MLI: A Multi-level Inference Mechanism for User Attributes in Social Networks

HANG ZHANG and YAJUN YANG, State Key Laboratory of Communication Content Cognition, People's Daily Online, Beijing 100733, China, College of Intelligence and Computing, Tianjin University, China
XIN WANG, College of Intelligence and Computing, Tianjin University, China
HONG GAO, Faculty of Computing, Harbin Institute of Technology, China
QINGHUA HU, College of Intelligence and Computing, Tianjin University, China

In the social network, each user has attributes for self-description called user attributes, which are semantically hierarchical. Attribute inference has become an essential way for social platforms to realize user classifications and targeted recommendations. Most existing approaches mainly focus on the flat inference problem neglecting the semantic hierarchy of user attributes, which will cause serious inconsistency in multi-level tasks. In this article, we propose a multi-level model MLI, where information propagation part collects attribute information by mining the global graph structure, and the attribute correction part realizes the mutual correction between different levels of attributes. Further, we put forward the concept of generalized semantic tree, a way of representing the hierarchical structure of user attributes, whose nodes are allowed to have multiple parent nodes unlike the regular tree. Both regular and generalized semantic trees are commonly used in practice, and can be handled by our model. Besides, by making the inference start from sub-networks with sufficient attribute information, we design a "Ripple" algorithm to improve the efficiency and effectiveness of our model. For evaluation purposes, we conduct extensive verification experiments on DBLP datasets. The experimental results show the superior effect of MLI, compared with the state-of-the-art methods.

CCS Concepts: • **Information systems → Data mining**; **Social tagging systems;**

Additional Key Words and Phrases: Attribute inference, hierarchical inference, social network

ACM Reference format:
Hang Zhang, Yajun Yang, Xin Wang, Hong Gao, and Qinghua Hu. 2022. MLI: A Multi-level Inference Mechanism for User Attributes in Social Networks. *ACM Trans. Inf. Syst.* 41, 2, Article 39 (December 2022), 30 pages.
https://doi.org/10.1145/3545797

This work is supported by National Key Research and Development Program of China (Grant No. 2019YFB2101903), State Key Laboratory of Communication Content Cognition (Grant No. A32003), National Natural Science Foundation of China (Grant No. 61972275).
Authors' addresses: H. Zhang and Y. Yang (corresponding author), State Key Laboratory of Communication Content Cognition, People's Daily Online, Beijing 100733, China, College of Intelligence and Computing, Tianjin University, Tianjin, China; emails: {aronzhang, yjyang}@tju.edu.cn; X. Wang and Q. Hu, College of Intelligence and Computing, Tianjin University, Tianjin, China; emails: {wangx, huqinghua}@tju.edu.cn; H. Gao, Faculty of Computing, Harbin Institute of Technology, Harbin, China; email: honggao@hit.edu.cn.

1046-8188/2022/12-ART39 $15.00
https://doi.org/10.1145/3545797

1 INTRODUCTION

With the development of the Internet, graphs have been widely used to model complex interactions among real-world entities like social network. In the social network, users are usually with attributes to indicate their hobbies, occupations or other aspects. However, due to the limitation of information collection and the existence of noise, attributes in the social network are always incomplete, only a fraction of users are attached with attributes and the remaining ones are unknown. Therefore, inferring users' attributes is of great significance for many applications, such as personalized services and product recommendations. Existing works assume all attributes are at a single level, while in real social networks user attributes are semantically organized in a hierarchical structure. For example, in an academic social network, users' research fields are organized with hierarchies. For a researcher in the computer field, the first-level attribute can be computer science, which is a broad representation. The second-level attribute is machine learning, a subfield of computer science. The third-level attribute can be computer vision, which is a more specific research field in machine learning. From top to bottom, it is a gradual refinement of the research fields, and the lower-level attributes are the fine-grained representations of the upper-level ones. Inferring attributes for users of each level is regarded as multi-level inference problem.

Inferring multi-level attributes can make the inference results more accurate and analyze users more effectively. However, most existing methods [5, 9, 21, 22, 27, 46] neglect the relationship implicit in the attribute hierarchy and cannot work well for real social networks when users have multi-level attributes. When the existing single-level methods are straightforwardly applied for the attribute inference in each level of the hierarchy, the results will have three problems: ***conflict***, ***indeterminacy***, and ***missing***.

In Figure 1, with a real social network DBLP, we illustrate the above three problems that arise when directly applying single-level methods to real applications with multi-level semantic attributes. DBLP is a database system of English essays in the computer field, where each user is a vertex and co-authors are connected by an edge. The research field as the user attribute to be inferred has a four-level structure, for example Zhanpeng Jin's research field is expressed as "CS-Network-Wireless-Localization" from coarse-grained to fine-grained. Figure 1 shows a part of the social network, where red users' attributes are unknown, and the remaining six are known users. We take Mohammad Pourhomayoun, Mark L. Fowler, and Chikahito Nakajima as examples; Figure 1 shows their inference results obtained by applying the single-level method [25] and our method MLI. In addition, we use blue nodes in Figure 2 to denote the inference results of these three users in the hierarchy. For the problem of conflict, as can be seen from Figure 2(a), even though utilizing the same method for every single-level, the inference results of different levels may be conflicted. In Mohammad's inference results, "CS," "Network," and "Wireless" are on the same path of the hierarchy, but the result of the fourth level is "Recognition," which should be classified under the category of "Learning-Image" according to the hierarchical structure, resulting in a semantic contradiction. For the problem of indeterminacy, as shown in Figure 2(b), since the distribution of "Bandwidth" and "Localization" around Mark L. Fowler is similar, it is hard for the single-level method to determine which one is the most suitable. For the problem of missing, in Figure 2(c), "Learning" is Chikahito Nakajima's second-level attribute; however, due to the lack of information about this level around him, it is impossible to obtain "Learning" only by single-level inference method.

As discussed in Figures 1 and 2, directly applying existing single-level methods is not appropriate for real applications, when user attributes are semantically hierarchical. To overcome the problems of conflict, indeterminacy and missing shown by the example in Figures 1 and 2, we propose a multi-level method to jointly correct the inference results at different levels, which can improve the inference effect. For the conflict problem in Figure 2(a), "Recognition" can be

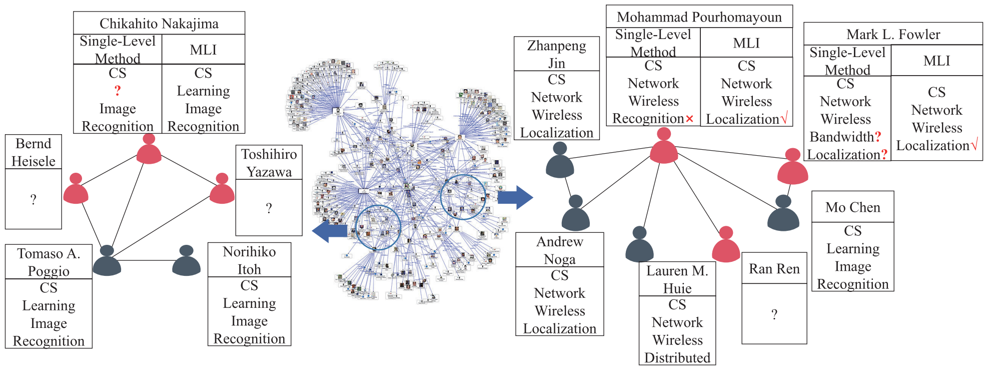

Chikahito Nakajima
Single-Level Method
MLI
CS ? Image Recognition
CS Learning Image Recognition
Bernd Heisele
?
Toshihiro Yazawa
?
Tomaso A. Poggio
CS Learning Image Recognition
Norihiko Itoh
CS Learning Image Recognition
Zhanpeng Jin
CS Network Wireless Localization
Mohammad Pourhomayoun
Single-Level Method
MLI
CS Network Wireless Recognition×
CS Network Wireless Localization√
Mark L. Fowler
Single-Level Method
MLI
CS Network Wireless Bandwidth? Localization?
CS Network Wireless Localization√
Andrew Noga
CS Network Wireless Localization
Lauren M. Huie
CS Network Wireless Distributed
Ran Ren
?
Mo Chen
CS Learning Image Recognition

Fig. 1. A real multi-level inference example in DBLP.

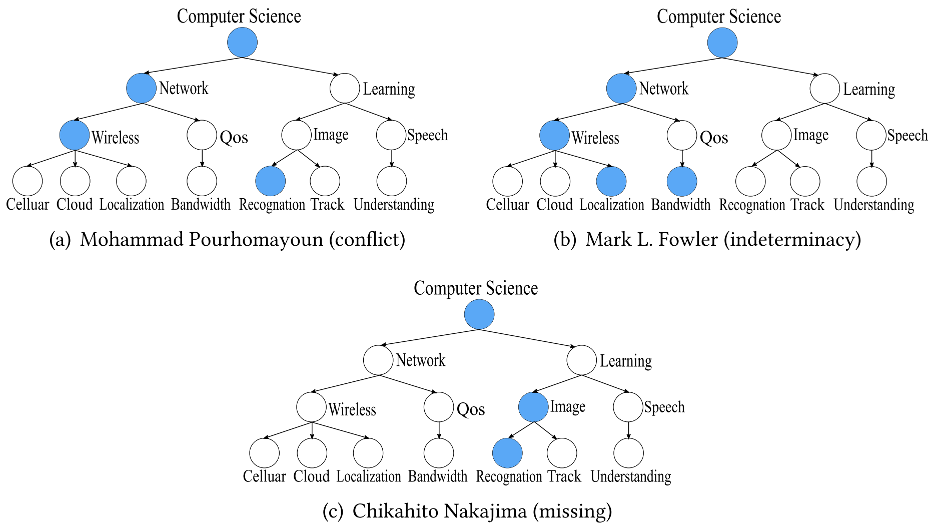


Fig. 2. Inference results of three unknown users in Figure 1 by single-level method.

corrected to "Localization" through the above three levels "CS-Network-Wireless." For the indeterminacy problem of the fourth level in Figure 2(b), since the third-level attribute can be determined as "Wireless" with a high probability, "Localization" is more likely to be Mark L. Fowler's attribute than "Bandwidth." For the problem of missing the second-level results in Figure 2(c), the third- and fourth-level results can help us infer "Learning" and complete the inference results.

Chakrabarti et al. [6] take the relationship between attributes into consideration, however it is still a flat structure rather than a hierarchy. Although Ariyaratne and Barbedo et al. [1, 2] discover the hierarchical nature of user attributes and propose a bottom-up solution, they just simply predict the attributes of the bottom level. Once the inference results of bottom level are wrong, the upper-level inference will also get wrong results, so this method cannot be applied to real datasets. Currently, the most researched areas of multi-level attributes are text classification [20, 37] and image classification [16, 41]. The input features are often text or RGB information, which is not suitable for our social network scenarios. To the best of our knowledge, this is the first time to consider multi-level attributes in social networks and try to solve them by considering both graph structures and the semantic hierarchical relationships among user attributes.

In this work, we propose a novel **Multi-level Inference (MLI)** model. This model contains two parts: information propagation and attribute correction. We use random walk based on maximum entropy theory to model the spread of user attribute information in social networks, manifested in the fact that information flows to users with high entropy values. We design a confidence model to define the credibility of users in social networks. Users with high credibility spread more information during information propagation, to improve the accuracy of inference. Attribute correction is to explore the relationship between attributes to correct the inference results. Thus, MLI can make accurate inferences on attributes with hierarchical structures.

The main contributions of our work can be summarized as follows:

- We study the task of multi-level attribute inference in the social network. We propose a multi-level inference model called MLI. This model can infer hierarchical attributes for unknown users by collecting attributes from nearby users under maximum entropy random walk. Meanwhile, we propose a correction method based on the predefined hierarchical structure to revise the results. In some application contexts, user attributes cannot be represented by a regular tree, so we propose a generalized semantic tree to describe the hierarchical relationship between attributes, so that our model can be applied to a wider range of scenarios.
- We propose a "Ripple" algorithm to make the inference start from sub-networks with sufficient attribute information and enable vertices who have obtained enough credible information to exit the iteration early, which can accelerate the inference and improve the effect.
- We design a series of experiments to evaluate the effectiveness of our proposed model on real data sets. The experimental results demonstrate the superior performance of our method.

The rest of this article is organized as follows: Section 2 introduces the related work. Section 3 defines the problem. Section 4 details our MLI model, including information propagation, attribute correction, confidence calculation, and generalized semantic tree. Section 5 describes our "Ripple" algorithm. Section 6 introduces our experiment setup. The experimental results and analysis are presented in Section 7, and we conclude the article in Section 8.

2 RELATED WORK

2.1 Flat Inference

Over the past several years, there has been an increasing interest in user attribute inference. Among them, the most concerning problem is when the attributes are at the same level, which is called flat inference. Methods for flat inference can be grouped into three categories, namely, content-based, graph structure-based, and machine learning-based.

Content-based methods infer users' attributes based on their text content in the social network such as posted blogs [4, 24], tweets [5, 8, 28, 45], and query logs [13]. Nowson et al. [24] reveal a number of features inside the blog contents and used SVM to infer authors' gender. Cheng et al. [8] infer users' locations based on words automatically identified from their tweets. Rao et al. [28] believe that different groups of people have specific writing styles, so they propose a novel stacked-SVM-based classification algorithms over a rich set of features obtained from users' tweets to infer attributes like regional origin and political orientation. As we can see, content-based methods rely on two key elements: (i) text that is highly related to the attributes and (ii) efficient classifiers for automatically identifying words or inferring attributes based on extracted words. The requirements for texts make this kind of method very limited, because most of the content in the massive text has nothing to do with inferred attributes. In addition, whether the classifier is appropriate also affects the final inference results. Especially, when putting the problem into a social network, neglecting the social connections between users cannot make accurate inferences.

