Edge Classification in Networks

Charu Aggarwal
IBM Watson Research Center
Yorktown Heights, New York 10598
charu@us.ibm.com

Gewen He
Florida State University
Tallahassee, Florida 32306
he@cs.fsu.edu

Peixiang Zhao
Florida State University
Tallahassee, Florida 32306
zhao@cs.fsu.edu

***Abstract*—We consider in this paper the edge classification problem in networks, which is defined as follows. Given a graph-structured network $G(N, A)$, where N is a set of vertices and $A \subseteq N \times N$ is a set of edges, in which a subset $A_l \subseteq A$ of edges are properly labeled a priori, determine for those edges in $A_u = A \setminus A_l$ the edge labels which are unknown. The edge classification problem has numerous applications in graph mining and social network analysis, such as relationship discovery, categorization, and recommendation. Although the vertex classification problem has been well known and extensively explored in networks, edge classification is relatively unknown and in an urgent need for careful studies. In this paper, we present a series of efficient, neighborhood-based algorithms to perform edge classification in networks. To make the proposed algorithms scalable in large-scale networks, which can be either disk-resident or stream-like, we further devise efficient, cost-effective probabilistic edge classification methods without a significant compromise to the classification accuracy. We carry out experimental studies in a series of real-world networks, and the experimental results demonstrate both the effectiveness and efficiency of the proposed methods for edge classification in large networks.**

I. Introduction

Graph and network mining algorithms have become increasingly popular in recent years because of their relevance to many applications in biological, communication, and social network analysis [1], [2]. Due in particular to the relative complexity of graph-shaped data, many graph and network mining problems can be formulated in a larger number of ways than other types of data. Likewise, most classical data mining problems such as clustering, classification, and outlier detection can be reformulated in the context of graphs and networked data as well [3], [4], [5]. The primary reason is the inherent complexity in the graph and network domains where the structural relationships (edges) are introduced between entities (vertices) [6], [7].

Consider the problem of graph classification as an example, which is generally understood as that of labeling *vertices* of graphs or networks. Specifically, given a graph $G(N, A)$ in which a subset $N' \subseteq N$ of vertices have been labeled properly, the labels of the remaining vertices in $N \setminus N'$ can be inferred using a *collective classification model* [8]. Although much work in the literature has been devoted to the problem of vertex classification [4], [9], [10], [11], little effort has been taken on the problem of *edge classification* in graphs and networks. In the edge classification problem, a subset of the edges in the graph have been affiliated with discrete labels, and it is desired to use these labels in order to infer the labels of edges where they are unspecified. In some cases, the class labels may be even numeric or continuous values. This immediately corresponds to the *edge regression* problem in networks. In this paper, we will discuss both variants for edge labeling in large-scale networked data.

The edge classification problem has numerous applications in a wide array of network-centric scenarios:

1) A social network may contain relationships of various types such as friends, families, or even rivals [12], [13], [14]. A subset of the edges may be labeled with the relevant relationships. It is desired to use the labeled edges to determine the labels on the subset of edges where they are unknown. The determination of such relationships can be useful for making targeted recommendations [15], [16];
2) An adversarial or terrorist network may contain unknown relationships between its participants although a subset of the relationships have been known already [17]. The relationships may correspond to various types of organizational characteristics of the network. The labeled relationships on the edges can thus be used to infer the labels of the edges where they are unknown in order to identify and circumvent large-scale outbreaks of cyber-attacks and crime [18];
3) In some cases, the edges may have numerical quantities associated with them corresponding to strengths of relationships, specification of ratings, likes, or dislikes [19], [20]. They can be further used to infer the numerical values on edges where such numeric strengths are unknown a priori. This variation is closed related to the regression modeling. Note that the problem of collaborative filtering is a special case of this scenario in which a user-item graph is constructed from the ratings matrix [8], [21]. Our formulation of edge regression, however, is even more general, where the relationship between various vertices can be of any type, rather than simply in the form of user-item relationships.

The problem of edge classification is generally harder than that of vertex classification. This is because vertex classification methods are primarily based on the notion of *homophily* [6], [7], which models the fact that similar vertices belonging to the same category tend to be connected within networks. However, it is generally much more difficult to apply the homophily principle and various implementations of the

structural similarity for structurally close edges. For example, consider a network where edges may correspond to different relationships, such as family, friends, and colleagues. Similar vertices within the same category will have relationships of different types and therefore, notions of *structural similarity* need to be defined more carefully in this case. In this paper, we reexamine the notion of structural similarity in order to learn a neighborhood based model between structural characteristics of the network and the labels of edges. We design exact, k-nearest neighbor based edge classification methods with both unweighted and weighted variations that can work in either fully labeled or partially labeled networks. In order to enable scalable edge classification in large-scale, dynamic networks, We further design efficient and cost-effective probabilistic classification methods that can be employed in disk-resident graphs or graph streams [22], [23]. We benchmark our edge classification methods in a series of real-world, fully or partially labeled networks, and the experimental results demonstrate that the proposed methods can achieve high-quality edge labeling results without a compromise on the efficiency and cost for edge classification, especially in large networks that are either disk-resident or stream-like.

