

HIGITCLASS: Keyword-Driven Hierarchical Classification of GitHub Repositories

Yu Zhang¹, Frank F. Xu², Sha Li¹, Yu Meng¹, Xuan Wang¹, Qi Li³, Jiawei Han¹

¹Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA

²Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA, USA

³Department of Computer Science, Iowa State University, Ames, IA, USA

{yuz9, shal2, yumeng5, xwang174, hanj}@illinois.edu, frankxu@cmu.edu, qli@iastate.edu

Abstract—GitHub has become an important platform for code sharing and scientific exchange. With the massive number of repositories available, there is a pressing need for topic-based search. Even though the topic label functionality has been introduced, the majority of GitHub repositories do not have any labels, impeding the utility of search and topic-based analysis. This work targets the automatic repository classification problem as *keyword-driven hierarchical classification*. Specifically, users only need to provide a label hierarchy with keywords to supply as supervision. This setting is flexible, adaptive to the users’ needs, accounts for the different granularity of topic labels and requires minimal human effort. We identify three key challenges of this problem, namely (1) the presence of multi-modal signals; (2) supervision scarcity and bias; (3) supervision format mismatch. In recognition of these challenges, we propose the HIGITCLASS framework, comprising of three modules: heterogeneous information network embedding; keyword enrichment; topic modeling and pseudo document generation. Experimental results on two GitHub repository collections confirm that HIGITCLASS is superior to existing weakly-supervised and dataless hierarchical classification methods, especially in its ability to integrate both structured and unstructured data for repository classification.

Index Terms—hierarchical classification, GitHub, weakly-supervised learning

I. INTRODUCTION

For the computer science field, code repositories are an indispensable part of the knowledge dissemination process, containing valuable details for reproduction. For software engineers, sharing code also promotes the adoption of best practices and accelerates code development. The needs of the scientific community and that of software developers have facilitated the growth of online code collaboration platforms, the most popular of which is GitHub, with over 96 million repositories and 31 million users as of 2018. With the overwhelming number of repositories hosted on GitHub, there is a natural need to enable search functionality so that users can quickly target repositories of interest. To accommodate this need, GitHub introduced topic labels¹ which allowed users to declare topics for their own repositories. However, topic-based search on GitHub is still far from ideal. For example, when searching for repositories related to “phylogenetics”, a highly relevant repository opentreel² with many stars and forks does not even show up in the first 10 pages of search

results as it does not contain the “phylogenetics” tag. Hence, to improve the search and analysis of GitHub repositories, a critical first step is *automatic repository classification*.

In the process of examining the automatic repository classification task, we identify three different cases of missing labels: (1) *Missing annotation*: the majority of repositories (73% in our MACHINE-LEARNING dataset and 78% in our BIOINFORMATICS dataset) have no topic labels at all; (2) *Incomplete annotation*: since topic labels can be arbitrarily general or specific, some repositories may miss coarse-grained labels while others miss fine-grained ones; (3) *Evolving label space*: related GitHub topics tags may not have existed at the time of creation, so the label is naturally missing. Missing annotation is the major drive behind automatic classification, but this also implies that labeled data is scarce and expensive to obtain. Incomplete annotation reflects the hierarchical relationship between labels: repositories should not only be assigned to labels of one level of granularity, but correspond to a path in the class hierarchy. Finally, the evolving label space requires the classification algorithm to quickly adapt to a new label space, or take the label space as part of the input. Combining these observations, we define our task as *keyword-driven hierarchical classification* for GitHub repositories. By keyword-driven, we imply that we are performing classification using only a few keywords as supervision.

Compared to the common setting of fully-supervised classification of text documents, *keyword-driven hierarchical classification* of GitHub repositories poses unique challenges. First of all, GitHub repositories are complex objects with metadata, user interaction and textual description. As a result, multi-modal signals can be utilized for topic classification, including user ownership information, existing tags and README text. To jointly model structured and unstructured data, we propose to construct a *heterogeneous information network* (HIN) centered upon words. By learning node embeddings in this HIN, we obtain word representations that reflect the co-occurrence of multi-modal signals that are unique to the GitHub repository dataset. We also face the supervision scarcity and bias problem as users only provide one keyword for each class as guidance. This single keyword may reflect user’s partial knowledge of the class and may not achieve good coverage of the class distribution. In face of this challenge, we introduce a *keyword enrichment* module that expands the single keyword to a

¹<https://help.github.com/en/articles/about-topics>

²<https://github.com/OpenTreeOfLife/opentreel>

keyword *set* for each category. The newly selected keywords are required to be close to the target class in the embedding space. Meanwhile, we keep mutual exclusivity among keyword sets so as to create a clear separation boundary. Finally, while users provide a label hierarchy, the classification algorithm ultimately operates on repositories, so there is a mismatch in the form of supervision. Since we already encode the structured information through the HIN embeddings, in our final classification stage we represent each repository as a document. To transform keywords into documents, we first model each class as a topic distribution over words and estimate the distribution parameters. Then based on the topic distributions, we follow a two-step procedure to generate pseudo documents for training. This also allows us to employ powerful classifiers such as CNNs for classification, which would not be possible with the scarce labels.

To summarize, we have the following contributions:

- We present the task of keyword-driven hierarchical classification of GitHub repositories. While GitHub has been of widespread interest to the research community, no previous efforts have been devoted to the task of automatically assigning topic labels to repositories, which can greatly facilitate repository search and analysis. To deal with the evolving hierarchical label space and circumvent expensive annotation efforts, we only rely on the user-provided label hierarchy and keywords to train the classifier.
- We design the HiGITCLASS framework, which consists of three modules: HIN construction and embedding; keyword enrichment; topic modeling and pseudo document generation. The three modules are carefully devised to overcome three identified challenges of our problem: the presence of multi-modal signals; supervision scarcity and bias; supervision format mismatch.
- We collect two datasets of GitHub repositories from the machine learning and bioinformatics research community. On both datasets we show that our proposed framework HiGITCLASS outperforms existing supervised and semi-supervised models for hierarchical classification.

The remainder of this paper is organized as follows. In Section II, we formally introduce our problem definition, multi-modal signals in a GitHub repository and heterogeneous information networks. In Section III, we elaborate our framework HiGITCLASS with its three components. Then in Section IV, we present experimental results and discuss our findings. Section V covers related literature and we conclude in Section VI.

II. PRELIMINARIES

A. Problem Definition

We study hierarchical classification of GitHub repositories where the categories form a tree structure. Traditional hierarchical classification approaches [7], [19] rely on a large set of labeled training documents. In contrast, to tackle the evolving label space and alleviate annotation efforts, we formulate our task as *keyword-driven* classification, where users just need to provide the label hierarchy and *one* keyword for each *leaf* category. This bears some similarities with the dataless

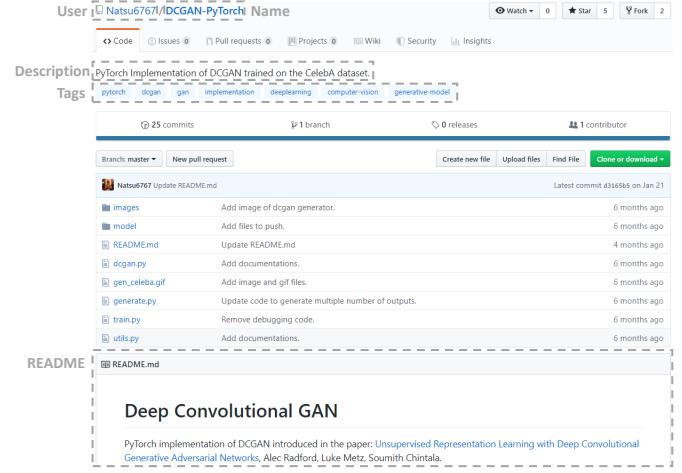


Fig. 1. A sample GitHub repository with the user's name, the repository name, description tags, and README (only the first paragraph is shown).

classification proposed in [33] that utilizes an user-defined label hierarchy and class descriptions. There is no requirement of *any* labeled repository. Formally, our task is defined as follows.