Following the above drawbacks, some studies explore the social structures to help infer user attributes. Dong et al. [10] discover several interesting social strategies that mobile users frequently use to maintain their social connections, so they propose a factor graph model to capture the interactions between demographic information and social structures. Through experiments, Chen et al. [7] explore how network characteristics impact user attribute inference and implement them with Naive Bayes, Decision Tree, and Logistic Regression. Zheleva et al. [47] show that in addition to friendship links, group affiliations can be carriers of significant information, which encourages them to propose a group-based inference model. Mislove et al. [21] believe that people in the same community share similar attributes, so they infer attributes by detecting local communities with some seed users. However, this assumption is not always correct. For instance, Trauda et al. [34] find that communities in an MIT female network are not correlated with residence attribute. Bhagat et al. [3] first propose a kind of method based on propagating the labels by performing random walks on the graph. However, it is the most traditional random walk method that ignores the hierarchy of user attributes and cannot solve the problems of conflict, indeterminacy, and missing, in addition to its inefficiency in running on large-scale graphs. To avoid the limitations of considering contents or structures alone, Li et al. [17] infer location attributes from both social network and user-centric data in a unified probabilistic framework. They propose a **social-behavior-attribute (SBA)** network model to gracefully integrate social structures, user behaviors and user attributes in a unified framework. The experimental results show that this combination method can correctly infer attributes for significantly more users than previous methods.

Recent studies focus on leveraging machine learning techniques to infer user attributes in the social network. Earlier, Rao et al. [28] propse stacked-SVM-based classification algorithms over a rich set of features to classify user attributes. However, due to the simplicity of the classifier, the inference results cannot reach high accuracy. Pennacchiotti et al. [27] generate features from user profile, tweeting behavior, linguistic content and social network. These features are used in conjunction with a supervised machine learning framework called Gradient Boosted Decision Trees for inference. Yang et al. [43, 44] find that inference for user links and attributes are strongly interleaving, so they propose a unified probabilistic framework through label propagation and graph construction to jointly learn user links and attributes by leveraging the data redundancy and mutual reinforcement. Zeng et al. [46] transform the inference problem as a task of classifying nodes over graphs. They propose the supervised label propagation model, which combines all the features on the social activities to address this problem. Jia et al. [12] model the social network as a pairwise Markov Random Field, they propose AttriInfer, a new method that infers attributes by learning prior probability and computing posterior probability. Because the model can leverage both positive training users and negative training users in the training dataset, AttriInfer outperforms state-of-the-art methods in terms of inference accuracy. Wu et al. [40] discover the correlation between item recommendation and attribute inference, which encourages them to propose an Adaptive Graph Convolutional Network approach for joint learning of these two tasks.

2.2 Hierarchical Inference

Some real-world cases present a pre-defined hierarchical structure of attributes from coarse to fine. However, flat inference methods mentioned above do not explore the relationship existing in the attribute hierarchy, which greatly reduce the effectiveness in the multi-level problem. To address this issue, recently people begin to design new models to deal with hierarchical structures.

The first type of solution is to construct local classifiers [14]. One such method is building a local classifier per attribute node. Valentini et al. [35] trains a binary classifier for each attribute in the hierarchy. By evaluating all the classifier nodes' outputs, they propose a bottom-up algorithm to make inconsistency correction. Wu et al. [39] also construct the classifier in this way,

Table 1. Main Notations Used in This Article

Notation	Description
T, Σ_T, Σ_g	semantic tree, attribute set of T, attribute set of T's gth level
$G(V, E, L)$	labeled graph
V_s,V_u	known vertices set, unknown vertices set
l_x,$w_x(v_i)$	attribute name, the probablity that user v_i has the attribute l_x
$H_g(v_i)$	entropy value of v_i's gth level
$C(v_i)$	credibility of the information provided by v_i
$P_g(v_i, v_j)$	probability that v_i propagate its gth-level attribute to v_j
$N(v_i)$	the set of one-hop neighbors of user v_i
$Parent(l_x)$	parent node of l_x in semantic tree
$DesSet(l_x)$	descendant nodes set of l_x in semantic tree
$BroSet(l_x)$	sibling nodes set of l_x in semantic tree

but they ignore the results of all low-level attributes when facing inconsistency. Secker et al. [30] construct a multi-class classifier for each parent node in the attribute hierarchy, therefore avoiding the problem of making inconsistent predictions. Another construction method is called the local-classifier-per-level approach. Taksa et al. [33] train one multi-class classifier for each level of the attribute hierarchy. Yan and Nakano et al. [23, 42] propose a multi-label active-learning method to improve the performance of classifier by selecting certain attributes of unknown users for manual annotation. However, in the social network scenario, user attributes like occupation and interest are personal privacy, making it difficult for people to manually label them, so this method is the same as applying the classifier directly. Classifier-based approaches have a high requirement for data quality. The construction of classifiers is complicated and the amount of calculation for training is huge.

Hierarchical inference has also been widely studied in text or image classification [20, 26, 32, 37], which is considered as a multi-label classification problem. Hierarchical inference for text takes text content as input, while in image classification, features are generally extracted from the RGB information of the image. Both of them are not available in social network scenarios.

A concept similar to hierarchical inference is hierarchical clustering [19, 38], however, it is a clustering problem that focuses on dividing users into hierarchical communities according to user attributes, while user attributes are still at a single level. Hierarchical clustering is a different research problem from this article, therefore, methods based on hierarchical clustering cannot be applied to our problem.

3 PROBLEM DEFINITION

In this section, we first introduce two vital related concepts and define the multi-level inference problem in the social network. For ease of presentation, we list the notations used throughout the article in Table 1.

3.1 Semantic Tree

The semantic tree is a predefined structure that semantically exists and is used to describe the hierarchical relationship between different user attributes. Each node in the tree is an attribute, represented by l_x. The semantic tree can be defined as T, and we use Σ_T to represent the set of all the attributes in T, which means that $l_x \in \Sigma_T$. $\Sigma_T = \Sigma_1 + \cdots + \Sigma_g + \cdots + \Sigma_d$, where Σ_g represents the attribute set of T's gth level and d is the total number of the hierarchical level. Attributes in Σ_{g+1}

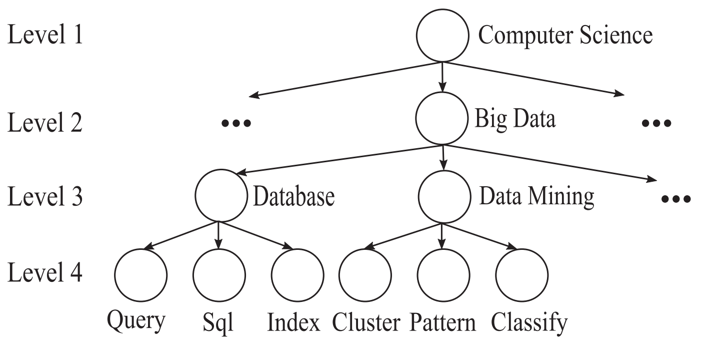
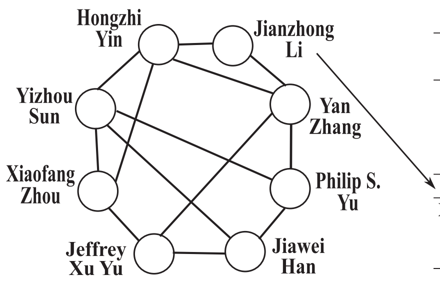


Fig. 3. An example of semantic tree.



Name	Attribute			
	Level 1	Level 2	Level 3	Level 4
Philip S. Yu				
Jiawei Han				
Yan Zhang				
Jeffrey Xu Yu				
Jianzhong Li	Computer Science	Big Data	Database	Query
Xiaofang Zhou				
Yizhou Sun				
Hongzhi Yin				

Fig. 4. An example of labeled graph.

are the refinement of the corresponding parent node in Σ_g, from Σ_1 to Σ_d is the representation of user attributes from coarse-grained to fine-grained.

As it is shown in Figure 3, research interests are organized from coarse- to fine-grained in semantics as the tree goes deeper. Nodes with lower levels represent some general attributes such as Database and Data Mining, while nodes with higher levels represent more specific attributes such as Index and Sql, which are detailed classifications of Database. For an example of Figure 3, $\Sigma_3 = \{\text{Database, Data Mining}, \cdots\}$.

3.2 Labeled Graph

A labeled graph is a simple undirected graph, denoted as $G = (V, E, L)$, where V is the set of vertices and E is the set of edges in G. Each vertex represents a user in the social network and each edge $e \in E$ represented by (v_i, v_j) means that there are some connections between user v_i and v_j. L is a function whose domain is every vertex $v_s \in V_s$, where V_s is the vertex set whose attributes are known, so $L(v_s)$ represents the attribute information of known users.

Figure 4 illustrates an example of labeled graph in a co-author social network. There are eight users in this graph that are researchers of computer science and the attributes are their research interests. In this example, the attributes of Jianzhong Li can be represented as $L(Jianzhong\ Li) = \{\text{Computer Science, Big Data, Database, Query}\}$, where Computer Science $\in \Sigma_1$ is the most coarse-grained description of research interests. Database $\in \Sigma_3$, Query $\in \Sigma_4$, and Database is the parent node of Query in T, which means that Query is a refinement of Database.

Problem. Given a labeled graph $G = (V, E, L)$ with semantic tree T, labeled vertices set V_s and unknown vertices set V_u. We aim to derive a function F, where F maps V_u to a cartesian product of T, defined as $F : V_u \rightarrow \Sigma_1 \times \Sigma_2 \times \cdots \times \Sigma_d$. For every vertex $v_u \in V_u$, $F(v_u)$ outputs the inferred attributes of v_u and the result should satisfy the hierarchy constraint.

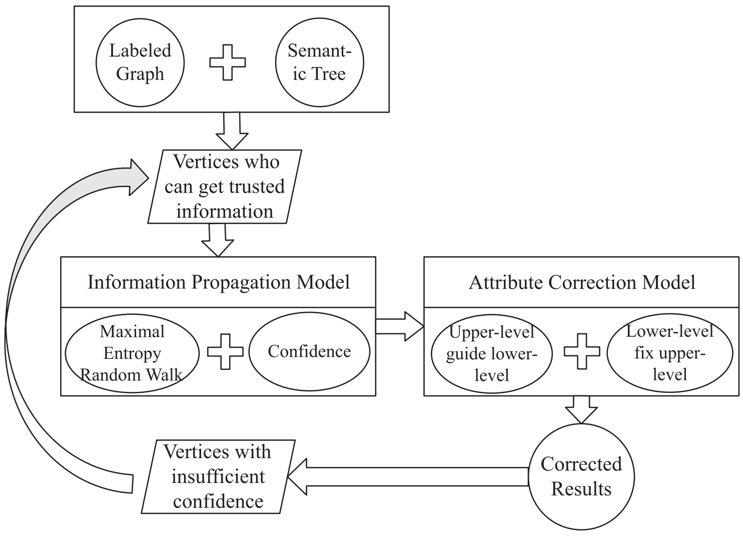

Labeled Graph
Semant-ic Tree
Vertices who can get trusted information
Information Propagation Model
Maximal Entropy Random Walk
Confidence
Attribute Correction Model
Upper-level guide lower-level
Lower-level fix upper-level
Corrected Results
Vertices with insufficient confidence

Fig. 5. Framework of the attribute inference model.

Definition 3.1 (Hierarchy Constraint). The inference result $F(v_u)$ must be on the same path of the semantic tree, which means that for a target user $v_u \in V_u$, $F(v_u) = \{l_1, l_2, \ldots, l_g, \ldots, l_d\}$ satisfy that (i) $l_1 \in \Sigma_1, l_2 \in \Sigma_2, \ldots, l_g \in \Sigma_g, \ldots, l_d \in \Sigma_d$; and (ii) $\forall i \in [1, d-1], l_i = Parent(l_{i+1})$, where $Parent(\cdot)$ represents the father relationship in the semantic tree.

4 PROPOSED MODEL

4.1 Overview

Figure 5 shows the framework of our inference model, which is an iterative procedure. As introduced in Section 3, we use labeled graph to model social networks and semantic tree to represent the hierarchy of attributes. We express the attribute information as a probability structure, which can be divided into two parts, attribute name l_x and the corresponding weight $w_x(\cdot)$. Given a user $v_u \in V_u$, $w_x(v_u)$ is a value that changes with iteration and will eventually reach convergence. If v_u has the attribute l_x in the real world, then as the number of iterations increases, $w_x(v_u)$ will become larger and closer to 1. For a known user $v_s \in V_s$, the value of $w_x(v_s)$ is either 0 or 1 and will not change through iterations. Formally, we use a tuple set $L_g(v) = \{< l_x, w_x(v) >, l_x \in \Sigma_g\}$ to represent the attribute information of user v's gth level of semantic tree.

We first propose an information propagation model to initially calculate the probability of each attribute in the semantic tree for every user $v_u \in V_u$. Based on the maximum entropy theory and one-step random walk, vertices in V_s propagate their attribute information at each level to other vertices. Unknown vertices in V_u who can get trusted information will start inferring. The main idea is that vertices with high entropy collect more information and vertices with high confidence spread more information.

In the process of information propagation, vertices receive multiple attributes with probabilities. But due to the model being carried out separately in levels, it may lead to a result that does not conform to the hierarchy constraint, so based on the semantic tree, we propose an attribute correction model. This model realizes the upper-level attributes guide the inference of lower-level ones and the lower-level attributes correct the upper level's result. Carrying out corrections in a top-down manner can make the final results more accurate.