The main contributions of this paper is summarized as follows,

1) We formulate the edge classification problem and the edge regression problem in graph-structured networks. To the best of our knowledge, this is the first work to consider classifying edges of graphs in a general way;
2) We propose a structural similarity model for edge classification. A series of edge classification methods are designed for fully labeled and partially labeled networks with the weighted Jaccard coefficient as the underlying structural proximity metric;
3) In order to support edge classification in large-scale networks that are either disk-resident or stream-like, we devise probabilistic, min-hash based edge classification algorithms that are efficient and cost-effective without comprising the classification accuracy significantly;
4) We carry out extensive experimental studies in different real-world networks, and the experimental results verify the efficiency and accuracy of our edge classification techniques.

This paper is organized as follows. The remainder of this section introduces the related work. Section II details the edge classification model and a straightforward algorithm to perform the edge classification. We also show how this approach can be extended for the edge regression modeling. In section III, we present how to make the edge classification approach more efficient for large-scale networks. The experimental studies and main results are reported and discussed in section IV. Finally, the conclusions are summarized in section V.

A. Related Work

The problem of vertex classification has been studied in the social network and graph mining literature, and especially for relational data in the context of *label or belief propagation* [24], [25], [26]. A detailed survey of such methods can be found in [27]. These propagation based techniques can also be used as a tool for semi-supervised learning with both labeled and unlabeled examples [28]. A method that used link-based similarity for vertex classification has been proposed [29]. Recently, similar techniques were used in the context of blogs as well [30]. Another work [31] discussed the problem of label acquisition in the context of collective classification. Note that this is an important problem because such basic labels are necessary information for vertex classification. A method to perform collective classification for email speech acts has been proposed in [32]. It has been shown that the analysis of relational aspects of emails (such as emails in a particular thread) significantly improves the classification accuracy. It has also been shown in [33], [34] that the use of graph structures during categorization can improve the classification accuracy of web pages. It is worth noting that all these existing methods are designed for the problem of vertex classification, whereas our goal is to study the problem of edge classification, which is significantly more difficult because the straightforward notion of vertex-wise homophily cannot be directly applied for edge classification in networks.

A number of recent methods have discussed the inference of ties in different network applications by leveraging the domain-specific characteristics of graph-structured data. In social networks, link prediction methods for the signs of edges, such as friends or enemies, trust or distrust, have been proposed [35], [36], [37], [38]. These methods are primarily based on the *behavior relation interplay (BRI)* model that leverages behavioral evidence within social networks to infer social interactions and exploits learned relations to tie users's behavior. The work in [39] has studied the problem of mining advisor-advisee relationships in a research publication network. However, this approach is completely unsupervised, and relies primarily on the domain-specific characteristics of the networks for inference. Similarly, the work in [40] tried to infer relationships in an email network by using the behavioral characteristics of the end-user interactions. Finally, the work in [41] adopted a partially supervised approach to inferring edge types, although it still relied on domain-specific data available in the attributes and correlation factors of networks. Note that these factors need to be instantiated in a domain-specific way for a particular network, and they require domain-specific insight of the network or relationships at hand. As a result, these methods cannot exactly solve the edge classification problem as discussed in this paper, since they are designed under different, more specialized, assumptions. Our edge classification model, however, is quite general, and thus can work within any arbitrary network in a variety of settings without specific assumptions about the problem domains or characteristics of graphs.

II. The Edge Classification Model

In the following, we will provide a formal model for edge classification in networks. We assume that we have an undirected network $G = (N, A)$, where N denotes the set of vertices, A denotes the set of edges, and $n = |N|$, $m = |A|$. A subset of edges in A, denoted by A_l, are associated with edge labels. The label of the edge (u, v) is denoted by l_{uv}. The remaining subset of unlabeled edges is denoted by $A_u = A \setminus A_l$. For ease in notation, we will assume binary edge labels drawn from $\{-1, 1\}$, though our proposed model and algorithms can be generalized in a straightforward way to address the k-way edge classification problem by applying the standard one-against-one or one-against-all procedures [42]. Furthermore, while we consider the edge classification problem for the case of undirected networks, the proposed approaches can easily be generalized to directed networks as well.

To this end, the edge classification problem can be formally defined as follows.

