] further considers to perform heterogeneous network embedding in a hyperbolic space. Despite their *respective* success in capturing hierarchy and metadata information, to the best of our knowledge, there lacks a framework which allows *simultaneous* modeling of tree-structure dependencies, text semantics and metadata signals.

6 CONCLUSIONS

We present HiMeCat, an embedding-based generative framework for hierarchical metadata-aware document categorization under weak supervision. The framework is featured by a joint hierarchy-metadata-text representation learning module and a hierarchical data augmentation module. We propose a generative process in the spherical space to guide the design of both modules. Through experiments on three datasets from different domains, we show the superiority of HiMeCat towards competitive baselines in our task. We also conduct ablation studies to validate the contribution of our proposed embedding and augmentation modules. Interesting future work include: (1) discovering new categories from the unlabeled dataset and put them into the existing hierarchy and (2) integrating

different forms of weak supervision (e.g., annotated documents and class-related keywords) in hierarchical text classification.

ACKNOWLEDGMENTS

Research was sponsored in part by US DARPA KAIROS Program No. FA8750-19-2-1004 and SocialSim Program No. W911NF-17-C-0099, National Science Foundation IIS-19-56151, IIS-17-41317, IIS 17-04532, and IIS 16-18481, and DTRA HDTRA11810026. Any opinions, findings, and conclusions 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. The U.S. Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright annotation hereon.