Except for the vertices whose confidence is sufficient enough, the results after correction will be used as the input of the next round of propagation. These two models are described in detail below.

4.2 Information Propagation Model

In this section, considering that closely connected users in social networks are more likely to have similar attributes, we introduce a method based on maximum entropy random walk to transfer attributes between vertices, to calculate the probability that the target vertex has each attribute. First, we need to determine the propagation probability on each edge. We assume that the probability of a user propagating attribute information to his neighbors is not equal but the sum is 1. The main idea is that vertices whose attributes are more uncertain need to collect more information from the network, so the transition probability to this vertex is relatively large. We use entropy to measure the uncertainty of user attributes, which is proposed by Shannon et al. [31] and commonly used to quantify the uncertainty of random variable results.

The probabilities of user attributes in each iteration are different among different levels. Attributes of some levels may already be distinguishable, while some levels require a large amount of information to be collected, so the information should spread independently at each level, so as the calculation of entropy value and transition probability. As mentioned before, attribute information of v_j's gth level can be represented by $L_g(v_j) = \{< l_x, w_x(v_j) >, l_x \in \Sigma_g\}$. Then the entropy value of v_j's gth level $H_g(v_j)$ can be calculated as follows:

$$H_g(v_j) = -\sum_{l_x \in \Sigma_g} w_x(v_j) \times \ln w_x(v_j), \tag{1}$$

where $H_g(v_j)$ indicates the uncertainty about the attributes of v_j's gth level. In the first iteration, for each vertex $v_j \in V_u$, $w_x(v_j) = \frac{1}{|\Sigma_g|}, l_x \in \Sigma_g$. There is a special case that when $w_x(v_j) = 0, H_g(v_j) = 0$, which means that this user certainly does not have attribute l_x, no additional information from other vertices is needed.

Based on maximum entropy theory, if v_i is a neighbor of v_j, the transition probability from v_i to v_j at gth level is computed as follows:

$$P_g(v_i, v_j) = C(v_i) \times \frac{H_g(v_j)}{\sum_{v_k \in N(v_i)} H_g(v_k)}, \tag{2}$$

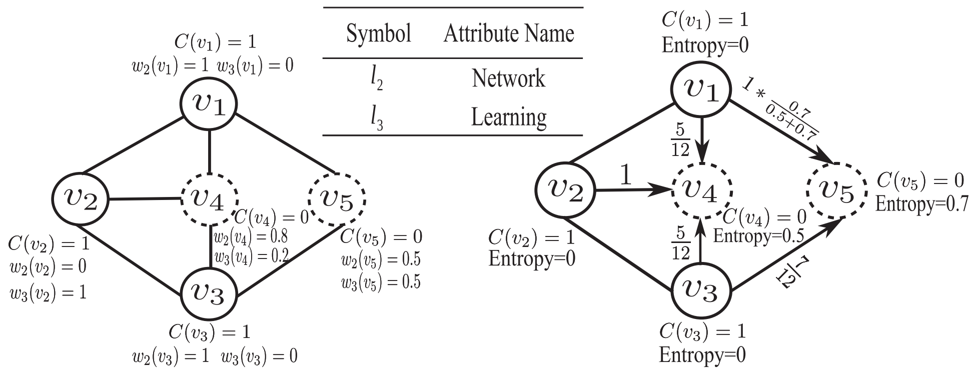
where $C(v_i)$ is called confidence, representing the credibility of the information provided by v_i. We discuss how to compute vertex's confidence in Section 4.4.

Utilizing the transition probability, users in V_u receive multiple attributes' probabilities. We use the following equation to integrate that information and map the results to $[0, 1]$ as the updated value of $w_x(v_j)$ in this iteration:

$$w_x(v_j) = \frac{\sum_{v_i \in N(v_j)} P_g(v_i, v_j) \times w_x(v_i)}{\sum_{l_y \in \Sigma_g} \sum_{v_i \in N(v_j)} P_g(v_i, v_j) \times w_y(v_i)}. \tag{3}$$

By calculating entropy and transition probability at each level of the semantic tree, the attribute information is spread through the labeled graph in the form of probability independently by levels, and the unknown vertices obtain this information to make a preliminary inference.

Example 1. Here, we present Figure 6 to explain the information propagation process. Figure 6(a) is a local part of labeled graph where v_1, v_2, and v_3 are known vertices, v_4 and v_5 are unknown vetices to be inferred. $C(v_1) = 1$ represents v_1's confidence, which is used to quantify the credibility of the attribute probabilities provided by v_1 and $w_2(v_1) = 1$ means the probability that user v_1 has



Symbol	Attribute Name
l_2	Network
l_3	Learning

(a) a labeled graph with confidence and attribute information of Level2 (b) calculate vertex entropy and transition probability

Fig. 6. An example of the information propagation process.

attribute l_2 is 100%. l_2 and l_3 are two attributes in the second level of the semantic tree, representing "Network" and "Learning." In this example, we only show the propagation in Level 2 of the semantic tree, and the calculation method is the same for other levels.

We first calculate the second-level entropy of each vertex according to Equation (1). $H_2(v_1) = H_2(v_2) = H_2(v_3) = 0, H_2(v_4) = -[w_2(v_4)\times\ln w_2(v_4)+w_3(v_4)\times\ln w_3(v_4)] = -(0.8ln0.8 + 0.2ln0.2) = 0.5$, $H_2(v_5) = -[w_2(v_5) \times \ln w_2(v_5) + w_3(v_5) \times \ln w_3(v_5)] = -(0.5ln0.5 + 0.5ln0.5) = 0.7$. As shown in Figure 6(b), the transition probability from v_1 to v_5, denoted as $P_2(v_1, v_5) = C(v_1) \times \frac{H_2(v_5)}{H_2(v_2)+H_2(v_4)+H_2(v_5)} = 1 \times \frac{0.7}{0.5+0.7+0} = \frac{7}{12}$. Then, we propagate the attribute probabilities of v_1, v_2 and v_3 according to the calculated transition probability on each edge. The information v_4 receives is $w_2(v_4) = w_2(v_1)\times P_2(v_1, v_4)+w_2(v_2)\times P_2(v_2, v_4)+w_2(v_3)\times P_2(v_3, v_4) = 1\times\frac{5}{12}+0 \times 1 + 1\times\frac{5}{12} = \frac{5}{6}$, and $w_3(v_4) = 1 \times 1 = 1$. Finally, we map the results to $[0, 1]$, and get the result after this round of iteration that v_4 has the attribute "Network" with a probability of $\frac{5}{11}$ and "Learning" with a probability of $\frac{6}{11}$.

4.3 Attribute Correction Model

Since the propagation is carried out independently in levels, inference results obtained by the information propagation model may not satisfy the hierarchical constraint. In this section, we introduce our attribute correction model, which modifies the results by establishing connections between attribute levels.

Definition 4.1 (Descendant-attributes Set). For an attribute node l_x on the semantic tree T, let T_x be the subtree on T rooted at node l_x, $T_x \subseteq T$ and $Root(T_x) = l_x$. Then l_x's descendant-attributes set is defined as $DesSet(l_x) = \{\Sigma_{T_x} - \{l_x\}\}$.

Definition 4.2 (Brother-attributes Set). For an attribute node l_x on the semantic tree T, l_x's brother-attributes set is defined as $BroSet(l_x) = \{l_y|Parent(l_y) = Parent(l_x)\}$.

The main idea of attribute correction is that in a certain round of iteration, attributes at the same level of the semantic tree may get the same or wrong results after random walk, but there is a clear distinction between their parent-attribute or descendant-attributes. Therefore, attributes whose parent-attribute or descendant-attributes have a high weight are more likely to appear in the real result. Based on this, we distribute the probability of the parent-attribute to the target attribute and its brother-attributes in a top-down manner. We regard the target attribute and the descendants-attributes as a whole, and the sum of the probabilities is used as the distributed weight.

We believe that descendant-attributes can play a very positive role in attribute correction. For an attribute l_x, if the attributes obtained during the information propagation show a high probability of $w_x(v_j)$, it means that the user is also likely to have the attributes in the $DesSet(l_x)$. Based on this, in the correction process, we merge the descendant-attributes, and consider it with the probability collected during the information propagation model. After combining these two parts, the probability is denoted as $W_x(v_j)$, which is used as the weight for distribution:

$$W_x(v_j) = (1-\alpha) \times w_x(v_j) + \alpha \times \sum_{l_y \in DesSet(l_x)} w_y(v_j), \tag{4}$$

where α represents a correction strength. When the value of α is large, the result is more inclined to the hierarchy of the semantic tree; otherwise, it is more inclined to the information collected during the propagation.

$Parent(l_x)$ is the coarse-grained representation of l_x, and $DesSet(l_x)$ are its fine-grained attributes, they affect the probability of l_x jointly. Therefore, the probability of l_x's parent-attribute $w_{Parent(l_x)}(v_j)$ will be distributed by Equation (5) based on the weight obtained by Equation (4). Results after distribution are the revised attribute probabilities, which are used as the final output of this round of iteration:

$$w_x(v_j) = w_{Parent(l_x)}(v_j) \times \frac{W_x(v_j)}{W_x(v_j) + \sum_{l_y \in BroSet(l_x)} W_y(v_j)}. \tag{5}$$

There is another case for the attributes at the highest level. Since they do not have descendant-attributes, only the results of the brother-attributes after random walk are considered as the weight for distribution. The probability of this type of attributes is corrected as follows:

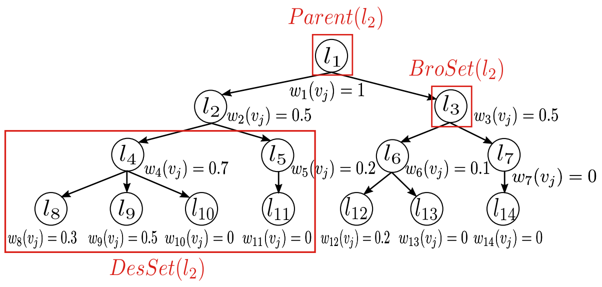
$$w_x(v_j) = w_{Parent(l_x)}(v_j) \times \frac{w_x(v_j)}{w_x(v_j) + \sum_{l_y \in BroSet(l_x)} w_y(v_j)}. \tag{6}$$

By attribute correction, we make the probability of the upper-level attributes equal to the sum of the probabilities of all its descendant attributes. From Equations (5) or (6), we get Lemma 4.1 straightforwardly.

Lemma 4.1. *For any unknown user v_j whose inference results have been corrected and any attribute l_x, we have $w_{Parent(l_x)}(v_j) = w_x(v_j) + \sum_{l_y \in BroSet(l_x)} w_y(v_j)$.*

Example 2. Figure 7 is a semantic tree with the value of $w_x(v_j)$ of user v_j for every attribute l_x in a certain iteration. $w_2(v_j) = 0.5$ means that the probability that user v_j has attribute l_2 is 0.5, which is calculated by information propagation introduced in Section 4.2. l_1 to l_{14} represent user attributes, which correspond to real-world research interests as shown in the table. According to Definitions 4.1 and 4.2, $DesSet(l_2) = \{l_4, l_5, l_8, l_9, l_{10}, l_{11}\}$ and $BroSet(l_2) = \{l_3\}$. In this example, we show how to modify $w_2(v_j)$ through the hierarchical relationship of the semantic tree.

From Figure 7, we can get that $w_{Parent(l_2)}(v_j) = w_1(v_j) = 1$. The probabilities of $DesSet(l_2)$ can be calculated as $\sum_{l_y \in DesSet(l_2)} w_y(v_j) = w_4(v_j) + w_5(v_j) + w_8(v_j) + w_9(v_j) + w_{10}(v_j) + w_{11}(v_j) = 0.7 + 0.2 + 0.3 + 0.5 + 0 + 0 = 1.7$. After that, we combine the target attribute with descendant-attributes by Equation (4), $W_2(v_j) = (1-\alpha) \times w_2(v_j) + \alpha \times \sum_{l_y \in DesSet(l_2)} w_y(v_j) = 0.5 \times 0.5 + 0.5 \times 1.7 = 1.2$ (Parameter α is set to 0.5). In the same way, the probabilities of brother-attributes and their descendant-attributes is calculated by $\sum_{l_y \in BroSet(l_2)} W_y(v_j) = W_3(v_j) = (1-\alpha) \times w_3(v_j) + \alpha \times \sum_{l_y \in DesSet(l_3)} w_y(v_j) = 0.5 \times 0.5 + 0.5 \times (0.1 + 0.2) = 0.4$. Finally, through Equation (5), we



Symbol	Attribute Name	Symbol	Attribute Name
l_1	Computer Science	l_8	Cellular
l_2	Network	l_9	Cloud
l_3	Learning	l_{10}	Localization
l_4	Wireless	l_{11}	Bandwidth
l_5	Qos	l_{12}	Recognition
l_6	Image	l_{13}	Track
l_7	Speech	l_{14}	Understanding

Fig. 7. The value of $w_x(v_j)$ of user v_j for every attribute l_x in semantic tree after a certain iteration.

can distribute $w_{Parent(l_2)}(v_j)$ between $W_2(v_j)$ and $W_y(v_j), l_y \in BroSet(l_2)$ to get the final corrected result $w_2(v_j) = w_{Parent(l_2)}(v_j) \times \frac{W_2(v_j)}{W_2(v_j)+\sum_{l_y \in BroSet(l_2)} W_y(v_j)} = 1 \times \frac{1.2}{1.2+0.4} = 0.75$.