Definition 1 (Edge Classification). *Given an undirected network $G = (N, A)$, and a set $A_l \subseteq A$ of labeled edges, where each edge $(u, v) \in A_l$ has a binary label $l_{uv} \in \{-1, 1\}$, the edge classification problem is to determine the labels for the edges in $A_u = A \setminus A_l$.* □

In some scenarios, it is also possible for the edges in A to be labeled with numeric quantities. As a result, edge classification turns out to be the *edge regression* problem in networks, defined as follows,

Definition 2 (Edge Regression). *Consider an undirected network $G = (N, A)$, and a set $A_l \subseteq A$ of edges, each of which is annotated by numerical labels, denoted by $l_{uv} \in \mathbb{R}$. The edge regression problem is to determine the numerical labels for the edges in $A_u = A \setminus A_l$.* □

In the sequel, we will discuss a neighborhood-based model for edge classification with a series of exact solutions. Then, we will extend the model and solutions for edge regression in networks. We will also show how to make the edge classification process more efficient and scalable on large-scale networks with a use of a probabilistic approach.

A. Structural Similarity Model for Edge Classification

We will propose a structural similarity model for edge classification. This approach shares a number of similarities with the nearest neighbor classification, except that the structural similarity of one edge to another is much harder to define in the context of a network. Consider an edge $(u, v) \in A_u$, which needs to be classified within the network $G = (N, A)$. In order to determine the edges which are similar to (u, v), the following steps will be executed step-by-step:

1) The top-k most similar vertices to u, denoted by $S(u) = \{u_1 \dots u_k\}$, are first determined based on a structural similarity measure discussed slightly later. Note that $u \in S(u)$ because u is considered the most similar vertex to itself. This structural similarity measure will further include the use of the class distribution of the top ranked relevant vertices;

2) The top-k most similar vertices to v, denoted by $S(v) = \{v_1 \dots v_k\}$, are determined based on the same structural similarity measure as in the case of vertex u;

3) The set of edges in $A_l \cap [S(u) \times S(v)]$ are selected and the dominant class label of these edges is assigned as the relevant edge label for the edge (u, v).

Next, we describe how the most similar vertices to a given vertex $u \in N$ are determined in a network. Here the key step is to define a pair-wise similarity function for vertices, and we consider in this paper Jaccard coefficient for vertex-wise similarity quantification. Jaccard coefficient (or Jaccard metric) has been extensively used in many domains including natural language processing [43] and web sciences [44]. In [45], the authors introduced *shingles* and *min-wise independent permutations* to sketch Jaccard coefficient for web documents. [46] gave a way of sketching arbitrary non-negative vectors in a way that preserves their weighted Jaccard metrics. In this paper, we adopt the *weighted Jaccard coefficient* considering the edges belonging only to the same edge label. Formally, let $I^-(u) \subseteq N$ be the set of vertices incident on the vertex u belonging to the edge label -1, and let $I^+(u) \subseteq N$ be the set of vertices incident on u belonging to the edge label 1. The Jaccard coefficient of vertices u and v, $J^+(u, v)$, on the positive edges (bearing the edge label 1) is defined as follows:

$$J^+(u, v) = \frac{|I^+(u) \cap I^+(v)|}{|I^+(u) \cup I^+(v)|} \tag{1}$$

The Jaccard coefficient always lies in the range $(0, 1)$ with higher values being more indicative of vertex similarity. Furthermore, since the Jaccard coefficient $J(u, u)$ of a vertex u with itself is 1, the top-k most similar vertices to a given vertex u always contain u itself. Note that if both u and v do not have any positive edges, then the Jaccard coefficient is set to 0 by default.

The Jaccard coefficient on the negative edges can be defined in an analogous way, as follows:

$$J^-(u, v) = \frac{|I^-(u) \cap I^-(v)|}{|I^-(u) \cup I^-(v)|} \tag{2}$$

As a result, the similarity between vertices u and v can be defined by a weighted average of the values of $J^+(u, v)$ and $J^-(u, v)$. What weights should be used for averaging these two values? The most effective way to achieve this goal is to use the fraction of vertices belonging to the two classes as follows. The fraction of vertices incident on u belonging to the edge label 1 is given by $f_u^+ = \frac{|I^+(u)|}{|I^+(u)|+|I^-(u)|}$. Similarly, the value of $f^-(u)$ is defined as $(1 - f^+(u))$. The average fraction across both vertices is defined by $f^+(u, v) = (f^+(u) + f^+(v))/2$, and $f^-(u, v) = (f^-(u) + f^-(v))/2$. It is easy to verify that $f^+(u, v) + f^-(u, v) = 1$. Therefore, the overall Jaccard similarity, $J(u, v)$, of the vertices u and v is defined

by the weighted average between the corresponding similarity values on the positive and negative edges, respectively:

$$J(u,v) = f^+(u,v) \cdot J^+(u,v) + f^-(u,v) \cdot J^-(u,v) \quad (3)$$

This similarity function is the key to determining the classification of unlabeled edges in networks.