REFERENCES

[1] Paul N Bennett and Nam Nguyen. 2009. Refined experts: improving classification in large taxonomies. In *SIGIR'09*. 11–18.
[2] Silvere Bonnabel. 2013. Stochastic gradient descent on Riemannian manifolds. *IEEE Trans. Automat. Control* 58, 9 (2013), 2217–2229.
[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS'13*. 2787–2795.
[4] Yile Chen, Cheng Long, Gao Cong, and Chenliang Li. 2020. Context-aware deep model for joint mobility and time prediction. In *WSDM'20*. 106–114.
[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT'19*. 4171–4186.
[6] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *KDD'17*. 135–144.
[7] Susan Dumais and Hao Chen. 2000. Hierarchical classification of web content. In *SIGIR'00*. 256–263.
[8] Ronald Aylmer Fisher. 1953. Dispersion on a sphere. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 217, 1130 (1953), 295–305.
[9] Tao-yang Fu, Wang-Chien Lee, and Zhen Lei. 2017. Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. In *CIKM'17*. 1797–1806.
[10] Octavian Ganea, Gary Becigneul, and Thomas Hofmann. 2018. Hyperbolic Entailment Cones for Learning Hierarchical Embeddings. In *ICML'18*. 1646–1655.
[11] Rayid Ghani, Seán Slattery, and Yiming Yang. 2001. Hypertext categorization using hyperlink patterns and meta data. In *ICML'01*. 178–185.
[12] Siddharth Gopal and Yiming Yang. 2013. Recursive regularization for large-scale classification with hierarchical and graphical dependencies. In *KDD'13*. 257–265.
[13] Siddharth Gopal and Yiming Yang. 2014. Von mises-fisher clustering models. In *ICML'14*. 154–162.
[14] Siddharth Gopal and Yiming Yang. 2015. Hierarchical bayesian inference and recursive regularization for large-scale classification. *TKDD* 9, 3 (2015), 1–23.
[15] 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'19*. 1051–1060.
[16] Jihyeok Kim, Reinald Kim Amplayo, Kyungjae Lee, Sua Sung, Minji Seo, and Seung-won Hwang. 2019. Categorical Metadata Representation for Customized Text Classification. *TACL* 7 (2019), 201–215.
[17] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *EMNLP'14*. 1746–1751.
[18] Daphne Koller and Mehran Sahami. 1997. Hierarchically Classifying Documents Using Very Few Words. In *ICML'97*. 170–178.
[19] Sachin Kumar and Yulia Tsvetkov. 2019. Von mises-fisher loss for training sequence to sequence models with continuous outputs. In *ICLR'19*.
[20] Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *TACL* 3 (2015), 211–225.
[21] 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'19*. 379–388.
[22] Tie-Yan Liu, Yiming Yang, Hao Wan, Hua-Jun Zeng, Zheng Chen, and Wei-Ying Ma. 2005. Support vector machines classification with a very large-scale taxonomy. *ACM SIGKDD Explorations Newsletter* 7, 1 (2005), 36–43.
[23] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, Nov (2008), 2579–2605.
[24] Yuning Mao, Jingjing Tian, Jiawei Han, and Xiang Ren. 2019. Hierarchical Text Classification with Reinforced Label Assignment. In *EMNLP'19*. 445–455.
[25] Yuning Mao, Tong Zhao, Andrey Kan, Chenwei Zhang, Xin Luna Dong, Christos Faloutsos, and Jiawei Han. 2020. Octet: Online Catalog Taxonomy Enrichment with Self-Supervision. In *KDD'20*.
[26] Kanti V Mardia and Peter E Jupp. 2009. *Directional statistics*. Vol. 494. John Wiley & Sons.
[27] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys'13*. 165–172.
[28] Dheeraj Mekala, Xinyang Zhang, and Jingbo Shang. 2020. META: Metadata-Empowered Weak Supervision for Text Classification. In *EMNLP'20*.
[29] Yu Meng, Jiaxin Huang, Guangyuan Wang, Chao Zhang, Honglei Zhuang, Lance Kaplan, and Jiawei Han. 2019. Spherical text embedding. In *NeurIPS'19*. 8208–8217.
[30] Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. 2019. Weakly-supervised hierarchical text classification. In *AAAI'19*. 6826–6833.
[31] Yu Meng, Yunyi Zhang, Jiaxin Huang, Yu Zhang, Chao Zhang, and Jiawei Han. 2020. Hierarchical topic mining via joint spherical tree and text embedding. In *KDD'20*. 1908–1917.
[32] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS'13*. 3111–3119.
[33] Maximillian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. In *NIPS'17*. 6338–6347.
[34] Maximillian Nickel and Douwe Kiela. 2018. Learning Continuous Hierarchies in the Lorentz Model of Hyperbolic Geometry. In *ICML'18*. 3779–3788.
[35] Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, and Qiang Yang. 2018. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *WWW'18*. 1063–1072.
[36] Yangqiu Song and Dan Roth. 2014. On dataless hierarchical text classification. In *AAAI'14*. 1579–1585.
[37] Mark Steyvers, Padhraic Smyth, Michal Rosen-Zvi, and Thomas Griffiths. 2004. Probabilistic author-topic models for information discovery. In *KDD'04*. 306–315.
[38] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune BERT for text classification?. In *CCL'19*. 194–206.
[39] Duyu Tang, Bing Qin, and Ting Liu. 2015. Learning semantic representations of users and products for document level sentiment classification. In *ACL'15*. 1014–1023.
[40] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *WWW'15*. 1067–1077.
[41] Chenglong Wang, Feijun Jiang, and Hongxia Yang. 2017. A hybrid framework for text modeling with convolutional rnn. In *KDD'17*. 2061–2069.
[42] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE TKDE* 29, 12 (2017), 2724–2743.
[43] Xiao Wang, Yiding Zhang, and Chuan Shi. 2019. Hyperbolic heterogeneous information network embedding. In *AAAI'19*. 5337–5344.
[44] Jonatas Wehrmann, Ricardo Cerri, and Rodrigo Barros. 2018. Hierarchical multi-label classification networks. In *ICML'18*. 5075–5084.
[45] Richard C Wilson, Edwin R Hancock, Elżbieta Pekalska, and Robert PW Duin. 2014. Spherical and hyperbolic embeddings of data. *IEEE TPAMI* 36, 11 (2014), 2255–2269.
[46] Jiawei Wu, Wenhan Xiong, and William Yang Wang. 2019. Learning to Learn and Predict: A Meta-Learning Approach for Multi-Label Classification. In *EMNLP'19*. 4345–4355.
[47] Mingfang Wu, Michael Fuller, and Ross Wilkinson. 2001. Using clustering and classification approaches in interactive retrieval. *Information Processing & Management* 37, 3 (2001), 459–484.
[48] Huiru Xiao, Xin Liu, and Yangqiu Song. 2019. Efficient Path Prediction for Semi-Supervised and Weakly Supervised Hierarchical Text Classification. In *WWW'19*. 3370–3376.
[49] Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *NAACL-HLT'15*. 1006–1011.
[50] Carl Yang, Yuxin Xiao, Yu Zhang, Yizhou Sun, and Jiawei Han. 2020. Heterogeneous Network Representation Learning: A Unified Framework with Survey and Benchmark. *arXiv preprint arXiv:2004.00216* (2020).
[51] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *NAACL'16*. 1480–1489.
[52] Yu Zhang, Yu Meng, Jiaxin Huang, Frank F. Xu, Xuan Wang, and Jiawei Han. 2020. Minimally Supervised Categorization of Text with Metadata. In *SIGIR'20*. 1231–1240.
[53] Yu Zhang, Wei Wei, Binxuan Huang, Kathleen M Carley, and Yan Zhang. 2017. RATE: Overcoming Noise and Sparsity of Textual Features in Real-Time Location Estimation. In *CIKM'17*. 2423–2426.
[54] Yu Zhang, Frank F. Xu, Sha Li, Yu Meng, Xuan Wang, Qi Li, and Jiawei Han. 2019. HiGitClass: Keyword-Driven Hierarchical Classification of GitHub Repositories. In *ICDM'19*. 876–885.
[55] Jie Zhou, Chunping Ma, Dingkun Long, Guangwei Xu, Ning Ding, Haoyu Zhang, Pengjun Xie, and Gongshen Liu. 2020. Hierarchy-Aware Global Model for Hierarchical Text Classification. In *ACL'20*. 1106–1117.