4.4 Confidence Calculation

In this section, we introduce the computing method of confidence. The confidence of v_j denoted as $C(v_j)$ represents the credibility of the information provided by v_j. In the beginning, for all vertices $v_s \in V_s$, $C(v_s)$ is set to 1 and for all vertices $v_u \in V_u$, the default confidence is 0. Vertices with larger confidence spread more information during the process of propagation. Confidence can be computed as follows:

$$C(v_j) = \frac{\sum_{v_i \in N(v_j)} C'(v_i)}{|\{v_i | v_i \in N(v_j)\}|}. \tag{7}$$

The calculation of confidence adopts the idea of weighted voting. Considering that the information provided by users who are closer to the target vertex is more reliable, we use the parameter $\beta \in (0, 1)$ to limit the voting weight, which means that the credibility of the information provided by two-hop nodes is reduced by β:

$$C'(v_i) = \begin{cases} C(v_i), & v_i \in V_s, \\ \beta \times C(v_i), & v_i \in V_u. \end{cases} \tag{8}$$

With the continuous iteration of the algorithm, the confidence of unknown vertices will show a gradual upward trend.

Example 3. In this example, we show how to update the unknown users' confidence after a certain iteration. In Figure 8, the confidence of three known users v_1, v_2, and v_3 is 1, $C(v_4) = 0$ and $C(v_5) = 0.5$ mean that the confidence of v_4 and v_5 is currently 0 and 0.5, respectively.

Taking user v_4 as an example, since the attributes of v_4 are inferred from its one-hop neighbors, the confidence of v_4 is determined by their credibility. According to Equation (7), $C(v_4) = \frac{C'(v_1)+C'(v_3)+C'(v_5)}{3}$. Among them, since v_1 and v_3 are known vertices, $C'(v_1) = C(v_1) = 1, C'(v_3) = C(v_3) = 1$. As an unknown node, v_5's attribute information is provided by its one-hop neighbors, which are the two-hop neighbors of v_4, so according to Equation (8), $C'(v_5) = \beta \times C_(v_5) = 0.8 \times 0.5 = 0.4$. Finally, we update $C_(v_4)$ by $C(v_4) = \frac{C'(v_1)+C'(v_3)+C'(v_5)}{3} = \frac{1+1+0.4}{3} = 0.8$.

4.5 Generalized Semantic Tree

Since the hierarchical structure of attributes is given by the users according to the actual application scenarios, we propose the concept of generalized semantic tree, so that MLI can be applied to a variety of tree-like hierarchical structures.

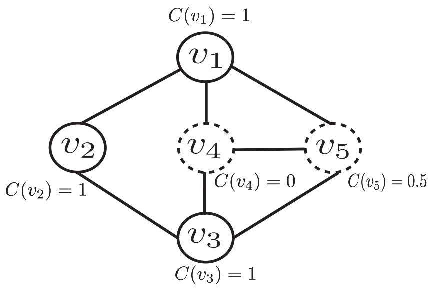


Fig. 8. A labeled graph with confidence during an iteration ($\beta = 0.8$).

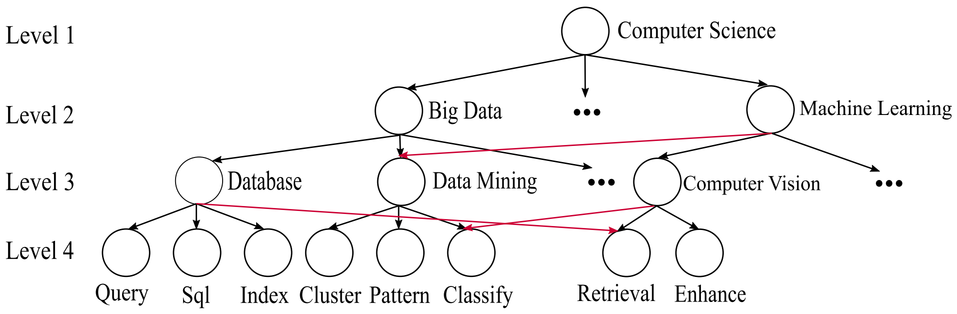


Fig. 9. An example of generalized semantic tree.

As we introduced in Section 3, the user-defined attribute hierarchy is organized in the form of tree, while in some cases, the non-intersection restriction between subtrees cannot fully describe the relationship between attributes. To enable our model to be applied to a wider range of scenarios, we propose a new structure representing the hierarchical relationship of attributes called generalized semantic tree. We still use T to formally describe the hierarchy and Σ_g to represent the attribute set at the gth level of the generalized semantic tree.

As an example of generalized semantic tree shown in Figure 9, we can see that different from the regular semantic tree, attribute nodes such as Data Mining, Classify, and Retrieval may have multiple parents, which means that these attributes are interdisciplinary and $parent(l_x)$ becomes a set. This is reasonable, because Classify can not only be the research interest under Data Mining, Computer Vision also studies the classification of images. Both generalized semantic tree and regular semantic tree are commonly used in practice and can be handled by our model MLI.

In the process of attribute correction, Equations (5) and (6) need to be adjusted to adapt to the situation of multiple parent nodes as follows, where the parent node of $BroSet(l_x)$ is l_z:

$$w_x(v_j) = \sum_{l_z \in Parent(l_x)} w_z(v_j) \times \frac{W_x(v_j)}{W_x(v_j) + \sum_{l_y \in BroSet(l_x)} W_y(v_j)}, \tag{9}$$

$$w_x(v_j) = \sum_{l_z \in Parent(l_x)} w_z(v_j) \times \frac{w_x(v_j)}{w_x(v_j) + \sum_{l_y \in BroSet(l_x)} w_y(v_j)}. \tag{10}$$

Through experiments in Section 7.4, we verify that in some scenarios, generalized semantic tree can better describe the hierarchical structure of attributes, and resulting in a more accurate result of inference.

5 ATTRIBUTE INFERENCE ALGORITHM

5.1 The "Ripple" Algorithm

By making the inference start from sub-networks with sufficient attribute information, we propose a "Ripple" algorithm to improve the efficiency and effectiveness of our model. The input is a labeled graph $G(V, E, L)$, semantic tree T, and known vertices set V_s, where the attributes on every vertex $v_s \in V_s$ are completely given. The output of each iteration is $F(v_u)$ for every vertex $v_u \in V - V_s$. $F(v_u)$ is iteratively updated, and the final result will be obtained after convergence.

The main idea of this algorithm is similar to minesweeper games. In minesweeper, some squares have limited information, which is impossible to judge, so we usually ignore them temporarily and give priority to the discernible squares. As the game progresses, more information will appear, squares that could not be judged before can be solved accurately now. So as in our problem, we prioritize the inference on vertices who can get enough trusted information, and then gradually spread them to all vertices. Based on this, we use a vector to store users' confidence and a matrix to store users' entropy by levels. In each iteration, vertices whose neighbors' confidence is higher than a certain value are selected for priority inference. Then, based on the entropy, we calculate the transition probability on each edge related to these users. After that, vertices obtain attribute information hierarchically from their nearby vertices according to the transition probability. Then, we revise the result by redistributing the probability of parent-attributes on semantic tree in a top-down manner. After iteration, some users' confidence is improved, so that additional users will be inferred in the next iteration. Moreover, to improve the efficiency, users whose confidence is large enough are no longer inferred. Before entering the next iteration, if no vertex meets the confidence requirement, then the confidence limit should be relaxed. Inference results are continuously updated with the iteration until the convergence is satisfied.

The details of the algorithm are shown in Algorithm 1. First, we use Equation (1) to calculate entropy $H_g(v_u)$ for all $v_u \in V_u$ at each level (lines 2–6). Before start inferring, we advancely calculate the confidence after inference $C_{pre}(v_u)$ (called the expected confidence) to select vertices with sufficient surrounding information. Vertices whose expected confidence are no less than ε call Algorithm 2 to start infer hierarchically (lines 8–10). Algorithm 2 is the calculation process of attribute probability $w_x(v_u)$. For $v_i \in N(v_u)$, we first use the entropy value to calculate the transition probability $P_g(v_i, v_u)$ on each edge by Equation (2). Next, based on the transition probability, algorithm performs a random walk process and calculates the attribute probability $w_x(v_u)$ by Equation (3). After all levels' information are collected, correction can be performed by Equations (5) or (6). Finally, Equation (7) is used to update the confidence. The output $F(v_u)$ is the inference result of this round of iteration.

Back to Algorithm 1, lines 11–13 mean that vertices with confidence no less than θ are removed from V_u. They are trusted enough to be treated as known vertices just like those in V_s and will no longer be inferred in subsequent iterations. Before entering the next iteration, judge whether $\{v_u | C_{pre}(v_u) > \epsilon, v_u \in V_u\}$ is empty. If the set is empty, then it means that no vertex's surrounding information can meet the current requirements, and the expected confidence limit should be appropriately relaxed, so we reduce ϵ to half of the previous one (lines 18–20). After that, $F(v_u)$ for all $v_u \in V_u$ enter the next iteration, and Algorithm 1 return to step 2 until the convergence is satisfied.

The condition of convergence is given by the following equation:

$$\sum_{v_u \in V - V_s} \sum_{l_x \in \Sigma_T} |diffw_x(v_u)| \leq |V - V_s| \times |\Sigma_T| \times \sigma, \tag{11}$$

where $diff(w_x(v_u))$ is the difference on $w_x(v_u)$ after the inference algorithm is executed, $|\Sigma_T|$ is the total number of attributes, and σ is a threshold to control the number of iterations. As the

ALGORITHM 1: Multi-level Attribute Inference(G, T, V_s)

```
Input:  $G(V, E, L)$, semantic tree $T$ and $V_s$.
Output: $F(v_u)$ for every vertex $v_u \in V - V_s$.
1:  $V_u = V - V_s$
2:  for level $g$ in [2,d] do
3:      for every vertex $v_u \in V_u$ do
4:          compute $H_g(v_u)$
5:      end for
6:  end for
7:  for every vertex $v_u \in V_u$ do
8:      if $C_{pre}(v_u) > \varepsilon$ then
9:          $F(v_u)$ = Probability Compute $(v_u, G, T)$
10:     end if
11:     if $C(v_u) > \theta$ then
12:         remove $v_u$ from $V_u$
13:     end if
14: end for
15: if $\sum_{v_u \in V - V_s} \sum_{l_x \in \Sigma_T} |diffw_x(v_u)| \leq |V - V_s| \times |\Sigma_T| \times \sigma$ then
16:     return $F(v_u)$ for every unknown vertex $v_u \in V - V_s$
17: else
18:     if $\{v_u | C_{pre}(v_u) > \varepsilon, v_u \in V_u\}$ is empty then
19:         $\varepsilon \leftarrow \frac{1}{2}\varepsilon$
20:     end if
21:     return step 2
22: end if
```

ALGORITHM 2: Probability Compute(v_u, G, T)

```
1:  for level $g$ in [2, d] do
2:      for every vertex $v_i \in N(v_u)$ do
3:          compute $P_g(v_i, v_u)$
4:      end for
5:      for every attribute $l_x \in \Sigma_g$ do
6:          compute $w_x(v_u)$
7:      end for
8:  end for
9:  for every attribute $l_x \in \Sigma_T$ do
10:     correct $w_x(v_u)$
11: end for
12: update $C(v_u)$
13: return $F(v_u)$
```

discussion in Reference [18], the convergence can be satisfied for a maximal entropy random walk, which also shows that our method can finally satisfy convergence.

Example 4. We use the example given in Figure 10 to introduce the "Ripple" algorithm. There are three vertices (v_i, v_j, v_k) to be inferred in Figure 10(a). The current confidence of known vertices is 1 and for the unknown vertices is 0. To judge whether the vertex should enter this iteration, we advancely calculate the confidence after inference by using Equation (7), which is called expected confidence. For example, if v_i is inferred in this iteration, its confidence will become $C_{pre}(v_i)$ =

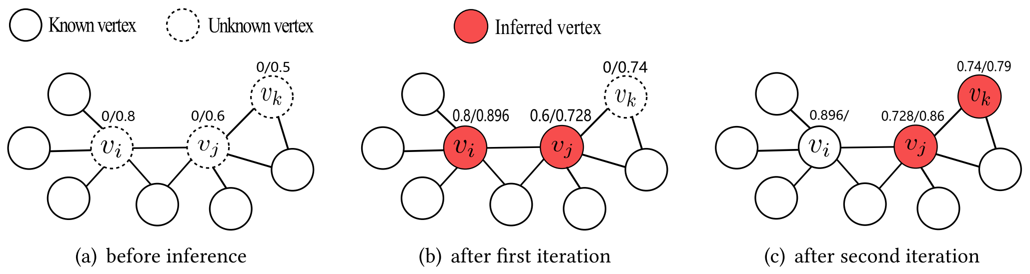


Fig. 10. An example of the algorithm execution process ($\varepsilon = 0.5, \theta = 0.85, \beta = 0.8$).

$\frac{\sum_{v \in N(v_i)} C'(v)}{|\{v|v \in N(v_i)\}|} = \frac{1+1+1+1+0}{5} = 0.8$. Similarly, $C_{pre}(v_j) = 0.6, C_{pre}(v_k) = 0.5$. Numbers above the vertex represents the current confidence/expected confidence.

Vertices whose expected confidence exceeds the parameter ($\varepsilon = 0.5$ in this example) mean that they can obtain sufficient information and will give priority to inference. Since $C_{pre}(v_i) = 0.8 > 0.5, C_{pre}(v_j) = 0.6 > 0.5$ and $C_{pre}(v_k) = 0.5$, in the first iteration, as shown in Figure 10(b), v_i and v_j participate in the inference. After completing the first round of inference, v_i and v_j's current confidence has been changed to 0.8 and 0.6. Take v_k as an example, because it did not participate in the previous iteration, the current confidence is still 0, but the expected confidence changes as the surrounding users have been inferred, $C_{pre}(v_k) = \frac{\sum_{v \in N(v_k)} C'(v)}{|\{v|v \in N(v_k)\}|} = \frac{1+0.6*0.8}{2} = 0.74 > 0.5$. This time, all three vertices meet the confidence threshold, which means that all of them are inferred in the second iteration. We can observe from Figure 10(c) that after two times of iteration, the current confidence of v_i has reached 0.896 (exceed the parameter $\theta = 0.85$), which is already credible enough and inference on this vertex will not change the result significantly, so v_i is regarded as known vertices and no longer participate in subsequent iterations. v_j and v_k continue to be inferred until the current confidence exceeds θ or reach the maximum number of iterations.