Problem Definition. (KEYWORD-DRIVEN HIERARCHICAL CLASSIFICATION.) *Given a collection of unlabeled GitHub repositories, a tree-structured label hierarchy \mathcal{T} and one keyword w_{i0} for each leaf class C_i ($i = 1, \dots, \mathcal{L}$), our task is to assign appropriate category labels to the repositories, where the labels can be either a leaf or an internal node in \mathcal{T} .*

B. GitHub Repositories

Fig. 1 shows a sample GitHub repository³. With the help of GitHub API⁴, we are able to extract comprehensive information of a repository including metadata, source code and team dynamics. In HiGITCLASS, we utilize the following information:

User. Users usually have consistent interests and skills. If two repositories share the same user ("Natsu6767" in Fig. 1), they are more likely to have similar topics (e.g., deep learning or image generation).

Name. If two repositories share the same name, it is likely that one is forked from the other and they should belong to the same topic category. Besides, indicative keywords can be obtained by segmenting the repository name properly (e.g., "DCGAN" and "PyTorch" in Fig. 1).

Description. The description is a concise summary of the repository. It usually contains topic-indicating words (e.g., "DCGAN" and "CelebA" in Fig. 1).

Tags. Although a large proportion of GitHub repositories are not tagged, when available, tags are strong indicators of a repository's topic (e.g., "dcgan" and "generative-model" in Fig. 1).

³<https://github.com/Natsu6767/DCGAN-PyTorch>

⁴<https://developer.github.com/v3/>

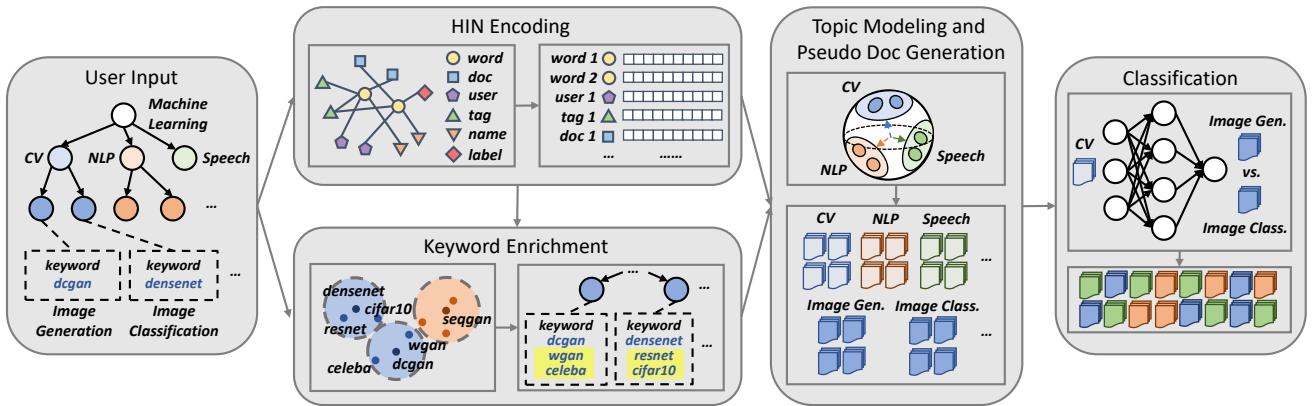


Fig. 2. The HiGITCLASS framework. Three key modules (i.e., HIN encoding, keyword enrichment, and pseudo document generation) are used to tackle the aforementioned three challenges, respectively.

README. The README file is the main source of textual information in a repository. In contrast to the description, it elaborates more on the topic but may also diverge to other issues (e.g., installation processes and code usages). The latter introduces noises to the task of topic inference.

We concatenate the description and README fields into a single **Document** field, which serves as the textual feature of a repository.

C. Heterogeneous Information Networks

In our proposed framework HiGITCLASS, we model the multi-modal signals of GitHub repositories as a heterogeneous information network (HIN) [34], [35]. HINs are an extension of homogeneous information networks to support multiple node types and edge types. We formally define a heterogeneous information network as below:

Heterogeneous Information Network (HIN). An HIN is defined as a graph $G = (\mathcal{V}, \mathcal{E})$ with a node type mapping $\phi : \mathcal{V} \rightarrow \mathcal{T}_{\mathcal{V}}$ and an edge type mapping $\psi : \mathcal{E} \rightarrow \mathcal{T}_{\mathcal{E}}$. Either the number of node types $|\mathcal{T}_{\mathcal{V}}|$ or the number of relation types $|\mathcal{T}_{\mathcal{E}}|$ is larger than 1.

As we all know, one advantage of networks is the ability to go beyond direct links and model higher-order relationships which can be captured by paths between nodes. We introduce the notion of meta-paths, which account for different edge types in HINs.

Meta-Path. In an HIN, meta-paths [35] are an abstraction of paths proposed to describe multi-hop relationships. For an HIN $G = (\mathcal{V}, \mathcal{E})$, a meta-path is a sequence of edge types $\mathcal{M} = E_1 \cdot E_2 \dots \cdot E_L$ ($E_i \in \mathcal{T}_{\mathcal{E}}$). Any path that has the same types as the meta-path is an instance of the meta-path. When edge types are a function of the node types, we also represent a meta-path as $\mathcal{M} = V_1 \cdot V_2 \dots \cdot V_L$ ($V_i \in \mathcal{T}_{\mathcal{V}}$ and $V_i \cdot V_{i+1} \in \mathcal{T}_{\mathcal{E}}$ for any i).

III. METHOD

We lay out our HiGITCLASS framework in Fig. 2. HiGITCLASS consists of three key modules, which are proposed to solve the three challenges mentioned in Introduction, respectively.

To deal with *multi-modal signals*, we propose an *HIN encoding* module (Section III-A). Given the label hierarchy and keywords, we first construct an HIN to characterize different kinds of connections between words, documents, users, tags, repository names and labels. Then we adopt ESIM [31], a meta-path guided heterogeneous network embedding technique, to obtain good node representations.

To tackle *supervision scarcity and bias*, we introduce a *keyword enrichment* module (Section III-B). This module expands the user-specified keyword to a semantically concentrated keyword set for each category. The enriched keywords are required to share high proximity with the user-given one from the view of embeddings. Meanwhile, we keep mutual exclusivity among keyword sets so as to create a clear separation boundary.

To overcome *supervision format mismatch*, we present a *pseudo-document generation* technique (Section III-C). We first model each class as a topic distribution over words and estimate the distribution parameters. Then based on the topic distributions, we follow a two-step procedure to generate pseudo documents for training. This step allows us to employ powerful classifiers such as *convolutional neural network* [16]. Intuitively, the neural classifier is fitting the learned word distributions instead of a small set of keywords, which can effectively prevent it from overfitting.