An important variation of the edge classification process is that all edges are not given an equal weight in the process of determining the dominant edge label. Consider the labeled edge (u_r, v_q), where $u_r \in S(u)$, and $v_q \in S(v)$. The weight of the label of the edge (u_r, v_q) can be set to $J(u, u_r) \times J(v, v_q)$ in order to ensure greater importance of edges, which are more similar to the target vertices u and v. It is also interesting to note that the above approach can easily be extended to the k-way case by defining a separate Jaccard coefficient on a class-wise basis, and then aggregating in a similar way, as discussed above.

B. Handling Sparse Labelings

In real-world cases, the edges are often labeled rather sparsely, which makes it extremely difficult to compute similarities with only the labeled edges. In such cases, a separate component of the similarity, $J^0(u,v)$, between vertices u and v is considered only with unlabeled edges, just like $J^+(u,v)$ and $J^-(u,v)$ which are the Jaccard coefficients with positive and negative edges, respectively. Instead of using Equation 3 to compute the integrated similarity value, the following formula is used to accommodate unlabeled edges:

$$J(u,v) = f^+(u,v) \cdot J^+(u,v) + f^-(u,v) \cdot J^-(u,v) + \mu J^0(u,v) \quad (4)$$

Here, μ is a discount factor less than 1, which is used to prevent excessive impact of the unlabeled edges on the similarity computation. This approach can be viewed as a variation of a semi-supervised approach, which is inherently designed to handle sparsely labeled data.

C. Numerical Class Variables

The case of numerical class variables in edge regression is similar to that of binary or categorical class variables in edge classification, except that the vertex similarities and the class averaging steps are computed differently in order to account for the numerical class variables. In this case, a simplified approach is adopted to compute the similarity between vertices. Let $I^+(u)$ be the set of all vertices incident on a particular vertex u, such that for each $v \in I^+(u)$, the edge (u,v) has a numerical label whose value is greater than the average of the labeled edges incident on u. The remaining vertices incident on u, whose labeled edge values are below the average, are put in another set, $I^-(u)$. This way, the similarity values $J^+(u,v)$, $J^-(u,v)$, and $J(u,v)$ can be defined in an analogous way according to the previous binary edge classification case. We remark that the only difference in the similarity computation between the classification and regression cases is in terms of how $I^+(u)$ and $I^-(u)$ are defined and quantified.

In order to determine the numeric label of an edge (u,v), a similar procedure is employed as in the previous case. The first step is to determine the closest k vertices $S(u) = \{u_1 \ldots u_k\}$ to u, and the closest k vertices $\{v_1 \ldots v_k\}$ to v with the use of the aforementioned Jaccard similarity function. Then, the numeric labels of the edges in $A_l \cap [S(u) \times S(v)]$ are averaged. As in the case of edge classification, different edges can be given different weights in the regression process. Consider the labeled edge (u_r, v_q), where $u_r \in S(u)$, and $v_q \in S(v)$. The weight of the label of edge (u_r, v_q) can be set to $J(u, u_r) \times J(v, v_q)$ in order to ensure greater importance of edges, which are more similar to the target vertices u and v.

III. Efficient Probabilistic Algorithms

The main problem with the exact approaches discussed in the previous section is that they work most efficiently when the entire graph is memory-resident. However, when the graph is very large and disk-resident, the computation of par-wise vertex similarity is likely to be computationally expensive because all the adjacent edges of different vertices need to be accessed, thus resulting in a lot of random accesses to hard disks. In many real-world dynamic cases, the graph where edge classification is performed is no long static but evolving in a fast speed, which, without loss of generality, is typically modeled in the form of a graph stream [22], [47], [23]. In such cases, the graph cannot even be materialized on disks and it is impossible to examine the entire graph multiple times. Therefore, a natural question arises, "*how can the proposed edge classification methods be implemented with only succinct, memory-resident data structures in support of efficient edge classification in large-scale networks that are either disk-resident or stream-like?*"

In order to achieve this goal, we will consider a probabilistic, min-hash based approach [45], which can be applied effectively for both disk-resident graphs and graph streams. In this min-hash approach, the core idea is to associate a succinct data structure, termed *min-hash index*, with each vertex, which keeps track of the set of adjacent vertices. Specifically, the positive, negative, and unlabeled edges (and their incident adjacent vertices) of a given vertex are handled separately in different min-hash indexes. Note that the min-hash indexes for the unlabeled edges need to be maintained only in the case where they are used in the augmented similarity function of Equation 4. Otherwise, it is sufficient to maintain the min-hash indexes for only the positively and negatively labeled edges. Henceforth, we will assume each edge is in the form $\langle u, v, l_{uv} \rangle$. For unlabeled edges, the value of l_{uv} is set to '?'. As a result, in the binary class scenario, the value of l_{uv} can be $+1$, -1, or '?'.