5.2 Time Complexity

We assume that the labeled graph G has n vertices, the semantic tree has d levels and p attributes. For $v_u \in V_u$, we inference k users at each iteration. We need to compute the entropy of all vertices independently by levels, so the time complexity is $O(d|V_u|)$. Then to compute the transition probability, for each vertex, we need to go through all its neighbors. Therefore, the time complexity is $O(dnm)$, where m is the average degree of the vertices in G. When calculating $w_x(v_j)$, it is necessary to traverse all one-hop neighbors of v_j to obtain his attribute weights, so the time complexity to calculate the weights of all attributes for all unknown users is $O(pkm)$. Above all, the time complexity of information propagation is $O(d|V_u| + dnm + pkm) = O(dnm + pkm)$. After that, we need to modify every attribute for each user by the complexity of $O(pk)$. Besides, the time complexity of updating confidence is $O(km)$. To sum up, the total time complexity of our algorithm for one iteration is $O(dnm + pkm)$.

When we use the "Ripple" algorithm for optimization, the number of users participating in each iteration will decrease, so that $O(pkm)$ will be reduced, thereby reducing the time complexity of the algorithm.

6 EXPERIMENTAL SETUP

6.1 Dataset

DBLP Dataset. DBLP is a database system of English essays in the computer field. In the experiment, each author of DBLP is a vertex, and their research fields are used as the attributes to be

Table 2. Statistics of DBLP Dataset

#Users	#Relationships	Average degree
10K	71,482	14.28
20K	157,150	15.71
40K	359,967	17.99
80K	609,265	15.61

inferred. When two users have published a paper as co-authors, they are connected by an edge. By checking and manually annotating the user attributes with the research fields in ACM Computing Classification system and China Computer Federation, we organize a DBLP co-authorship network with 80K users for experiments. To further discuss how the method is affected by the number of users, we randomly sample 10K, 20K, and 40K users from it as subsets. The statistical information is shown in Table 2. We release the dataset and code with user guides on GitHub.[1]

The semantic tree is regarded as an input of our multi-level inference problem. In DBLP dataset, every user has a set of characters, which includes a large number of coarse-grained and fine-grained attributes, but they are very messy and have no uniform hierarchical relationship. For example, "#Saeed Salem# #361433# #Data Mining; Clustering; Big Data; Maximal; Patterns; Sequential; Gene; Effective; Similarity; Cohesive;#" is a piece of data in DBLP, including username, user ID, and user attributes. We spent a lot of manpower and time dealing with the attributes in DBLP dataset and constructing a semantic tree for experiments. The constructed semantic tree has four levels, the root node is "Computer Science," Level 2 has 8 attributes, Level 3 has 17 attributes, and Level 4 has 56 attributes. The construction method is as follows.

We first clean the user attributes in the dataset. For the problems that there are too many research fields and most of them rarely appear in the dataset, we merge attributes with the same or similar semantics, such as "Cluster," "Clusters," and "Clustering," "Tracking" and "Trace," remove the attributes with unclear semantics, such as "z, un, old," and only keep frequently occurring attributes. After that, we focus on how to construct a reasonable hierarchy from a large number of attributes, which should not only semantically conform to the professional domain background but also make most users in the dataset satisfy the hierarchy constraint. We construct the semantic tree in a top-down manner, and the root node is "Computer Science." Taking the classifications of computer research fields by ACM Computing Classification System (https://dl.acm.org/ccs) and China Computer Federation (https://www.ccf.org.cn) as background knowledge, we find that "Network," "Big Data," "Hardware," "Security," "Learning," "Computing," and "Software" appear most frequently in the dataset and can basically cover the research fields of computer science, so we take these eight research fields as the second-level attributes of the semantic tree. When constructing the next level, we first select some attributes, which according to background knowledge are sub-research fields of second-level attributes and also appear frequently in the dataset. For Saeed Salem in the example, "Big Data" is his second-level attribute, and then we select "Data Mining" as the third-level attribute. Finally, we use the same method to construct the fourth level, achieving level-by-level refinement of user attributes.

To construct the generalized semantic tree, on the basis of background knowledge, we establish connections for attributes that are not directly connected in the semantic tree but still have obvious semantic relations. For example, both "Computer Science-Big Data-Data Mining-Classification" and "Computer Science-Learning-Image-Classification" are legal paths of the generalized semantic tree. Because semantically, classification can belong to the field of data mining or image processing.

[1]The dataset and code with running instructions are available at https://github.com/HangLotily/MLI.

These two types of hierarchies are both commonly used in practice, just depending on how the user defines them.

6.2 Comparative Approaches

We compare our method MLI with the following classic attribute inference baselines.

Local Classifier (LC) [35, 42]. Local Classifier approach is a common method for multi-level inference. In this method, we construct a multi-class classifier for each level of user attributes. Taking the known users as the training set, the users are coded according to the attributes of the users' one-hop neighbors, and then the SVM classifier is trained to infer unknown users' attributes at each level.

Hierarchical Structure Classification (HSC) [23, 36]. Active learning methods use classifiers combined with manual labeling to achieve the classification of attributes, however, in social network scenarios, user attributes such as occupation and interests are personal privacy, which are difficult to label manually, so we directly utilize the classifiers in the article as a baseline. HSC is a multi-label decision tree classifier capable of handling hierarchical structures. Taking the inference of attribute l_x as an example, in the training phase, users with l_x are used as positive instances, and users without l_x but with $Parent(l_x)$ are used as negative instances, to learn a decision tree for each hierarchical edge.

Community Detection (CD) [21]. Community Detection is a classical method for inferring attributes based on graph structure. We apply the effective community detection method to merge structurally related users into a community. Unknown users in the same community share attributes with other users, which means that attributes with the largest proportion in the community are the inference results of unknown users.

Traditional Random Walk (TRW) [25]. Traditional Random Walk propagates the information of known users to unknown users in a way of random walk based on maximum entropy. Propagation probability is calculated by the entropy value of user attributes. Users with high entropy are assigned a larger propagation probability, because they need more information for inference. This method can mine the global graph structure, but is a single-level method. We apply the algorithm separately at each level of the semantic tree as another baseline.

Node2Vec [11]. Node2Vec is an embedding method that can automatically extract node features based on network structure and output a multi-dimensional vector. We take the known users as the training set, the node vector output by Node2Vec as the input, and the one-hot encoding of the user attributes as the training target to train a deep neural network. After several rounds of iterations, the trained model can be used to infer the attributes of unknown users.

Feature Propagation (FP) [29]. Feature Propagation is a recently proposed approach for handling missing node features in graph-learning tasks. FP is based on the minimization of Dirichlet energy and can be implemented by propagating known features in the graph. In our experiments, we treat user attributes as features and implement inference through propagating known features hierarchically by FP.

6.3 Evaluation Metrics

In our experiment, we randomly select a certain number of users and hide their attributes as groud-truth. After inference, compare the results with the ground-truth attributes to evaluate the effect of the model. We use six metrics to make a comprehensive evaluation of the inference results.

hP, hR, hF. Due to the hierarchical nature of the predicted attributes, common flat evaluation measures such as precision and recall are not suitable for our multi-level problem. Intuitively, misinference to a sibling of the correct attribute is much better than misinference to a distant node. To evaluate the multi-level inference module reasonably, we modify **hierarchical precision (*hP*)**,

hierarchical recall (*hR*), and **hierarchical f-measure (*hF*)** proposed by Kiritchenko et al. [15]. The calculation methods are as follows:

$$hP = \frac{1}{|V_u|} \times \sum_{v_u \in V_u} \sum_i \frac{|P_i(\hat{v_u}) \cap T_i(\hat{v_u})|}{d \times |P_i(\hat{v_u})|}, \tag{12}$$

$$hR = \frac{1}{|V_u|} \times \sum_{v_u \in V_u} \sum_i \frac{|P_i(\hat{v_u}) \cap T_i(\hat{v_u})|}{d \times |T_i(\hat{v_u})|}, \tag{13}$$

$$hF = \frac{2 \times hP \times hR}{hP + hR}, \tag{14}$$

where $P_i(\hat{v_u})$ is the set consisting of the attributes predicted for v_u in level i and all their ancestor attributes except the root attribute, similarly $T_i(\hat{v_u})$ is the set of ground-truth attributes in level i and all their ancestor attributes except the root attribute. d is the level number of the semantic tree.

User Accuracy (UAcc). In our problem, a user must have all the ancestor attributes of the correct attribute, and can never be assigned with any descendant attributes of irrelevant attributes. So only when the attributes at all levels are inferred correctly can the user be considered inferred correctly. We define this strict indicator as User Accuracy:

$$UAcc = \frac{1}{|V_u|} \times |\{v_u | v_u \in V_u \wedge P(v_u) = T(v_u)\}|, \tag{15}$$

where $P(v_u)$ and $T(v_u)$ are the set of predicted result and ground-truth attributes of v_u separately.

Jaccard Distance (JD). Jaccard Distance is used to describe the similarity between sets, which is quite suitable for judging the results of multiple inference tasks. The calculation method is as follows:

$$JD = \frac{1}{|V_u|} \times \sum_{v_u \in V_u} \frac{|P(v_u) \cap T(v_u)|}{|P(v_u) \cup T(v_u)|}. \tag{16}$$

Hamming Loss (HL). Hamming Loss is commonly used in hierarchical classification. It directly counts the number of misinferred attributes, which can reflect the effect of the inference model:

$$HL = \frac{1}{|V_u|} \times \sum_{v_u \in V_u} \frac{XOR(P_i(v_u), T_i(v_u))}{d}, \tag{17}$$

where XOR means that if $P_i(v_u)$ is the wrong predicted value, the results will be punished by loss.

For all metrics except Hamming Loss, a larger value means better performance.

7 RESULTS AND ANALYSIS

7.1 Effectiveness of MLI

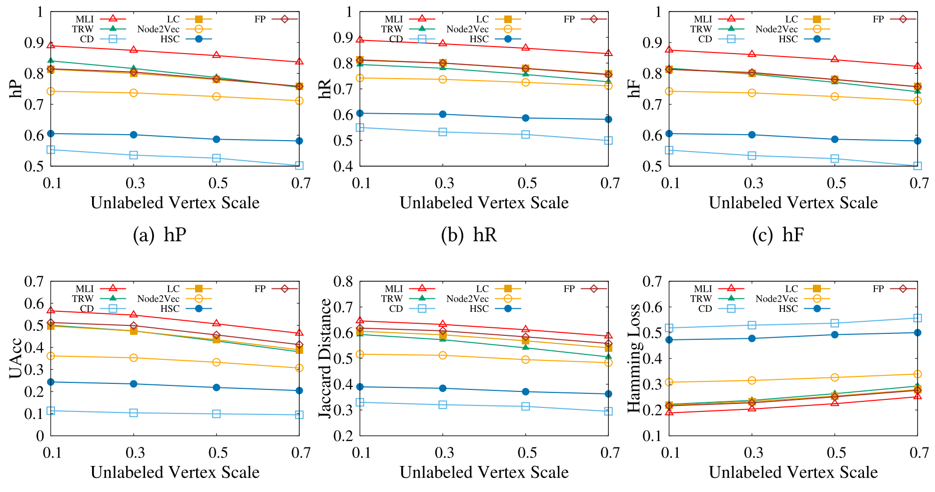
We evaluate the performance of our MLI model in the DBLP dataset with 10K, 20K, 40K, and 80K users, respectively. The proportion of unknown vertices is 50%. Table 3 shows the performance of our model compared to the baselines.

From the results, we can see that MLI has obvious advantages over baselines in various indicators on datasets of different sizes. On the 80K dataset, MLI outperforms the best baseline by 6.07%, 5.09%, 6.37%, 5.1%, 3.26%, and 2.65% on each metric. As the number of users increases, MLI is the least affected method. In the case of hF, when the data set has expanded eight times, hF has dropped by 4.4%, while the best baseline has dropped by 5.63%. Besides, we also found that the method based on community is the least effective in solving hierarchical inference problems. It is probably that CD does not consider the relationship between attributes at all and it is also a difficult task to find out reasonable communities when the number of users is large.