A. HIN Construction and Embedding

HIN Construction. The first step of our model is to construct an HIN that can capture all the interactions between different types of information regarding GitHub repositories. We include six types of nodes: words (W), documents (D), users (U), tags (T), tokens segmented from repository names (N) and labels (L). There is a one-to-one mapping between documents (D) and repositories (R), thus document nodes may also serve as a representation of its corresponding repository in the network. Since the goal of this module is to learn accurate word representations for the subsequent classification step, we adopt a *word-centric star schema* [36], [37]. The schema is shown in Fig. 3(a). We use a sample ego network of the word “DCGAN” to help illustrate our schema (Fig. 3(b)). The word

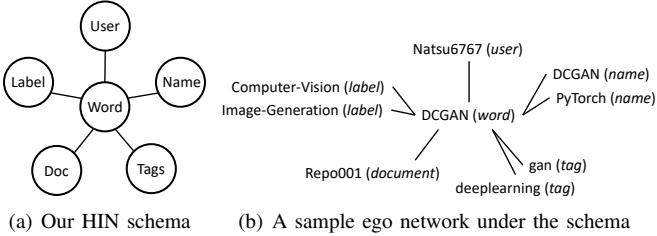


Fig. 3. Our HIN schema and a sample network under the schema. The five edge types characterize different kinds of second-order proximity between words.

vocabulary is the union of words present in the documents, tags, segmented repository names and user-provided keywords.

Following the star schema, we then have 5 types of edges in the HIN that represent 5 types of word co-occurrences:

(1) $W-D$. The *word-document* edges describe document-level co-occurrences, where the edge weight between word w_i and document d_j indicates the number of times w_i appears in d_j (i.e., term frequency, or $tf(w_i, d_j)$). From the perspective of *second-order proximity* [38], $W-D$ edges reflect the fact that two words tend to have similar semantics when they appear in the same repository’s document.

(2) $W-U$. We add an edge between a word and a user node if the user is the owner of a repository that contains the word in its document field. The edge weight between word w_i and user u_j is the sum of the term frequency of the word w in each of the user’s repositories:

$$\sum_{k: \text{document } d_k \text{ belongs to user } u_j} tf(w_i, d_k).$$

(3) $W-T$. The *word-tag* relations encode tag-level word co-occurrences. The edge weight between word w_i and tag t_j is

$$\sum_{k: \text{document } d_k \text{ has tag } t_j} tf(w_i, d_k).$$

(4) $W-N$. We segment the repository name using “-”, “_” and whitespace as separators. For example, we obtain two tokens “DCGAN” and “PyTorch” by segmenting the repository name “DCGAN-PyTorch” in Fig. 1. The edge weight between word w_i and name token n_j also defined through term frequency:

$$\sum_{k: \text{document } d_k \text{ has name token } n_j} tf(w_i, d_k).$$

(5) $W-L$. The *word-label* relations describe category-level word co-occurrences. Only user-provided keywords will be linked with label nodes and its parents. For example, if we select “DCGAN” as the keyword of a (leaf) category “\$IMAGE-GENERATION”, “DCGAN” will have links to “\$IMAGE-GENERATION” and all of its parent categories (e.g., “\$COMPUTER-VISION”).

HIN Embedding. Once we have an HIN, we proceed to learn representations for nodes in the network. Then the embedding vectors of word nodes can be applied for repository topic classification.

There are many popular choices for network representation learning on HINs, such as METAPATH2VEC [6] and HIN2VEC [9]. We adopt ESIM [31] as our HIN embedding algorithm as it achieves the best performance in our task. (We will validate this choice in Section IV-C.) Random walk based methods [11], [27] have enjoyed success in learning node embeddings on homogeneous graphs in a scalable manner. ESIM adapts the idea for use on HINs, and restricts the random walk under guidance of user-specified meta-paths. In HiGITCLASS, we choose $W-D-W$, $W-U-W$, $W-T-W$, $W-N-W$ and $W-L-W$ as our meta-paths, modeling the five different types of second-order proximity between words.

Following the selected meta-paths, we can sample a large number of meta-path instances in our HIN (e.g., $W-D-W$ is a valid node sequence, while $D-W-D$ is not). Given a meta-path \mathcal{M} and its corresponding node sequence $\mathcal{P} = u_1-u_2-\dots-u_l$, we assume that the probability of observing a path given a meta-path constraint follows that of a first-order Markov chain:

$$\Pr(\mathcal{P}|\mathcal{M}) = \Pr(u_1|\mathcal{M}) \prod_{i=1}^{l-1} \Pr(u_{i+1}|u_i, \mathcal{M}),$$

where

$$\Pr(v|u, \mathcal{M}) = \frac{\exp(f(u, v, \mathcal{M}))}{\sum_{v' \in V} \exp(f(u, v', \mathcal{M}))} \quad (1)$$

and

$$f(u, v, \mathcal{M}) = \mu_{\mathcal{M}} + p_{\mathcal{M}}^T e_u + q_{\mathcal{M}}^T e_v + e_u^T e_v.$$

Here, $\mu_{\mathcal{M}}$ is the global bias of meta-path \mathcal{M} . $p_{\mathcal{M}}$ and $q_{\mathcal{M}}$ are d -dimensional local bias of \mathcal{M} . e_u and e_v are d -dimensional embedding vectors of nodes u and v , respectively. e_u , e_v , $p_{\mathcal{M}}$, $q_{\mathcal{M}}$ and $\mu_{\mathcal{M}}$ can be learned through maximizing the likelihood.

However, the denominator in Equation (1) requires summing over all nodes, which is very computationally expensive given the large network size. In our actual computation, we estimate this term through negative sampling [24].

$$\Pr(v|u, \mathcal{M}) = \frac{\exp(f(u, v, \mathcal{M}))}{\sum_{v' \in V^-} \exp(f(u, v', \mathcal{M})) + \exp(f(u, v, \mathcal{M}))},$$

where V^- is the set of nodes that serve as negative samples.

B. Keyword Enrichment

Since we only ask users to provide *one* keyword for each category, in case of scarcity and bias, we devise a keyword enrichment module to automatically expand the single keyword w_{i0} to a keyword set $\mathcal{K}_i = \{w_{i0}, w_{i1}, \dots, w_{iK_i}\}$ so as to better capture the semantics of the category.

From the HIN embedding step, we have obtained the embedding vector e_w for each word w . We perform normalization so that all embedding vectors reside on the unit sphere (i.e., $e_w \leftarrow e_w / \|e_w\|$). Then the inner product of two embedding vectors $e_{w_1}^T e_{w_2}$ is adopted to characterize the proximity between two words w_1 and w_2 . For each class C_i , we add words sharing the highest proximity with w_{i0} into its enriched keyword set. Meanwhile, to create a clear

Algorithm 1 KEYWORDENRICH($w_{10}, \dots, w_{\mathcal{L}0}$)

```

1:  $\mathcal{K}_i = \{w_{i0}\}, i = 1, \dots, \mathcal{L}$ 
2:  $w_{i, \text{last}} = w_{i0}, i = 1, \dots, \mathcal{L}$ 
3: while  $\mathcal{K}_i \cap \mathcal{K}_j = \emptyset (\forall i, j)$  do
4:   for  $i = 1$  to  $\mathcal{L}$  do
5:      $w_{i, \text{last}} = \arg \max_{w \notin \mathcal{K}_i} e_{w_{0i}}^T e_w$ 
6:      $\mathcal{K}_i = \mathcal{K}_i \cup \{w_{i, \text{last}}\}$ 
7:   end for
8: end while
9:  $\mathcal{K}_i = \mathcal{K}_i / \{w_{i, \text{last}}\}, i = 1, \dots, \mathcal{L}$  //Remove the last added
   keyword to keep mutual exclusivity
10: output  $\mathcal{K}_1, \dots, \mathcal{K}_{\mathcal{L}}$ 

```

separation boundary between categories, we require $\mathcal{K}_1, \dots, \mathcal{K}_{\mathcal{L}}$ to be *mutually exclusive*. Therefore, the expansion process terminates when any two of the keyword sets tend to intersect. Algorithm 1 describes the process.