In this probabilistic approach, we use d mutually independent min-wise hash functions $f_1(\cdot), \ldots f_d(\cdot)$, which are to create the min-hash index at each vertex of the network. As we will see later, the value of d regulates the trade-off between accuracy and storage efficiency. Each hash function $f_r(\cdot)$ $(1 \leq r \leq d)$ takes as its argument a vertex identifier and maps the identifier to a random value in $(0, 1)$. The basic

idea of this min-hash approach is to implicitly create a sort order among the vertices with the use of each hash function. Because these d hash functions are independent of one another, this creates as a result d different and mutually independent sort-orders.

Now, let us examine a pair of vertices (u, v) and consider all the vertices incident on u and v, respectively. For the purpose of argument, let us focus only on edges corresponding to the positive label (For edges with the negative label, similar conclusions can be made). What is the probability that the top-ranked vertices incident on u are the same as the top-ranked vertices incident on v? This is equal to the probability that a vertex in $I^+(u) \cap I^+(v)$ is top ranked from the set $I^+(u) \cup I^+(v)$. This probability is exactly $\frac{|I^+(u) \cap I^+(v)|}{|I^+(u) \cup I^+(v)|}$, which is the same as the Jaccard coefficient $J^+(u, v)$ on the positive edges. Therefore, by examining d mutually independent sort orders and estimating the *fraction* of them in which the top-ranked vertices are the same, one can *estimate* the Jaccard coefficient similarity efficiently.

How do we estimate the fraction of matching vertices? For the vertex pair (u, v) and a hash function $f_r(1 \leq r \leq d)$, we determine the vertex $u_r \in I^+(u)$ for which $f_r(u_r)$ is with the least value among all vertices of $I^+(u)$. In other words, we have:

$$u_r = \text{argmin}_{w \in I^+(u)} f_r(w) \ \forall r \in \{1 \ldots d\} \tag{5}$$

Therefore, u_r is the r-th *min-hash index* among all the vertices incident on u with a positive edge. The corresponding *min-hash value* is $f_r(u_r)$. Note that a min-hash index and min-hash value pair $(u_r, f_r(u_r))$ exists for each value of $r \in \{1 \ldots d\}$. Similarly, we can determine all the min-hash indices and min-hash values for the vertex v:

$$v_r = \text{argmin}_{w \in I^+(v)} f_r(w) \ \forall r \in \{1 \ldots d\} \tag{6}$$

Once the min-hash indices $u_1 \ldots u_d$ of the vertex u and the min-hash indices $v_1 \ldots v_d$ of the vertex v have been determined, the Jaccard coefficient $J^+(u, v)$ can be estimated using the fraction of the d min-hash indexes for which u_r and v_r are identical:

$$J^+(u, v) \approx Q^+(u, v) = \frac{\sum_{r=1}^{d} \delta(u_r = v_r)}{d} \tag{7}$$

Here $\delta(\cdot)$ is an indicator function, which takes on the value of 1 when u_r and v_r are the same, and 0 otherwise. The values of $J^-(u, v)$ and $J^0(u, v)$ can be computed in similar ways using negative and unsupervised edges, respectively.

In the following description, we will assume edge classification is performed in the most challenging case where the edges are received in the form of a graph stream, although this probabilistic approach can also be applied to disk-resident graphs. This is because the approach requires only one-pass exploration of the graph, and thus can be used for either disk-resident graphs or graphs streams. In either of these scenarios, it is assumed that the edges of the graph are processed sequentially, and the goal is to *dynamically* maintain the min-hash structures for the incident vertices of positive, negative, and unlabeled edges that stream in. Assume that the three relevant min-hash data structures for the positive, negative and unlabeled edges are denoted by $\mathcal{M}^+$, $\mathcal{M}^-$, and $\mathcal{M}^0$, respectively. Each data structure contains a set of $n \cdot d$ pairs of min-hash indexes and values, where n is the number of vertices, and d is the number of min-wise hash functions. Namely, there are d such pairs of min-hash indexes and values maintained for each of the n vertices of the graph. In order to dynamically maintain such min-hash data structures, the updates are performed adaptively as edges stream in. For each incoming edge of the form $\langle u, v, l_{uv} \rangle$, the following steps are executed:

1) The label of the incoming edge is checked first. Depending on the incoming edge label, exactly one of the min-hash structures corresponding to $\mathcal{M}^+$, $\mathcal{M}^-$, or $\mathcal{M}^0$ is updated. For the purpose of description, we consider the case where the incoming edge is a positive edge ($l_{uv} = +1$). Namely, $\mathcal{M}^+$ needs be updated and maintained dynamically, while updates for $\mathcal{M}^-$ or $\mathcal{M}^0$ can be performed in an analogous way;
2) The values of $f_1(u) \ldots f_d(u)$ and $f_1(v) \ldots f_d(v)$ are computed. For the r-th min-hash pair for the vertex u, the min-hash value $f_r(v)$ at its incident vertex v is retrieved. If $f_r(v)$ is less than any of the d min-hash values stored for u, the pair $(v, f_r(v))$ is inserted among the top-d values of u. Similarly, if $f_r(u)$ is less than any of the the d min-hash values stored for v, the pair $(u, f_r(u))$ is inserted among the top-d values of v;
3) The number of edges with each label are maintained separately for each vertex. Therefore, depending on the label l_{uv} of the incoming edge, the number of edges of that type is incremented by 1 for each of the incident vertices u and v. This meta-information is important to facilitate the eventual computation of Equations 3 and 4.

Note that these operations are very simple and can be executed in a straightforward way over the course of the graph stream. Furthermore, for a graph with n vertices, this approach requires $O(n \cdot d)$ space, where the parameter d is a constant regulating the trade-off between accuracy and space-efficiency. As we will see in Section IV, very accurate classification results can be obtained with small values of d.

Next, we describe the process of classifying an unlabeled edge (u, v) with the use of the min-hash index. The first step is to determine the top-k closest vertices $u_1 \ldots u_k$ and $v_1 \ldots v_k$ to u and v, respectively, with the use of the min-hash index. The similarity over the positive edges is computed using the portion $\mathcal{M}^+$ of the min-hash data structure according to Equation 7. The similarity on the negative and unsupervised edges can be computed using $\mathcal{M}^0$ and $\mathcal{M}^-$, respectively, in an analogous way. The similarities over the various types of edges can be combined using either Equation 3 or Equation 4, depending on whether or not unlabeled links are used in the similarity computation. After the vertices in $S(u)$ and $S(v)$ have been determined, the dominant class labels of the edges in $A_l \cap [S(u) \times S(v)]$ are identified accordingly. The main

technical problem here is that the edge set A_l is no longer explicitly maintained by the min-hash data structure. However, for each vertex u, we do maintain the min-hash edges in $\mathcal{M}^+$, $\mathcal{M}^0$ and $\mathcal{M}^-$, respectively. Among these labeled edges, we determine the ones, which intersect with $S(u) \times S(v)$. In the end, the dominant label among these edges is reported as the relevant edge label of the edge (u, v). The main advantage of the min-hash data structures is that they are compact, space-efficient, and thus can be safely maintained in main memory. As a result, both the updates to the min-hash data structures and the whole edge classification process can be performed efficiently based only on a series of memory-resident operations.

The effectiveness of this probabilistic edge classification approach is dependent on the accuracy of the similarity computation, which are estimated with the use of the min-hash data structures. Consider the similarity value $J^+(u, v)$, which is computed using Equation 7. Note that this provides only an approximate similarity computation between vertices. Therefore, it is instructive to bound the accuracy of the approximation in terms of the number d of min-wise hash functions. Specifically, we can show and prove the following bounds:

Lemma 1 (Lower Tail Bound). *For any $\delta \in (0, 1)$, Equation 7 provides an approximate value $Q^+(u, v)$ of the Jaccard coefficient, which lies outside a factor $(1-\delta)$ of the true value $J^+(u, v)$ with the following probability:*

$$\Pr(Q^+(u,v) < (1-\delta) \cdot J^+(u,v)) \leq exp(-d \cdot J^+(u,v) \cdot \delta^2/2) \quad (8)$$

Proof: Equation 7 is replicated here as follows:

$$J^+(u,v) \approx Q^+(u,v) = \frac{\sum_{r=1}^{d} \delta(u_r = v_r)}{d}$$

$$d \cdot J^+(u,v) \approx d \cdot Q^+(u,v) = \sum_{r=1}^{d} \delta(u_r = v_r)$$

The second summation is the sum of d i.i.d. Bernoulli variables. This particular form of the summation can be directly used in conjunction with the Chernoff bound [48]. Note that each element $\delta(u_r = v_r)$ in the aforementioned summation is a Bernoulli random variable, which takes on the value of 1 with the probability $J^+(u, v)$. Therefore, we can directly use the lower-tail Chernoff bound on the summation of these elements. The application of the lower-tail Chernoff bound results in the following condition:

$$\begin{aligned} & \Pr(d \cdot Q^+(u,v) < (1-\delta) \cdot d \cdot J^+(u,v)) \\ \leq \quad & \exp(-d \cdot J^+(u,v) \cdot \delta^2/2) \end{aligned}$$