Table 3. Performance Comparison of Diverse Methods for DBLP Dataset with Different User Size

Methods	|G| = 10K						|G| = 20K					
	hP	hR	hF	UAcc	JD	HL	hP	hR	hF	UAcc	JD	HL
MLI	0.9032	0.8734	0.8881	0.5815	0.6625	0.1767	0.8916	0.8355	0.8783	0.5616	0.6457	0.1883
LC	0.8263	0.8263	0.8263	0.5264	0.6305	0.2052	0.8177	0.8177	0.8177	0.5100	0.6154	0.2145
HSC	0.6157	0.6157	0.6157	0.2446	0.3983	0.4685	0.6054	0.6054	0.6054	0.2310	0.3861	0.4782
CD	0.6527	0.6421	0.6473	0.2791	0.4428	0.4062	0.6522	0.6421	0.6471	0.2216	0.4254	0.4108
TRW	0.8625	0.8107	0.8358	0.5245	0.6175	0.2038	0.8472	0.8019	0.8239	0.5089	0.5996	0.2166
Node2Vec	0.8163	0.8163	0.8163	0.4896	0.6061	0.2249	0.7860	0.7860	0.7860	0.4359	0.5709	0.2584
FP	0.8408	0.8326	0.8367	0.5626	0.6494	0.1939	0.8259	0.7968	0.8229	0.5279	0.6297	0.2074

Methods	|G| = 40K						|G| = 80K					
	hP	hR	hF	UAcc	JD	HL	hP	hR	hF	UAcc	JD	HL
MLI	0.8833	0.8572	0.8703	0.5583	0.6416	0.1931	0.8576	0.8309	0.8441	0.5076	0.6116	0.2246
LC	0.8115	0.8115	0.8115	0.4973	0.6058	0.2207	0.7800	0.7800	0.7800	0.4350	0.5685	0.2541
HSC	0.6013	0.6013	0.6013	0.2371	0.3867	0.4779	0.5869	0.5869	0.5869	0.2186	0.3710	0.4925
CD	0.6094	0.6019	0.6057	0.1968	0.3946	0.4554	0.5258	0.5228	0.5243	0.0991	0.3139	0.5368
TRW	0.8323	0.7925	0.8119	0.4959	0.5863	0.2250	0.7869	0.7557	0.7710	0.4287	0.5412	0.2628
Node2Vec	0.7701	0.7701	0.7701	0.4118	0.5524	0.2746	0.7252	0.7252	0.7252	0.3328	0.4955	0.3263
FP	0.8138	0.8100	0.8119	0.5141	0.6173	0.2171	0.7820	0.7788	0.7804	0.4566	0.5840	0.2511



(a) hP (b) hR (c) hF

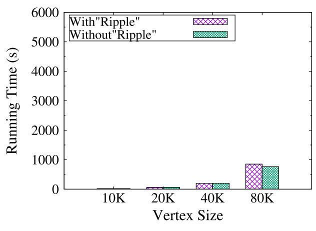
(d) User Accuracy (e) Jaccard Distance (f) Hamming Loss

Fig. 11. Performance of diverse methods for DBLP dataset on different proportion of unknown vertices.

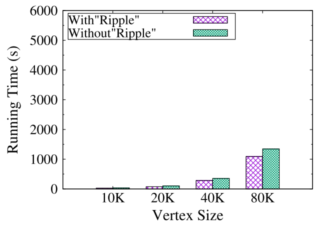
7.2 Impact of The Proportion of Unknown Users

In this experiment, we study the effect of the proportion of unknown users based on the dataset of 80K users, according to our MLI model and baselines. We set the unlabeled scale 10%, 30%, 50%, and 70%, respectively. Figure 11 reports the inference effect of various methods under different unlabeled scales.

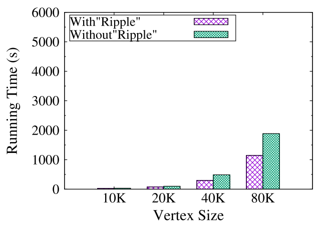
Several observations can be made from the results that in different proportion of unknown vertices, our model performs significantly better than other competitor baselines. Due to the increase in the proportion of unknown users, information that can be used for inference in social networks is continuously compressed, which has an inferior influence on the inference process. We can find that the slowest downtrend methods are Node2Vec and our MLI model, but the overall inference effect of MLI is much better than Node2Vec. It is interesting to see that even though the attributes



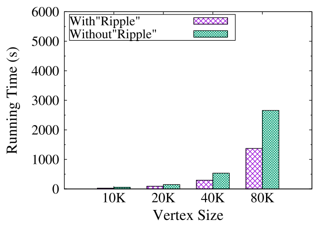
(a) Running Time ($\sigma = 1e - 1$)



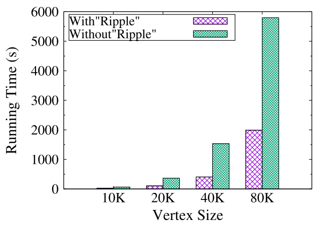
(b) Running Time ($\sigma = 1e - 2$)



(c) Running Time ($\sigma = 1e - 3$)



(d) Running Time ($\sigma = 1e - 4$)



(e) Running Time ($\sigma = 1e - 5$)

Fig. 12. Running time for different vertex size under different σ.

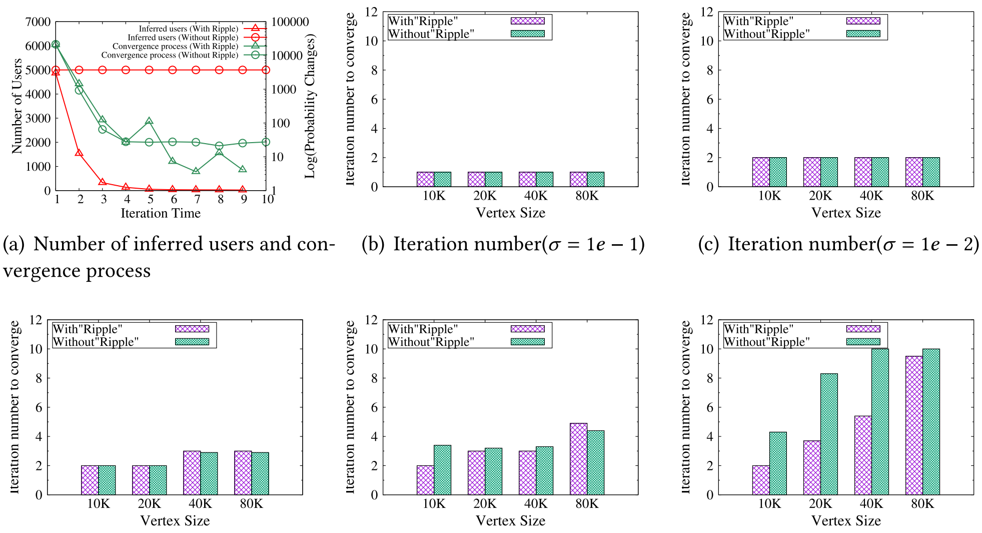
of most users in social networks are unknown, hP, hR, and hF of our method can reach 0.8367, 0.8094, and 0.8228 at the condition of 70% vertices lack of attributes.

7.3 Effectiveness of The "Ripple" Algorithm

By designing the "Ripple" algorithm, we make the inference start from sub-networks with sufficient information and reduce the number of users processed in each iteration. In this part, we carry out a comparative experiment to demonstrate that the algorithm can improve both the efficiency and effectiveness of inference. Parameters in the algorithm are set to $\varepsilon = 0.5$ and $\theta = 0.95$, the proportion of unknown vertices is 50%. We set the maximum number of iterations to 10.

Figure 12 shows the running time of inference with and without the "Ripple" algorithm for different vertex size under different σ. As shown in Equation (11), σ is a parameter that can affect the number of iterations, the convergence conditions become more strict as σ increases. According to Figure 12, except when $\sigma = 0.1$, the running time of using the "Ripple" algorithm is significantly less than that without, and the acceleration effect is more obvious as σ increases. In addition, under the same convergence conditions, as the vertex size increases, the efficiency improvement becomes more significant. Taking Figure 12(d) as an example, when $\sigma = 1e - 4$ and vertex size is 80K, using the "Ripple" algorithm will reduce the running time by half. When $\sigma = 0.1$, the algorithm will terminate after only 1~2 rounds, because the convergence condition is too relaxed, so the running time is similar.

There are two main reasons why the "Ripple" algorithm can speed up inference. First, the number of users in each iteration is decreasing. With the "Ripple" algorithm, only users whose confidence exceeds ε can participate in the inference and whose confidence exceeds θ are removed before the next round of iteration, so the method with "Ripple" does not infer all unknown users in each iteration. Red lines in Figure 13(a) show the number of users inferred per iteration when there are 5,000 unknown users and $\sigma = 1e - 4$. With the "Ripple" algorithm, the number of users that need to be inferred has dropped significantly in the first three rounds, however method without the "Ripple" algorithm has to deal with all 5,000 users in each round, although a considerable number of them have already collected enough credible information.



(a) Number of inferred users and convergence process (b) Iteration number($\sigma = 1e-1$) (c) Iteration number($\sigma = 1e-2$)

(d) Iteration number($\sigma = 1e-3$) (e) Iteration number($\sigma = 1e-4$) (f) Iteration number($\sigma = 1e-5$)

Fig. 13. Effectiveness of the “Ripple” algorithm.

Second, the “Ripple” algorithm can reduce the number of iterations to reach convergence especially when the convergence conditions are strict. Green lines in Figure 13(a) show the convergence process of the algorithm, where y-axis is the log value of the changes on $w_x(v_u)$ for each attribute l_x and each unknown user v_u after a certain iteration. It can be seen from the results that the value basically maintains a rapid decline trend after applying the “Ripple” algorithm, however, without the “Ripple,” it starts to fluctuate around 25 after four rounds of iterations, which shows that method with “Ripple” can reach more strict convergence. Figures 13(b)–13(f) show the number of iterations for the algorithm to converge in different vertex size under different values of σ. From the results, we can see that when σ is small, the number of iterations of the two algorithms is basically the same, but the inference accuracy cannot be guaranteed. When σ exceeds $1e-4$, under the premise of ensuring the inference accuracy, the algorithm with “Ripple” can reach convergence through significantly fewer iterations than the algorithm without. From Figure 13(e), we find that when $\sigma = 1e-4$ and vertex size is 80K, the method with “Ripple” needs one more iteration to reach convergence than the method without. This is because at this time the convergence condition value is 26, after the fourth iteration, the method without “Ripple” reaches 25 and fluctuates around 25 in the subsequent iterations, while the method with “Ripple” is 28 but can continue to decline in subsequent iterations. This small gap results in one more iteration, but the method with “Ripple” can achieve strict convergence and make inference results more accurate. Since the number of users inferred by the “Ripple” algorithm decreases significantly with the iteration, although it requires one more iteration, it still requires less inference time. When $\sigma = 1e-5$ and vertex size is 80K, the algorithm without the “Ripple” still does not converge after reaching the maximum number of iterations, so the advantage will be more obvious than that shown in Figure 13(f).

Experimental results in Table 4 demonstrate the improvement on inference effect by the “Ripple” algorithm. Vertex size is 80K and the proportion of unknown vertices is 50%. Although our approach processes user vertices a lot less, we can still observe from Table 4 that the inference effect has not deteriorated. Since the inference starts from the sub-network with sufficient information and then gradually spreads to other users, vertices processed in the first iteration are users who

Table 4. Performance Comparison for DBLP Dataset with or without "Ripple" Algorithm

	hP	hR	hF	UAcc	JD	HL
With "Ripple"	0.8576	0.8309	0.8441	0.5076	0.6116	0.2246
Without "Ripple"	0.8529	0.8236	0.8380	0.4986	0.5970	0.2318

Table 5. Performance of Users Whose Inferences Are Wrong under the Regular Semantic Tree but Conforms to the Generalized Tree

	|G| = 10K	|G| = 20K	|G| = 40K	|G| = 80K
hP	0.9082	0.9046	0.9174	0.9130
hR	0.8921	0.8907	0.9037	0.9014
hF	0.9001	0.8346	0.9167	0.9072
UAcc	0.5700	0.5594	0.5901	0.5638
JD	0.7052	0.6272	0.7113	0.6522
HL	0.1494	0.1559	0.1280	0.1398

Table 6. Performance Comparison for DBLP Dataset on Different Kinds of Trees

	hP	hR	hF	UAcc	JD	HL
Generalized Semantic Tree	0.8576	0.8309	0.8441	0.5076	0.6116	0.2246
Normal Semantic Tree	0.8116	0.7992	0.8053	0.4784	0.5941	0.2355

are confident to be inferred correctly, which reduces the incorrect inference results propagating in the social network to a certain extent, so that the inference effect is improved.

7.4 Effectiveness of Generalized Semantic Tree

We also study the significance of generalized semantic tree. In a real social network, regular trees cannot fully describe the hierarchical structure between user attributes. Some users' attributes probably only meet the hierarchy constraint of the generalized semantic tree instead of the regular tree. Table 5 shows the inference effect of our MLI model on this part of users.

According to the results, we can observe that for users whose inference result is wrong under the regular semantic tree but conforms to the generalized tree, our MLI mode can achieve a high inference effect, which means that our model works well with the newly added relations of the generalized semantic tree.

Attribute correction under regular semantic tree force certain attributes to be corrected to the same branch, which are actually located on different branches of the tree, leading to incorrect inference results. This is not caused by the model but is limited by the description ability of the semantic tree. From Table 6 where vertex size is 80K and the proportion of unknown vertices is 50%, we can observe that utilizing generalized semantic tree to model the hierarchy of attributes performs significantly better.

7.5 Parameter Sensitivity Study

There are four parameters in our model. We study the effect of different parameter values and show the results in Figures 14–17. The experiments are conducted on a data set of 80K users, and users with unknown attributes accounted for 50%.

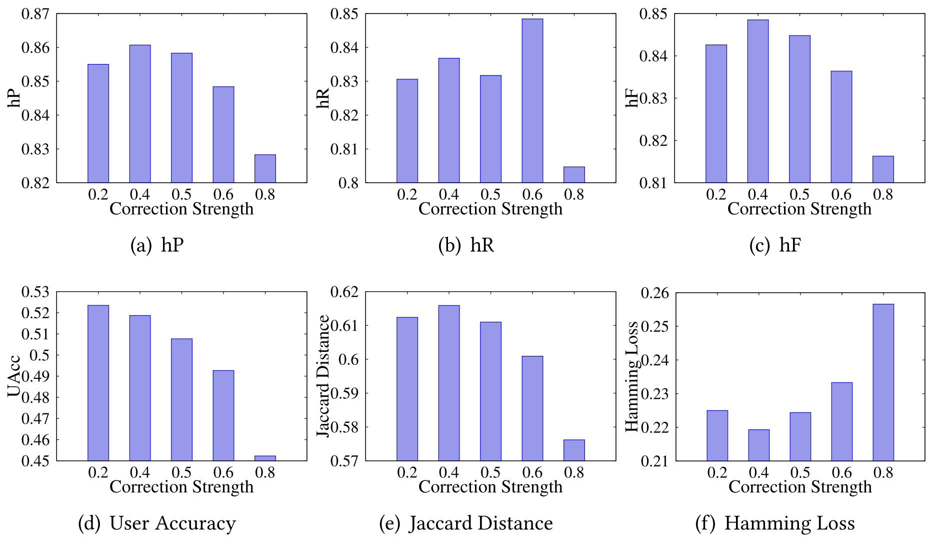


Fig. 14. Inference performance for DBLP dataset on different value of α.