Note that on a unit sphere, the inner product is a reverse measure of the spherical distance between two points. Therefore, we are essentially expanding the keyword set with the nearest neighbors of the given keyword. The termination condition is that two “neighborhoods” have overlaps.

C. Topic Modeling and Pseudo Document Generation

To leverage keywords for classification, we face two problems: (1) a typical classifier needs labeled repositories as input; (2) although the keyword sets have been enriched, a classifier will likely be overfitted if it is trained solely on these keywords. To tackle these issues, we assume we can generate a training document \tilde{d} for class C_i given \mathcal{K}_i through the following process:

$$q(\tilde{d}|C) = q(\tilde{d}|\Theta_i)p(\Theta_i|\mathcal{K}_i).$$

Here $q(\cdot|\Theta_i)$ is the topic distribution of C_i parameterized by Θ_i , with which we can “smooth” the small set of keywords \mathcal{K}_i to a continuous distribution. For simplicity, we adopt a “bag-of-words” model for the generated documents, so $q(\tilde{d}|\Theta_i) = \prod_{i=0}^{|\tilde{d}|} q(w_i|\Theta_i)$. Then we draw samples of words from $q(\cdot|\Theta_i)$ to form a pseudo document following the technique proposed in [22].

Spherical Topic Modeling. Given the normalized embeddings, we characterize the word distribution for each category using a mixture of von Mises-Fisher (vMF) distributions [1], [10]. To be specific, the probability to generate keyword w from category C_i is defined as

$$q(w|C_i) = \sum_{j=1}^m \alpha_j f(e_w|\mu_j, \kappa_j) = \sum_{j=1}^m \alpha_j c_p(\kappa_j) \exp(\kappa_j \mu_j^T e_w),$$

where $f(e_w|\mu_j, \kappa_j)$, as a vMF distribution, is the j -th component in the mixture with a weight α_j . The vMF distribution can be interpreted as a normal distribution confined to a unit sphere. It has two parameters: the mean direction vector μ_j and the concentration parameter κ_j . The keyword embeddings

concentrate around μ_j , and are more concentrated if κ_j is large. $c_p(\kappa_j)$ is a normalization constant.

Following [23], we choose the number of vMF components differently for leaf and internal categories: (1) For a *leaf* category C_j , the number of components m is set to 1 and the mixture model degenerates to a single vMF distribution. (2) For an *internal* category C_j , we set the number of components to be the number of C_j ’s children in the label hierarchy.

Given the enriched keyword set \mathcal{K}_j , we can derive μ_j and κ_j using Expectation Maximization (EM) [1]. Recall that the keywords of an internal category are aggregated from its children categories. In practice, we use the approximation procedure based on Newton’s method [1] to derive κ_j .

Pseudo Document Generation. To generate a pseudo document \tilde{d} for C_j , we first sample a document vector $e_{\tilde{d}}$ from $f(\cdot|C_j)$. Then we build a local vocabulary $V_{\tilde{d}}$ that contains top- τ words similar with \tilde{d} in the embedding space. ($\tau = 50$ in our model.) Given $V_{\tilde{d}}$, we repeatedly generate a number of words from a background distribution with probability β and from the document-specific distribution with probability $1 - \beta$. Formally,

$$\Pr(w|\tilde{d}) = \begin{cases} \beta p_B(w), & w \notin V_{\tilde{d}} \\ \beta p_B(w) + (1 - \beta) \frac{\exp(e_w^T e_{\tilde{d}})}{\sum_{w' \in V_{\tilde{d}}} \exp(e_{w'}^T e_{\tilde{d}})}, & w \in V_{\tilde{d}} \end{cases}$$

where $p_B(w)$ is the background distribution (i.e., word distribution in the entire corpus).

Here we generate the pseudo document in two steps: first sampling the document vector and then sampling words from a mixture of the document language model and a background language model. Compared to directly sampling words from $f(\cdot|C_j)$, the two-step process ensures better coverage of the class distribution. In direct sampling, with high probability we will include words that are close to the centroid of the topic C_j . The classifier may learn to ignore all other words and use only these words to determine the predicted class. By first sampling the document vector, we would like to lead the classifier to learn that all documents that fall within the topic distribution belong to the same class.

The synthesized pseudo documents are then used as training data for a classifier. In HiGITCLASS, we adopt convolutional neural networks (CNN) [16] for the classification task. One can refer to [16], [22] for more details of the network architecture. The embedding vectors of word nodes obtained by ESIM in the previous HIN module are used as pre-trained embeddings.

Recall the process of generating pseudo documents, if we evenly split the fraction of the background distribution into the m children categories of C_j , the “true” label distribution (an m -dimensional vector) of a pseudo document \tilde{d} can be defined as

$$\text{label}(\tilde{d})_i = \begin{cases} (1 - \beta) + \beta/m, & \tilde{d} \text{ is generated from child } i \\ \beta/m, & \text{otherwise} \end{cases}$$

Since our label is a distribution instead of a one-hot vector, we compute the loss as the KL divergence between the output

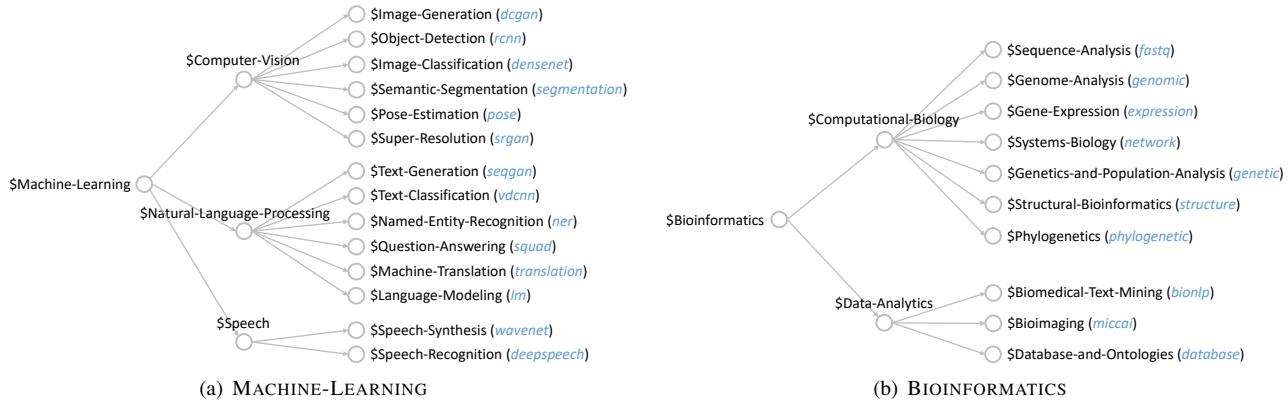


Fig. 4. Label hierarchy and provided keywords (in blue) on the two datasets.

label distribution and the pseudo label.

IV. EXPERIMENTS

We aim to answer two questions in our experiments. First, does HiGITCLASS achieve supreme performance in comparison with various baselines (Section IV-B)? Second, we propose three key modules in HiGITCLASS. How do they contribute to the overall performance? (The effects of these three modules will be explored one by one in Sections IV-C, IV-D and IV-E).