Namely,

$$\Pr(Q^+(u,v) < (1-\delta) \cdot J^+(u,v)) \leq \exp(-d \cdot J^+(u,v) \cdot \delta^2/2)$$

■

It is easy to see that this probabilistic lower bound tightens with an increasing value of d. The probability of a given error level reduces exponentially fast with an increasing number of min-wise hash functions, d. Therefore, one can typically obtain negligible probability values for a modest number of hash functions. This reflects the trade-off between the space requirement and the edge classification accuracy.

The aforementioned condition establishes the lower-tail bound. One can obtain a similar upper-tail bound for small values of δ, as follows.

Lemma 2 (Upper Tail Bound). *For any $\delta \in (0, 2 \cdot e - 1)$, Equation 7 provides an approximate value $Q^+(u, v)$ of the Jaccard coefficient, which lies outside a factor $(1 + \delta)$ of the true value $J^+(u, v)$ with the following probability.*

$$\Pr(Q^+(u,v) > (1+\delta) \cdot J^+(u,v)) \leq exp(-d \cdot J^+(u,v) \cdot \delta^2/4) \quad (9)$$

Proof: The proof of this lemma is similar to that of Lemma 1. As in the previous case, the expression $d \cdot Q^+(u, v)$ can be expressed as a sum of d independent Bernoulli random variables. In this case, the upper-tail Chernoff bound can be used in order to obtain the desired result. ■

These theoretical results also hold for the value of $J^-(u, v)$. As a result, the overall probabilistic similarity computation is robust. This approach can be extended to numerical labels as well. However, in the case of numerical labels, a simplification is that similarities between vertices are computed using the Jaccard coefficient on all the edges rather than separately computing the Jaccard coefficient on the positive, negative, and unlabeled edges, and then combining them. Such an approach can provide a reasonable approximation in many settings where numeric values are assigned upon edges of the networks.

IV. Experimental Results

In this section, we report our experimental studies to demonstrate the effectiveness and efficiency of our proposed edge classification methods in real-world networks. We will consider a variety of effectiveness and efficiency metrics on diverse networked data sets to show the broad applicability of our methods. All our experiments were carried out on a desktop PC with Intel Quad Core 3.20GHz CPU and 8GB memory running Windows 7 operating system. All the methods, including both exact and their probabilistic counterparts, were implemented in C++.

A. Datasets

We consider in our experimental studies four large-scale online social networks where each edge of networks has been properly labeled. The first three networks contain binary edge labels[1], while the fourth network contains multiple edge labels[2]. The details of these networks are as follows,

[1]Datasets are available at http://snap.stanford.edu

[2]The dataset is available at http://socialcomputing.asu.edu/datasets/YouTube

1) **Epinions** is a popular product review website consisting of users as vertices and user interactions, such as review ratings, as edges. The binary label *trust* (represented as 1) or *distrust* (represented as 0) is attached to each edge denoting if a user u trusts another user v or not. This network comprises $119,217$ vertices and $841,000$ edges, of which 85% have the edge label 1, and $80,668$ vertices have at least one incident edges;
2) **Slashdot** is a news website with users as vertices and friendship relations as edges. A user u may like another user v's comments so that there is an edge label 1 on the edge (u, v). Otherwise, the edge label is 0. There are $82,144$ vertices and $549,202$ edges in this graph. There are 77.4% edges with the edge label 1, and $70,284$ vertices have at least one incident edges;
3) **Wikipedia** includes $7,118$ users and an labeled edge indicates that a user votes positively (labeled as 1) or negatively (labeled as 0) for another user toward a promotion as Wikipedia administrators. There are $7,118$ vertices and $103,747$ edges in this graph, while 78.7% edge labels are positive, and $2,794$ vertices have at least one incident edges;
4) **Youtube** contains a set of $15,088$ users as vertices and $7,595,273$ user interactions as edges in the Youtube website. The user interactions are first categorized into 5 different types, such as *contact*, *co-contact*, *co-subscription*, *co-subscribed*, and *favorite*, and for each different interaction type, there exists a series of *intensity* values denoting the strengths of interactions, thus resulting in 107 edge labels in total.

In order to evaluate our edge classification methods on partially labeled networks, for each of the aforementioned datasets, we randomly select a percent t of all its edges ($t = 50\%$ by default) and eliminate the edge labels, thus generating a corresponding graph whose edges are sparsely labeled. These data sets are drawn from fairly diverse settings in order to show the generality of our approach across various problem settings. We will demonstrate that our edge classification techniques can still be applied in such sparsely labeled networks.