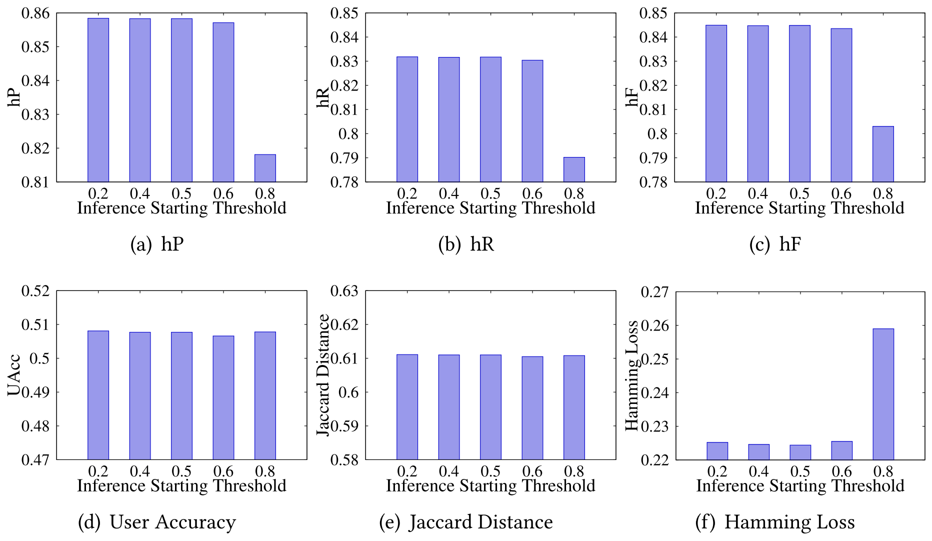


Fig. 15. Inference performance for DBLP dataset on different value of ε.

Correction Strength α is used in the attribute correction process. When the value of α is large, the result is more inclined to the hierarchy of the semantic tree, otherwise, it is more inclined to the information collected during the propagation. From the results of Figure 14, we can observe that the model can get the best effect on $\alpha = 0.4$, when the collected information and the correction can reach a balance. As the correction strength becomes larger, the inference effect becomes worse due to the inability of the information propagation model.

ε is the parameter to determine whether the user can start to infer and θ determines whether the user needs to be inferred in the next iteration. They implement the "Ripple" algorithm together. As can be seen in Figures 15 and 16, inference results are not very sensitive to these two parameters

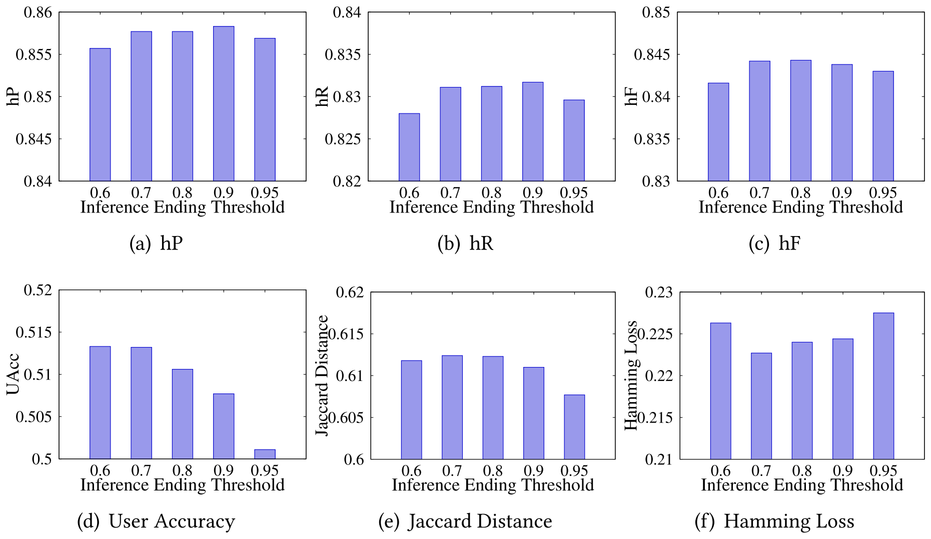


Fig. 16. Inference performance for DBLP dataset on different value of θ.

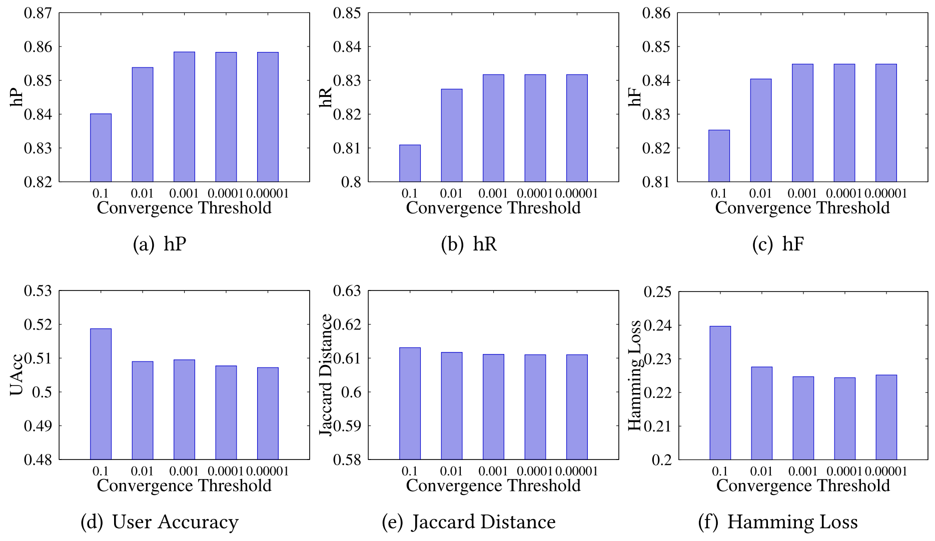
unless the value is too large. One possible reason is that the average degree of the vertices in our data set is small, resulting in the confidence of unknown vertices being either very large or small, which weakens the control ability of these two parameters. Theoretically, a larger ε means that it is more difficult for users to join the inference process, and the number of vertices processed in each iteration will be less. Meanwhile, a smaller θ means that the user can leave the inference earlier (although not enough credible information has been collected yet). Therefore, under the premise of ensuring the accuracy of inference, increasing ε and decreasing θ can improve the efficiency of the algorithm.

Figure 17 shows the inference performance of different value of σ, which controls the number of iterations. From the results, we can see that when $\sigma < 0.001$, due to the insufficient number of iterations, vertices have not collected enough information, and the inference effect of the model is limited. When $\sigma \geq 0.001$, most of the inference results have already converged, so even if the value of σ is increased, more iterations will not have a positive impact on the inference effect.

7.6 Real Case Study

In Table 7, we present the inference results of the unknown users in Figure 1, which gives a clear comparison between our method and TRW. We use this real-world example to demonstrate the effectiveness of our correction method and how MLI solves the problems mentioned in Figure 2.

For Mohammad Pourhomayoun, TRW independently collects information of each level, and the inferred fourth-level attribute is “Recognition,” because the collected fourth-level information is $w_{Localization}(MohammadPourhomayoun) = 0.17$, $w_{Classification}(MohammadPourhomayoun) = 0.21$, $w_{Recognition}(MohammadPourhomayoun) = 0.34$. But “Recognition” does not belong to the same research direction as the upper level “Network-Wireless.” With the fourth-level information, MLI can combine the hierarchical relationships and correct the inference result to “Localization.” Through TRW, the inference result of Mark L. Fowler on Level 4 is $w_{Localization}(MarkL.Fowler) = 0.30$, $w_{Bandwidth}(MarkL.Fowler) = 0.30$, which cannot be distinguished only through the information of this level. However, “Bandwidth” belongs to the “Network-qos” branch in the defined



(a) hP (b) hR (c) hF
(d) User Accuracy (e) Jaccard Distance (f) Hamming Loss

Fig. 17. Inference performance for DBLP dataset on different value of σ.

Table 7. Comparison of Inference Results by TRW and MLI

Author	Ground-truth			TRW result			MLI result		
	Level2	Level3	Level4	Level2	Level3	Level4	Level2	Level3	Level4
Mohammad Pourhomayoun	Network	Wireless	**Localization**	Network	Wireless	**Recognition**	Network	Wireless	**Localization**
Mark L. Fowler	Network	Wireless	**Localization**	Network	Wireless	**Localization Bandwidth**	Network	Wireless	**Localization**
Chikahito Nakajima	**Learning**	Image	**Recognition**		Image	Recognition	**Learning**	Image	Recognition

Table 8. Comparison of Inference Results by Common Semantic Tree and Generalized Semantic Tree

Author	Ground-truth			Common Semantic Tree			Generalized Semantic Tree		
	Level2	Level3	Level4	Level2	Level3	Level4	Level2	Level3	Level4
Xiaoou Tang	Learning	Image	**Recognition**	Learning	Image	**Segmentation**	Learning	Image	**Recognition**

semantic tree. Based on this, MLI can determine that "Localization" is more suitable for Mark L. Fowler. Due to the lack of information on the second level among Chikahito Nakajima's nearby users, TRW cannot infer any results for this level. However, MLI can find that the attributes of Level 3 and Level 4 are highly likely to be "Image-Recognition" during the correction process, so that the Level2 attribute is completed as "Learning."

In real social networks, user attributes are hierarchical; however, the existing methods are basically single-level methods. When dealing with multi-level problems, there will be problems like the TRW inference results in Table 7. Our method can revise the inference results according to the predefined semantic tree structure on the basis of hierarchically collected information, and accurately obtain users' hierarchical attributes, which can play an important role in practical applications such as target recommendation.

In addition, we also use an example in our experiment to prove the role of generalized semantic trees in practical applications. Xiaoou Tang is an expert in the field of image recognition, but in the user-defined regular semantic tree, "Recognition" is not under the subtree rooted by "Image" but under the subtree of "Speech" (because speech recognition is also one of the most popular research interests), therefore, the correct result cannot be inferred under this hierarchical structure. In the generalized semantic tree, both "Image-Recognition" and "Speech-Recognition" are the legal paths,

Table 9. Cases of Inference Failure of MLI

Author	Ground-truth			MLI		
	Level2	Level3	Level4	Level2	Level3	Level4
Murrill Szucs	Learning	Image	**Tracking**	Learning	Image	**Tracking** **Recognition** **Segmentation**
Haicheng Qu	**Computing**	**Distributed**	**Parallel**	**Learning**	**Image**	**Classification**

which enhance the expressive ability of the hierarchical structure and can also help obtain the correct results.

However, MLI does not infer user attributes accurately all the time but fails in some special cases. We analyze the reasons behind the inference failures by using the real-world examples in Table 9.

For Murrill Szucs, his ground-truth user attribute is “Computer Science-Learning-Image-Tracking.” When MLI infers his user attributes, the information collected for the bottom-level attribute is $w_{Tracking}(Murrill\ Szucs) = w_{Recognition}(Murrill\ Szucs) = w_{Segmentation}(Murrill\ Szucs) = 0.33$. Since these three attributes have the same parent attribute “Image” and do not have lower-level attributes to assist in correction, they cannot be further distinguished. As for another reason for failure, we take Haicheng Qu as an example, all the users around him are researchers in image classification, whose attributes are “Computer Science-Learning-Image-Classification.” By random walk, MLI naturally infers Haicheng Qu’s user attributes as image classification; however, his ground-truth attribute is “Computer Science-Computing-Distributed-Parallel.” Since not all users in social networks satisfy the principle of homogeneity, it also leads to the incorrect inference result by MLI.

8 CONCLUSION

We have proposed a novel model named MLI for multi-level inference problem in social networks. Previous works mainly focus on the user attributes that are at a single level. In this article, we take the semantic information implicit in the attribute hierarchy into consideration. MLI contains two procedures: (1) We use maximum entropy random walk to collect attributes from nearby users for preliminary inference; (2) to make the inference result satisfy the hierarchical constraint, we propose a correction method based on the predefined multi-level structure to conduct a cross-level correction. Since the hierarchical structure of attributes is given by the users according to the actual application scenarios, we propose the concept of generalized semantic tree, so that MLI can be applied to a wider range of scenarios. In addition, we propose a confidence model to formally describe the credibility of the attributes provided by the users to limit the spread of misinformation. Meanwhile, we design a “Ripple” algorithm, which can speed up the convergence and improve the effectiveness of inference.

Compared with the methods of classifiers, community detection, one-step random walk, node2vec, and feature propagation, experimental results on DBLP datasets have demonstrated the superior performance of our new method. MLI can not only obtain an accurate inference result under different unknown user ratios but also shorten the time used for inference.

In future work, we plan to think about how to automatically learn high-quality semantic trees from data, which can have a positive effect on real applications. In addition, advanced GNN models can mine the structural features of social networks more deeply, and applying GNNs to perform inference is our leftover future work. To further improve the inference accuracy, in addition to the graph structure, how to utilize users’ textual contents to assist inference is also worth exploring.