A. Experimental Setup

Datasets. We collect two datasets of GitHub repositories covering different domains.⁵ Their statistics are summarized in Table I.

TABLE I
DATASET STATISTICS.

Dataset	#Repos	#Classes (Level 1 + Level 2)
MACHINE-LEARNING	1,596	3 + 14
BIOINFORMATICS	876	2 + 10

- **MACHINE-LEARNING.** This dataset is collected by the Paper With Code project⁶. It contains a list of GitHub repositories implementing state-of-the-art algorithms of various machine learning tasks, where the tasks are organized as a taxonomy.
- **BIOINFORMATICS.** This dataset is extracted from research articles published on four venues *Bioinformatics*, *BioNLP*, *MICCAI* and *Database* from 2014 to 2018. In each article, authors may put a code link, and we extract the links pointing to a GitHub repository. Meanwhile, each article has an issue section, which is viewed as the topic label of the associated repository.

Note that more than 73% (resp., 78%) of the repositories in our MACHINE-LEARNING (resp., BIOINFORMATICS) dataset have no tags.

Baselines. We evaluate the performance of HiGITCLASS against the following hierarchical classification algorithms:

⁵Our code and datasets are available at <https://github.com/yuzhimanhua/HiGitClass>.

⁶<https://paperswithcode.com/media/about/evaluation-tables.json.gz>

- **HierSVM** [7] decomposes the training tasks according to the label taxonomy, where each local SVM is trained to distinguish sibling categories that share the same parent node.⁷
- **HierDataless** [33] embeds both class labels and documents in a semantic space using Explicit Semantic Analysis on Wikipedia articles, and assigns the nearest label to each document in the semantic space.⁸ Note that HierDataless uses Wikipedia as **external knowledge** in classification, whereas other baselines and HiGITCLASS solely reply on user-provided data.
- **WeSTClass** [22] first generates pseudo documents and then trains a CNN based on the synthesized training data.⁹
- **WeSHClass** [23] leverages a language model to generate synthesized data for pre-training and then iteratively refines the global hierarchical model on labeled documents.¹⁰
- **PCNB** [41] utilizes a path-generated probabilistic framework on the label hierarchy and trains a path-cost sensitive naive Bayes classifier.¹¹
- **PCEM** [41] makes use of the unlabeled data to ameliorate the path-cost sensitive classifier and applies an EM technique for semi-supervised learning.

Note that HierSVM, PCNB and PCEM can only take document-level supervision (i.e., labeled repositories). To align the experimental settings, we first label all the repositories using TFIDF scores by treating the keyword set of each class as a query. Then, we select top-ranked repositories per class as the supervision to train HierSVM, PCNB and PCEM. Since the baselines are all text classification approaches, we append the information of user, tags and repository name to the end of the document for each repository so that the baselines can exploit these signals.

⁷<https://github.com/globality-corp/sklearn-hierarchical-classification>

⁸<https://github.com/yqsong/DatalessClassification>

⁹<https://github.com/yumeng5/WeSTClass>

¹⁰<https://github.com/yumeng5/WeSHClass>

¹¹<https://github.com/HKUST-KnowComp/PathPredictionForTextClassification>

TABLE II
PERFORMANCE OF COMPARED ALGORITHMS ON THE MACHINE-LEARNING DATASET. HIERDATALESS DOES NOT HAVE A STANDARD DEVIATION SINCE IT IS A DETERMINISTIC ALGORITHM.

Method	Level-1 Micro	Level-1 Macro	Level-2 Micro	Level-2 Macro	Overall Micro	Overall Macro
HierSVM [7]	54.20 \pm 4.53	46.58 \pm 3.59	27.40 \pm 3.55	31.99 \pm 5.34	40.80 \pm 1.20	34.57 \pm 3.94
HierDataless [33]	76.63	33.44	13.72	8.80	45.18	13.15
WeSTClass [22]	61.78 \pm 3.90	48.00 \pm 2.04	37.71 \pm 2.72	34.34 \pm 1.70	49.75 \pm 3.24	36.75 \pm 1.67
WeSHClass [23]	70.69 \pm 2.14	52.73 \pm 2.18	38.08 \pm 2.07	33.87 \pm 2.23	54.39 \pm 2.11	37.60 \pm 1.67
PCNB [41]	77.79 \pm 1.92	62.53 \pm 2.55	26.77 \pm 3.89	22.85 \pm 1.98	52.28 \pm 1.92	29.85 \pm 1.74
PCEM [41]	77.28 \pm 2.00	59.27 \pm 2.29	22.34 \pm 4.02	19.28 \pm 2.25	49.81 \pm 2.00	26.34 \pm 1.83
HiGITCLASS w/o HIN	75.28 \pm 6.99	60.85 \pm 5.51	40.21 \pm 2.91	39.77 \pm 2.22	57.74 \pm 4.07	43.49 \pm 2.10
HiGITCLASS w/o ENRICH	86.48 \pm 1.41	72.19 \pm 2.53	36.71 \pm 1.13	43.75 \pm 2.85	61.60 \pm 0.25	48.77 \pm 1.94
HiGITCLASS w/o HIER	57.31 \pm 1.63	59.30 \pm 3.14	40.45 \pm 2.51	44.41 \pm 2.66	48.88 \pm 2.07	47.04 \pm 1.63
HiGITCLASS	87.68 \pm 2.23	72.06 \pm 4.39	43.93 \pm 2.93	45.97 \pm 2.29	65.81 \pm 3.55	50.57 \pm 2.98

TABLE III
PERFORMANCE OF COMPARED ALGORITHMS ON THE BIOINFORMATICS DATASET. HIERDATALESS DOES NOT HAVE A STANDARD DEVIATION SINCE IT IS A DETERMINISTIC ALGORITHM.

Method	Level-1 Micro	Level-1 Macro	Level-2 Micro	Level-2 Macro	Overall Micro	Overall Macro
HierSVM [7]	80.39 \pm 2.72	70.16 \pm 6.05	13.49 \pm 10.2	10.04 \pm 9.07	46.94 \pm 4.66	20.06 \pm 7.12
HierDataless [33]	81.39	78.75	39.27	36.57	60.33	43.60
WeSTClass [22]	61.78 \pm 5.75	52.73 \pm 4.86	20.39 \pm 3.48	17.52 \pm 2.59	41.09 \pm 4.28	23.91 \pm 2.82
WeSHClass [23]	63.17 \pm 3.56	59.65 \pm 3.51	26.44 \pm 1.33	24.94 \pm 0.98	44.80 \pm 2.26	30.72 \pm 1.30
PCNB [41]	77.03 \pm 2.89	59.43 \pm 3.51	31.77 \pm 3.80	22.90 \pm 4.84	54.40 \pm 2.89	28.99 \pm 4.82
PCEM [41]	78.51 \pm 3.06	61.99 \pm 4.18	32.80 \pm 2.88	18.93 \pm 5.28	55.66 \pm 3.06	26.11 \pm 5.41
HiGITCLASS w/o HIN	66.21 \pm 8.33	64.39 \pm 7.00	31.87 \pm 3.65	30.47 \pm 3.15	49.04 \pm 5.75	36.13 \pm 3.52
HiGITCLASS w/o ENRICH	54.55 \pm 10.0	53.57 \pm 9.15	22.95 \pm 1.46	23.18 \pm 1.89	38.74 \pm 4.30	28.24 \pm 0.87
HiGITCLASS w/o HIER	78.79 \pm 2.57	73.71 \pm 2.28	39.42 \pm 4.47	41.58 \pm 2.63	59.10 \pm 3.51	46.94 \pm 2.48
HiGITCLASS	81.71 \pm 3.95	77.11 \pm 3.89	42.44 \pm 8.46	41.67 \pm 8.04	62.08 \pm 6.11	47.57 \pm 7.34

Besides the baselines, we also include the following three ablation versions of HiGITCLASS into comparison.

- **w/o HIN** skips the HIN embedding module and relies on word2vec [24] to generate word embeddings for the following steps.
- **w/o Enrich** skips the keyword enrichment module and directly uses one single keyword in spherical topic modeling.
- **w/o Hier** directly classifies all repositories to the leaf layer and then assigns internal labels to each repository according to its leaf category.

Evaluation Metrics. We use F1 scores to evaluate the performance of all methods. Denote TP_i , FP_i and, FN_i as the instance numbers of true-positive, false-positive and false negative for category C_i . Let \mathcal{T}_1 (resp., \mathcal{T}_2) be the set of all Level-1 (resp., Level-2/leaf) categories. The Level-1 Micro-F1 is defined as $\frac{2PR}{P+R}$, where $P = \frac{\sum_{C_i \in \mathcal{T}_1} TP_i}{\sum_{C_i \in \mathcal{T}_1} (TP_i + FP_i)}$ and $R = \frac{\sum_{C_i \in \mathcal{T}_1} TP_i}{\sum_{C_i \in \mathcal{T}_1} (TP_i + FN_i)}$. The Level-1 Macro-F1 is defined as $\frac{1}{|\mathcal{T}_1|} \sum_{C_i \in \mathcal{T}_1} \frac{2P_i R_i}{P_i + R_i}$, where $P_i = \frac{TP_i}{TP_i + FP_i}$ and $R_i = \frac{TP_i}{TP_i + FN_i}$. Accordingly, Level-2 Micro/Macro-F1 and Overall Micro/Macro-F1 can be defined on \mathcal{T}_2 and $\mathcal{T}_1 \cup \mathcal{T}_2$.

B. Performance Comparison with Baselines

Tables II and III demonstrate the performance of compared methods on two datasets. We repeat each experiment 5 times (except HierDataless, which is a deterministic algorithm) with the mean and standard deviation reported.

As we can observe from Tables II and III, on both datasets, a significant improvement is achieved by HiGITCLASS com-

pared to the baselines. On MACHINE-LEARNING, HiGITCLASS notably outperforms the second best approach by 22.1% on average. On BIOINFORMATICS, the only metric in terms of which HiGITCLASS does not perform the best is Level-1 Macro-F1, and the main opponent of HiGITCLASS is HierDataless. As mentioned above, HierDataless incorporates Wikipedia articles as its external knowledge. When user-provided keywords can be linked to Wikipedia (e.g., “genomic”, “genetic” and “phylogenetic” in BIOINFORMATICS), HierDataless can exploit the external information well. However, when the keywords cannot be wikified (e.g., names of new deep learning algorithms such as “dcgan”, “rcnn” and “densenet” in MACHINE-LEARNING), the help from Wikipedia is limited. In fact, on MACHINE-LEARNING, HierDataless performs poorly. Note that on GitHub, it is common that names of recent algorithms or frameworks are provided as keywords.

Besides outperforming baseline approaches, HiGITCLASS shows a consistent and evident improvement against three ablation versions. The average boost of the HIN module over the six metrics is 15.0% (resp., 28.6%) on the MACHINE-LEARNING (resp., BIOINFORMATICS) dataset, indicating the importance of encoding multi-modal signals on GitHub. The Level-1 F1 scores of HiGITCLASS w/o ENRICH is close to the full model on MACHINE-LEARNING, but the keyword enrichment module demonstrates its power when we go deeper. This finding is aligned with the fact that the topic distributions of coarse-grained categories are naturally distant from each other on the sphere. Therefore, one keyword per category may be enough to estimate the distributions approximately.

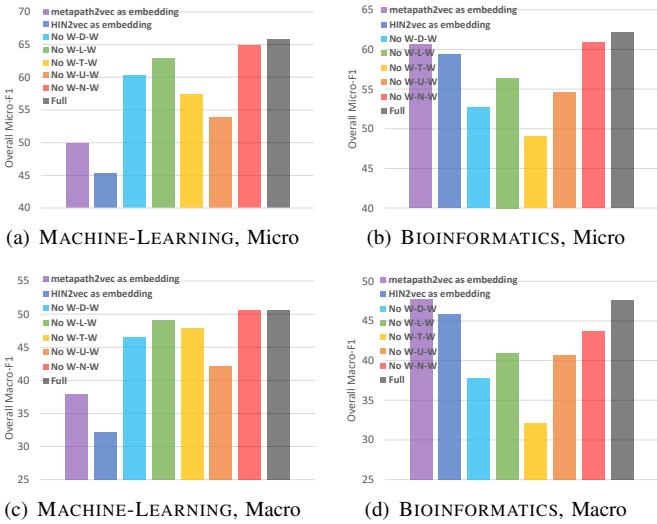


Fig. 5. Performance of algorithms with different HIN modules.

However, when we aim at fine-grained classification, the topic distributions become closer, and the inference process may be easily interfered by a biased keyword. HiGITCLASS w/o HIER performs poorly on MACHINE-LEARNING, which highlights the importance of utilizing the label hierarchy during the training process. The same phenomenon occurs in the comparison between WeSTClass and WeSHClass.

C. Effect of HIN Construction and Embedding

We have demonstrated the contribution of our HIN module by comparing HiGITCLASS and HiGITCLASS w/o HIN. To explore the effectiveness of HIN construction and embedding in a more detailed way, we perform an ablation study by changing one “factor” in the HIN and fixing all the other modules in HiGITCLASS. To be specific, our HIN has five types of edges, each of which corresponds to a meta-path. We consider five ablation versions (**No W-D-W**, **No W-U-W**, **No W-T-W**, **No W-N-W** and **No W-L-W**). Each version ignores one edge type/meta-path. Moreover, given the complete HIN, we consider to use two popular approaches, METAPATH2VEC [6] and HIN2VEC [9], as our embedding technique, which generates two variants **metapath2vec as embedding** and **HIN2vec as embedding**. Fig. 5 shows the performance of these variants and our **Full** model.

We have the following findings from Fig. 5. First, our **Full** model outperforms the five ablation models ignoring different edge types, indicating that each meta-path (as well as each node type incorporated in the HIN construction step) plays a positive role in classification. Second, our **Full** model outperforms METAPATH2VEC AS EMBEDDING and HIN2VEC AS EMBEDDING in most cases, which validates our choice of using ESIM as the embedding technique. The possible reason that ESIM is more suitable for our task may be that we have a simple word-centric star schema and a clear goal of embedding word nodes. Therefore, the choices of meta-paths can be explicitly specified (i.e., $W\text{-}?\text{-}W$) and do not need to be inferred from data (as HIN2VEC does). Third, among

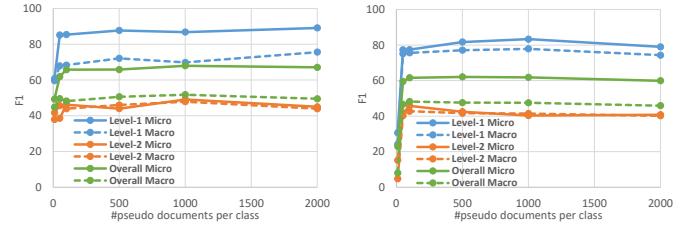


Fig. 6. Performance of HiGITCLASS with different numbers of pseudo documents.

the five ablation models ignoring edge types, **No W-U-W** performs the worst on MACHINE-LEARNING, which means $W\text{-}U$ edges (i.e., the user information) contribute the most in repository classification. Meanwhile, on BIOINFORMATICS, $W\text{-}T$ edges have the largest offering. This can be explained by the following statistics: in the MACHINE-LEARNING dataset, 348 pairs of repositories share the same user, out of which 217 (62%) have the same *leaf* label; in the BIOINFORMATICS dataset, there are 356 pairs of repositories having at least two overlapping tags, among which 221 (62%) belong to the same *leaf* category. Fourth, $W\text{-}D$ edges also contribute a lot to the performance. This observation is aligned with the results in [37], where document-level word co-occurrences play a crucial role in text classification.