REFERENCES

[1] Hasitha Bimsara Ariyaratne and Dengsheng Zhang. 2012. A novel automatic hierachical approach to music genre classification. In *Proceedings of the IEEE International Conference on Multimedia and Expo Workshops*. IEEE Computer Society, 564–569. https://doi.org/10.1109/ICMEW.2012.104

[2] Jayme Garcia Arnal Barbedo and Amauri Lopes. 2007. Automatic genre classification of musical signals. *EURASIP J. Adv. Signal Process.* (2007). https://doi.org/10.1155/2007/64960

[3] Smriti Bhagat, Graham Cormode, and S. Muthukrishnan. 2011. Node classification in social networks. In *Social Network Data Analytics*. Springer, 115–148. https://doi.org/10.1007/978-1-4419-8462-3_5

[4] John D. Burger and John C. Henderson. 2006. An exploration of observable features related to blogger age. In *Proceedings of the AAAI Spring Symposium on Computational Approaches to Analyzing Weblogs*. AAAI, 15–20.

[5] John D. Burger, John C. Henderson, George Kim, and Guido Zarrella. 2011. Discriminating gender on Twitter. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'11)*. ACL, 1301–1309.

[6] Deepayan Chakrabarti, Stanislav Funiak, Jonathan Chang, and Sofus A. Macskassy. 2018. Joint label inference in networks. In *Proceedings of the World Wide Web Conference (WWW'18)*. ACM, 483–487. https://doi.org/10.1145/3184558.3186238

[7] Jiayi Chen, Jianping He, Lin Cai, and Jianping Pan. 2016. Profiling online social network users via relationships and network characteristics. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM'16)*. IEEE, 1–6. https://doi.org/10.1109/GLOCOM.2016.7842176

[8] Zhiyuan Cheng, James Caverlee, and Kyumin Lee. 2010. You are where you tweet: A content-based approach to geo-locating twitter users. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management (CIKM'10)*. ACM, 759–768. https://doi.org/10.1145/1871437.1871535

[9] Daeseon Choi, Younho Lee, Seokhyun Kim, and Pilsung Kang. 2017. Private attribute inference from Facebook's public text metadata: A case study of Korean users. *Ind. Manag. Data Syst.* 117, 8 (2017), 1687–1706. https://doi.org/10.1108/IMDS-07-2016-0276

[10] Yuxiao Dong, Yang Yang, Jie Tang, Yang Yang, and Nitesh V. Chawla. 2014. Inferring user demographics and social strategies in mobile social networks. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'14)*. ACM, 15–24. https://doi.org/10.1145/2623330.2623703

[11] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 855–864. https://doi.org/10.1145/2939672.2939754

[12] Jinyuan Jia, Binghui Wang, Le Zhang, and Neil Zhenqiang Gong. 2017. AttriInfer: Inferring user attributes in online social networks using Markov random fields. In *Proceedings of the 26th International Conference on World Wide Web (WWW'17)*. ACM, 1561–1569. https://doi.org/10.1145/3038912.3052695

[13] Rosie Jones, Ravi Kumar, Bo Pang, and Andrew Tomkins. 2007. "I know what you did last summer": Query logs and user privacy. In *Proceedings of the 16th ACM Conference on Information and Knowledge Management (CIKM'07)*. ACM, 909–914. https://doi.org/10.1145/1321440.1321573

[14] Carlos Nascimento Silla Jr. and Alex Alves Freitas. 2011. A survey of hierarchical classification across different application domains. *Data Min. Knowl. Discov.* 22, 1–2 (2011), 31–72. https://doi.org/10.1007/s10618-010-0175-9

[15] Svetlana Kiritchenko, Stan Matwin, Richard Nock, and A. Fazel Famili. 2006. Learning and evaluation in the presence of class hierarchies: Application to text categorization. In *Proceedings of the 19th Conference of the Canadian Society for Computational Studies of Intelligence (Canadian AI'06)*, Vol. 4013. Springer, 395–406. https://doi.org/10.1007/11766247_34

[16] Kamran Kowsari, Rasoul Sali, Lubaina Ehsan, William Adorno, S. Asad Ali, Sean R. Moore, Beatrice C. Amadi, Paul Kelly, Sana Syed, and Donald E. Brown. 2020. HMIC: Hierarchical medical image classification, a deep learning approach. *Information* 11, 6 (2020), 318. https://doi.org/10.3390/info11060318

[17] Rui Li, Shengjie Wang, Hongbo Deng, Rui Wang, and Kevin Chen-Chuan Chang. 2012. Towards social user profiling: Unified and discriminative influence model for inferring home locations. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'12)*. ACM, 1023–1031. https://doi.org/10.1145/2339530.2339692

[18] Rong-Hua Li, Jeffrey Xu Yu, and Jianquan Liu. 2011. Link prediction: The power of maximal entropy random walk. In *Proceedings of the 20th ACM Conference on Information and Knowledge Management (CIKM'11)*, Craig Macdonald, Iadh Ounis, and Ian Ruthven (Eds.). ACM, 1147–1156. https://doi.org/10.1145/2063576.2063741

[19] Chun-Cheng Lin, Jia-Rong Kang, and Jyun-Yu Chen. 2015. An integer programming approach and visual analysis for detecting hierarchical community structures in social networks. *Inf. Sci.* 299 (2015), 296–311. https://doi.org/10.1016/j.ins.2014.12.009

[20] Yuning Mao, Jingjing Tian, Jiawei Han, and Xiang Ren. 2019. Hierarchical text classification with reinforced label assignment. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the 9th*

International Joint Conference on Natural Language Processing (EMNLP-IJCNLP'19). Association for Computational Linguistics, 445–455. https://doi.org/10.18653/v1/D19-1042

[21] Alan Mislove, Bimal Viswanath, P. Krishna Gummadi, and Peter Druschel. 2010. You are who you know: Inferring user profiles in online social networks. In *Proceedings of the 3rd International Conference on Web Search and Web Data Mining (WSDM'10)*. ACM, 251–260. https://doi.org/10.1145/1718487.1718519

[22] Dounia Mulders, Cyril de Bodt, Johannes Bjelland, Alex Pentland, Michel Verleysen, and Yves-Alexandre de Montjoye. 2020. Inference of node attributes from social network assortativity. *Neural Comput. Appl.* 32, 24 (2020), 18023–18043. https://doi.org/10.1007/s00521-018-03967-z

[23] Felipe Kenji Nakano, Ricardo Cerri, and Celine Vens. 2020. Active learning for hierarchical multi-label classification. *Data Min. Knowl. Discov.* 34, 5 (2020), 1496–1530. https://doi.org/10.1007/s10618-020-00704-w

[24] Scott Nowson and Jon Oberlander. 2006. The identity of bloggers: Openness and gender in personal weblogs. In *Proceedings of theAAAI Spring Symposium on Computational Approaches to Analyzing Weblogs*. AAAI, 163–167.

[25] Jing Pan, Yajun Yang, Qinghua Hu, and Hong Shi. 2016. A label inference method based on maximal entropy random walk over graphs. In *Proceedings of the 18th Asia-Pacific Web Conference on Web Technologies and Applications (APWeb'16) (Lecture Notes in Computer Science)*, Vol. 9931. Springer, 506–518. https://doi.org/10.1007/978-3-319-45814-4_41

[26] 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 *Proceedings of the World Wide Web Conference (WWW'18)*. ACM, 1063–1072. https://doi.org/10.1145/3178876.3186005

[27] Marco Pennacchiotti and Ana-Maria Popescu. 2011. A machine learning approach to twitter user classification. In *Proceedings of the 5th International Conference on Weblogs and Social Media*. The AAAI Press. Retrieved from http://www.aaai.org/ocs/index.php/ICWSM/ICWSM11/paper/view/2886.

[28] Delip Rao, David Yarowsky, Abhishek Shreevats, and Manaswi Gupta. 2010. Classifying latent user attributes in twitter. In *Proceedings of the 2nd International Workshop on Search and Mining User-generated Contents (SMUC@CIKM'10)*. ACM, 37–44. https://doi.org/10.1145/1871985.1871993

[29] Emanuele Rossi, Henry Kenlay, Maria I. Gorinova, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. 2021. On the unreasonable effectiveness of feature propagation in learning on graphs with missing node features. Retrieved from https://arxiv.org/abs/2111.12128.

[30] A. Secker, M. N. Davies, A. A. Freitas, J. Timmis, and D. R. Flower. 2007. An experimental comparison of classification algorithms for the hierarchical prediction of protein function. *Prediction of Protein Function Expert Update.* 9, 3 (2007), 17–22. https://kar.kent.ac.uk/14529/.

[31] Claude E. Shannon. 1948. A mathematical theory of communication. *Bell Syst. Tech. J.* 27, 3 (1948), 379–423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

[32] Aixin Sun and Ee-Peng Lim. 2001. Hierarchical text classification and evaluation. In *Proceedings of the IEEE International Conference on Data Mining*. IEEE Computer Society, 521–528. https://doi.org/10.1109/ICDM.2001.989560

[33] Isak Taksa. 2008. David taniar: Research and trends in data mining technologies and applications. *Inf. Retr.* 11, 2 (2008), 165–167. https://doi.org/10.1007/s10791-008-9056-x

[34] Amanda L. Traud, Peter J. Mucha, and Mason A. Porter. 2011. Social structure of facebook networks. Retrieved from http://arxiv.org/abs/1102.2166.

[35] G. Valentini and M. Re. 2009. Weighted true path rule: A multilabel hierarchical algorithm for gene function prediction. In *1st Workshop on Learning from Multi-Label Data (MLD) Held in Conjunction with ECML/PKDD*.

[36] Celine Vens, Jan Struyf, Leander Schietgat, Saso Dzeroski, and Hendrik Blockeel. 2008. Decision trees for hierarchical multi-label classification. *Mach. Learn.* 73, 2 (2008), 185–214. https://doi.org/10.1007/s10994-008-5077-3

[37] Boyan Wang, Xuegang Hu, Pei-Pei Li, and Philip S. Yu. 2021. Cognitive structure learning model for hierarchical multi-label text classification. *Knowl. Based Syst.* 218 (2021), 106876. https://doi.org/10.1016/j.knosys.2021.106876

[38] Chunrong Wu, Qinglan Peng, Jia Lee, Kenji Leibnitz, and Yunni Xia. 2021. Effective hierarchical clustering based on structural similarities in nearest neighbor graphs. *Knowl. Based Syst.* 228 (2021), 107295. https://doi.org/10.1016/j.knosys.2021.107295

[39] Feihong Wu, Jun Zhang, and Vasant G. Honavar. 2005. Learning classifiers using hierarchically structured class taxonomies. In *Proceedings of the 6th International Symposium on Abstraction, Reformulation, and Approximation (SARA'05) (Lecture Notes in Computer Science)*, Vol. 3607. Springer, 313–320. https://doi.org/10.1007/11527862_24

[40] Le Wu, Yonghui Yang, Kun Zhang, Richang Hong, Yanjie Fu, and Meng Wang. 2020. Joint item recommendation and attribute inference: An adaptive graph convolutional network approach. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'20)*. ACM, 679–688. https://doi.org/10.1145/3397271.3401144

[41] Degui Xiao, Qilei Chen, and Shanshan Li. 2016. A multi-scale cascaded hierarchical model for image labeling. *Int. J. Pattern Recogn. Artif. Intell.* 30, 9 (2016), 1660005:1–1660005:22. https://doi.org/10.1142/S0218001416600053

[42] Yifan Yan and Sheng-Jun Huang. 2018. Cost-effective active learning for hierarchical multi-label classification. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18).* ijcai.org, 2962–2968. https://doi.org/10.24963/ijcai.2018/411

[43] Carl Yang, Lin Zhong, Li-Jia Li, and Luo Jie. 2017. Bi-directional joint inference for user links and attributes on large social graphs. In *Proceedings of the 26th International Conference on World Wide Web Companion.* ACM, 564–573. https://doi.org/10.1145/3041021.3054181

[44] Shuang-Hong Yang, Bo Long, Alexander J. Smola, Narayanan Sadagopan, Zhaohui Zheng, and Hongyuan Zha. 2011. Like like alike: Joint friendship and interest propagation in social networks. In *Proceedings of the 20th International Conference on World Wide Web (WWW'11).* ACM, 537–546. https://doi.org/10.1145/1963405.1963481

[45] Take Yo and Kazutoshi Sasahara. 2017. Inference of personal attributes from tweets using machine learning. In *Proceedings of the IEEE International Conference on Big Data (BigData'17)*, Jian-Yun Nie, Zoran Obradovic, Toyotaro Suzumura, Rumi Ghosh, Raghunath Nambiar, Chonggang Wang, Hui Zang, Ricardo Baeza-Yates, Xiaohua Hu, Jeremy Kepner, Alfredo Cuzzocrea, Jian Tang, and Masashi Toyoda (Eds.). IEEE Computer Society, 3168–3174. https://doi.org/10.1109/BigData.2017.8258295

[46] Guangxiang Zeng, Ping Luo, Enhong Chen, and Min Wang. 2013. From social user activities to people affiliation. In *Proceedings of the IEEE 13th International Conference on Data Mining*. IEEE Computer Society, 1277–1282. https://doi.org/10.1109/ICDM.2013.101

[47] Elena Zheleva and Lise Getoor. 2009. To join or not to join: The illusion of privacy in social networks with mixed public and private user profiles. In *Proceedings of the 18th International Conference on World Wide Web (WWW'09).* ACM, 531–540. https://doi.org/10.1145/1526709.1526781

Received 26 August 2021; revised 29 April 2022; accepted 15 June 2022