D. Effect of Keyword Enrichment

Quantitatively, the keyword enrichment module has a positive contribution to the whole framework according to previous experiments. We now show its effect qualitatively. Table IV lists top-5 words selected by HiGITCLASS during keyword enrichment. Besides topic-indicating words (e.g., “tagger”, “trait”, “trees”, etc.), popular algorithm/tool names (e.g., “bidaf” and “pymol”), dataset names (e.g., “mpi”, “pdb”) and author names (e.g., “papandreou” and “lample”) are also included in the expanded keyword set. Note that some provided keywords are more or less ambiguous (e.g., “segmentation” and “structure”), and directly using them for topic modeling may introduce noises. In contrast, the expanded set as a whole can better characterize the semantics of each topic category.

E. Effect of Pseudo Documents

In all previous experiments, when we build a classifier for an internal category C_i , we generate 500 pseudo documents for each child of C_i . What if we use less/more synthesized training data? Intuitively, if the amount of generated pseudo documents is too small, signals in previous modules cannot fully propagate to the training process. On the contrary, if we have too many generated data, the training time will be unnecessarily long. To see whether 500 is a good choice, we plot the performance of HiGITCLASS with 10, 50, 100, 500, 1000 and 2000 pseudo documents in Fig. 6.

On the one side, when the number of pseudo documents is too small (e.g., 10, 50 or 100), information carried in the synthesized training data will be insufficient to train a good classifier. On the other side, when we generate too

TABLE IV
ENTITY ENRICHMENT RESULTS ON THE TWO DATASETS. FOUR LEAF CATEGORIES ARE SHOWN FOR EACH DATASET.

Class	\$SEMANTIC-SEGMENTATION	\$POSE-ESTIMATION	\$NAMED-ENTITY-RECOGNITION	\$QUESTION-ANSWERING
Keyword	segmentation	pose	ner	squad
Enriched Keywords	semantic	estimation	entity	question
	papandreou	person	tagger	answering
	scene	human	lample	bidaf
	pixel	mpii	-	-
	segment	3d	-	-
Class	\$GENE-EXPRESSION	\$GENETICS-AND-POPULATION	\$STRUCTURAL-BIOINFORMATICS	\$PHYLOGENETICS
Keyword	expression	genetic	structure	phylogenetic
Enriched Keywords	gene	traits	protein	trees
	genes	trait	pdb	newick
	rna	markers	residues	phylogenetics
	cell	phenotypes	pymol	phylogenies
	isoform	associations	residue	evolution

many pseudo documents (e.g, 1000 or 2000), putting efficiency aside, the performance is not guaranteed to increase. In fact, on both datasets, the F1 scores start to fluctuate when the number of pseudo documents becomes large. In our task, generating 500 to 1000 pseudo documents for each class will strike a good balance.

V. RELATED WORK

GitHub Repository Mining. As a popular code collaboration community, GitHub presents many opportunities for researchers to learn how people write code and design tools to support the process. As a result, GitHub data has received attention from both software engineering and social computing researchers. Analytic studies [5], [15], [21], [29], [39], [40] have investigated how user activities (e.g., collaboration, following and watching) affect development practice. Algorithmic studies [8], [28], [32], [44] exploit README files and repository metadata to perform data mining tasks such as similarity search [44] and clustering [32]. In this paper, we focus on the task of automatically classifying repositories whereas previous works [15], [29] have relied on human effort to annotate each repository with its topic.

HIN Embeddings. Many node embeddings techniques have been proposed for HIN, including [6], [9], [31], [42]. From the application point of view, typical applications of learned embeddings include node classification [6], [9], [31], [42], node clustering [6] and link prediction [9], [42]. Several studies apply HIN node embeddings into downstream classification tasks, such as malware detection [13] and medical diagnosis [12]. Different from the fully-supervised settings in [12], [13], our repository classification task relies on a very small set of guidance. Moreover, most information used in [12], [13] is structured. In contrast, we combine structured information such as user-repository ownership relation with unstructured text for classification.

Dataless Text Classification. Although deep neural architectures [14], [16], [43] demonstrate their advantages in fully-supervised text classification, their requirement of massive training data prohibits them from being adopted in some practical scenarios. Under weakly-supervised or dataless settings,

there have been solutions following two directions: *latent variable models* extending topic models (e.g., PLSA and LDA) by incorporating user-provided seed information [4], [17], [18], [20] and *embedding-based models* deriving vectorized representations for words and documents [3], [22], [37]. There are also some work on semi-supervised text classification [25], [30], but they require a set of labeled documents instead of keywords.

Hierarchical Text Classification. Under *fully-supervised* settings, [7] and [19] first propose to train SVMs to distinguish the children classes of the same parent node. [2] further defines hierarchical loss function and applies cost-sensitive learning to generalize SVM learning for hierarchical classification. [26] proposes a graph-CNN based model to convert text to graph-of-words, on which the graph convolution operations are applied for feature extraction. Under *weakly-supervised* or *dataless* settings, previous approaches include HierDataless [33], WeSHClass [23] and PCNB/PCEM [41], which have been introduced in Section IV-A. All above mentioned studies focus on text data without additional information. In HiGITCLASS, we are able to go beyond plain text classification and utilize multi-modal signals.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we have studied the problem of *keyword-driven hierarchical* classification of GitHub repositories: the end user only needs to provide a label hierarchy and one keyword for each leaf category. Given such scarce supervision, we design HiGITCLASS with three modules: heterogeneous information network embedding; keyword enrichment; pseudo document generation. Specifically, HIN embeddings take advantage of the multi-modal nature of GitHub repositories; keyword enrichment alleviates the supervision scarcity of each category; pseudo document generation transforms the keywords into documents, enabling the use of powerful neural classifiers. Through experiments on two repository collections, we show that HiGITCLASS consistently outperforms all baselines by a large margin, particularly on the lower levels of the hierarchy. Further analysis shows that the HIN module contributes the most to the boost in performance and keyword

enrichment demonstrates its power deeper down the hierarchy where the differences between classes become more subtle.

For future work, we would like to explore the possibility of integrating different forms of supervision (e.g., the combination of keywords and labeled repositories). The HIN embedding module may also be coupled more tightly with the document classification process by allowing the document classifier’s prediction results to propagate along the network.

ACKNOWLEDGMENT

We thank Xiaotao Gu for useful discussions. The research was sponsored in part by U.S. Army Research Lab. under Cooperative Agreement No. W911NF-09-2-0053 (NSCTA), DARPA under Agreement No. W911NF-17-C0099, National Science Foundation IIS 16-18481, IIS 17-04532, and IIS-17-41317, DTRA HDTRA11810026, and grant 1U54GM114838 awarded by NIGMS through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative (www.bd2k.nih.gov). The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies. We thank anonymous reviewers for valuable and insightful feedback.

REFERENCES

- [1] A. Banerjee, I. S. Dhillon, J. Ghosh, and S. Sra. Clustering on the unit hypersphere using von mises-fisher distributions. *Journal of Machine Learning Research*, 6(Sep):1345–1382, 2005.
- [2] L. Cai and T. Hofmann. Hierarchical document categorization with support vector machines. In *CIKM’04*, pages 78–87, 2004.
- [3] M.-W. Chang, L.-A. Ratinov, D. Roth, and V. Srikumar. Importance of semantic representation: Dataless classification. In *AAAI’08*, pages 830–835, 2008.
- [4] X. Chen, Y. Xia, P. Jin, and J. Carroll. Dataless text classification with descriptive lda. In *AAAI’15*, 2015.
- [5] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb. Social coding in github: transparency and collaboration in an open software repository. In *CSCW’12*, pages 1277–1286, 2012.
- [6] Y. Dong, N. V. Chawla, and A. Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *KDD’17*, pages 135–144, 2017.
- [7] S. Dumais and H. Chen. Hierarchical classification of web content. In *SIGIR’00*, pages 256–263, 2000.
- [8] Y. Fan, Y. Zhang, S. Hou, L. Chen, Y. Ye, C. Shi, L. Zhao, and S. Xu. idev: Enhancing social coding security by cross-platform user identification between github and stack overflow. In *IJCAI’19*, pages 2272–2278, 2019.
- [9] T.-y. Fu, W.-C. Lee, and Z. Lei. Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. In *CIKM’17*, pages 1797–1806, 2017.
- [10] S. Gopal and Y. Yang. Von mises-fisher clustering models. In *ICML’14*, pages 154–162, 2014.
- [11] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *KDD’16*, pages 855–864, 2016.
- [12] A. Hosseini, T. Chen, W. Wu, Y. Sun, and M. Sarrafzadeh. Heteromed: Heterogeneous information network for medical diagnosis. In *CIKM’18*, pages 763–772, 2018.
- [13] S. Hou, Y. Ye, Y. Song, and M. Abdulhayoglu. Hindroid: An intelligent android malware detection system based on structured heterogeneous information network. In *KDD’17*, pages 1507–1515, 2017.
- [14] J. Howard and S. Ruder. Universal language model fine-tuning for text classification. In *ACL’18*, pages 328–339, 2018.
- [15] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian. The promises and perils of mining github. In *MSR’14*, pages 92–101, 2014.
- [16] Y. Kim. Convolutional neural networks for sentence classification. In *EMNLP’14*, pages 1746–1751, 2014.
- [17] C. Li, J. Xing, A. Sun, and Z. Ma. Effective document labeling with very few seed words: A topic model approach. In *CIKM’16*, pages 85–94, 2016.
- [18] X. Li, C. Li, J. Chi, J. Ouyang, and C. Li. Dataless text classification: A topic modeling approach with document manifold. In *CIKM’18*, pages 973–982, 2018.
- [19] T.-Y. Liu, Y. Yang, H. Wan, H.-J. Zeng, Z. Chen, and W.-Y. Ma. Support vector machines classification with a very large-scale taxonomy. *SIGKDD Explorations*, 7(1):36–43, 2005.
- [20] Y. Lu and C. Zhai. Opinion integration through semi-supervised topic modeling. In *WWW’08*, pages 121–130, 2008.
- [21] W. Ma, L. Chen, X. Zhang, Y. Zhou, and B. Xu. How do developers fix cross-project correlated bugs? a case study on the github scientific python ecosystem. In *ICSE’17*, pages 381–392, 2017.
- [22] Y. Meng, J. Shen, C. Zhang, and J. Han. Weakly-supervised neural text classification. In *CIKM’18*, pages 983–992, 2018.
- [23] Y. Meng, J. Shen, C. Zhang, and J. Han. Weakly-supervised hierarchical text classification. In *AAAI’19*, pages 6826–6833, 2019.
- [24] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS’13*, pages 3111–3119, 2013.
- [25] T. Miyato, A. M. Dai, and I. J. Goodfellow. Adversarial training methods for semi-supervised text classification. In *ICLR’16*, 2016.
- [26] H. Peng, J. Li, Y. He, Y. Liu, M. Bao, L. Wang, Y. Song, and Q. Yang. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *WWW’18*, pages 1063–1072, 2018.
- [27] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *KDD’14*, pages 701–710, 2014.
- [28] G. A. A. Prana, C. Treude, F. Thung, T. Atapattu, and D. Lo. Categorizing the content of github readme files. *Empirical Software Engineering*, pages 1–32, 2018.
- [29] P. H. Russell, R. L. Johnson, S. Ananthan, B. Harnke, and N. E. Carlson. A large-scale analysis of bioinformatics code on github. *PLOS One*, 13(10):e0205898, 2018.
- [30] D. S. Sachan, M. Zaheer, and R. Salakhutdinov. Revisiting lstm networks for semi-supervised text classification via mixed objective function. In *AAAI’19*, pages 6940–6948, 2019.
- [31] J. Shang, M. Qu, J. Liu, L. M. Kaplan, J. Han, and J. Peng. Metapath guided embedding for similarity search in large-scale heterogeneous information networks. *arXiv preprint arXiv:1610.09769*, 2016.
- [32] A. Sharma, F. Thung, P. S. Kochhar, A. Sulistya, and D. Lo. Cataloging github repositories. In *EASE’17*, pages 314–319, 2017.
- [33] Y. Song and D. Roth. On dataless hierarchical text classification. In *AAAI’14*, pages 1579–1585, 2014.
- [34] Y. Sun and J. Han. Mining heterogeneous information networks: principles and methodologies. *Synthesis Lectures on Data Mining and Knowledge Discovery*, 3(2):1–159, 2012.
- [35] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *PVLDB*, 4(11):992–1003, 2011.
- [36] Y. Sun, Y. Yu, and J. Han. Ranking-based clustering of heterogeneous information networks with star network schema. In *KDD’09*, pages 797–806. ACM, 2009.
- [37] J. Tang, M. Qu, and Q. Mei. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *KDD’15*, pages 1165–1174, 2015.
- [38] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *WWW’15*, pages 1067–1077, 2015.
- [39] J. Tsay, L. Dabbish, and J. Herbsleb. Influence of social and technical factors for evaluating contribution in github. In *ICSE’14*, pages 356–366. ACM, 2014.
- [40] J. T. Tsay, L. Dabbish, and J. Herbsleb. Social media and success in open source projects. In *CSCW’12*, pages 223–226, 2012.
- [41] H. Xiao, X. Liu, and Y. Song. Efficient path prediction for semi-supervised and weakly supervised hierarchical text classification. In *WWW’19*, pages 3370–3376, 2019.
- [42] C. Yang, Y. Feng, P. Li, Y. Shi, and J. Han. Meta-graph based hin spectral embedding: Methods, analyses, and insights. In *ICDM’18*, pages 657–666, 2018.
- [43] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy. Hierarchical attention networks for document classification. In *NAACL’16*, pages 1480–1489, 2016.
- [44] Y. Zhang, D. Lo, P. S. Kochhar, X. Xia, Q. Li, and J. Sun. Detecting similar repositories on github. In *SANER’17*, pages 13–23, 2017.