B. Evaluation Methods

In this paper, we devise edge classification methods with a series of variations: *exact* algorithms compute the Jaccard coefficient similarity between vertices exactly (Equation 3 and Equation 4), while *probabilistic* algorithms estimate the similarity based on min-hash structures (Equation 7). Furthermore, when selecting the dominant class label from within the edge set $A_l \cap (S(u) \times S(v))$ as the edge label for the edge (u, v), we have two options for the existing edge labels that can be either *weighted* or *unweighted*. Meanwhile, the graph where edge classification is performed can be either *fully labeled* if each edge has an explicit edge label, or *partially labeled* otherwise (the exact computation is thus based on Equation 4). As a result, we consider six different edge classification algorithms in the experimental studies,

1) **ExtUF:** this is an exact edge classification method (Equation 3) on a fully labeled graph, where edge classification is based on an unweighted selection of the dominant label in the vertex neighborhood;
2) **ProbUF:** this is a probabilistic algorithm based on min-hash indexes in a fully labeled graph. The option of unweighted selection for the dominant edge label is enabled;
3) **ExtWF:** this is an exact edge classification method on a fully labeled graph, where edge classification is based on weighted selection of the dominant label in the vertex neighborhood;
4) **ProbWF:** this is a probabilistic algorithm based on min-hash indexes in a fully labeled graph. The option of weighted selection for dominant edge label is enabled;
5) **ExtWP:** this is an exact edge classification method (Equation 4) on a partially labeled graph, where edge classification is based on weighted selection of the dominant label in the vertex neighborhood;
6) **ProbWP:** this is a probabilistic algorithm based on min-hash in a partially labeled graph. The option of weighted selection for dominant edge label is enabled.

C. Evaluation Metrics

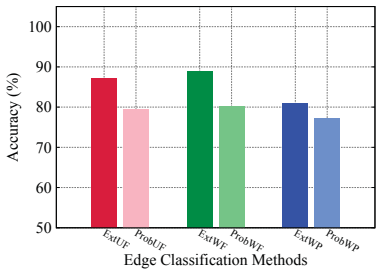
In order to evaluate different edge classification methods in real-world networks, we consider the following evaluation metrics in our experimental studies:

- **Accuracy:** we randomly select $1,500$ edges from each network dataset and examine the average accuracy of classification for these edges *w.r.t.* the true labels provided in the network;
- **Time:** we gauge the average time consumed for edge classification for the $1,500$ edges selected randomly from within the networks;
- **Space:** as opposed to the exact methods that need explicitly materialize the whole networks for edge classification, the probabilistic methods build space-efficient min-hash indexes, which are supposed to be succinct in size and memory-resident. We record the total space cost (in megabytes) of min-hash structures in the probabilistic methods, which need not store the entire graphs for edge classification.

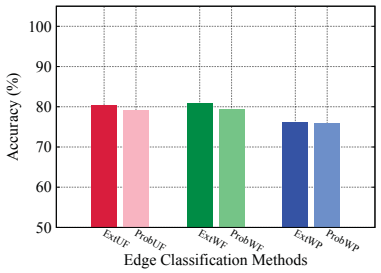
Note that these metrics are often in trade-off with one another. Therefore, exploring all of them provides an understanding of our edge classification approaches in a comprehensive way.

D. Parameter Settings

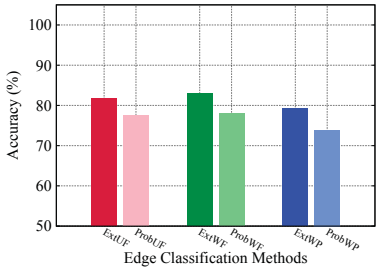
There also exist a series of algorithmic parameters for our edge classification methods. In the following, we report their default values (unless otherwise stated in the experimental section): (1) k: the number of most similar incident vertices, $|S(u)|$, of a vertex u, and the default value of k is 100; (2) μ: the discount factor for the unlabeled edges in Equation 4, and the default value of μ is 0.5; (3) d: the number of min-wise hash functions adopted in the min-hash structures, and the default value of d is 20; (4) t: the percentage of unlabeled



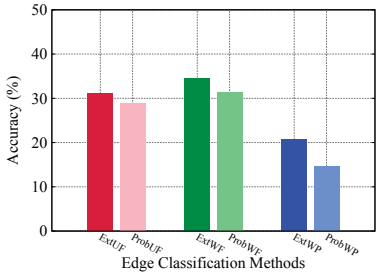
(a) Epinions



(b) Slashdot



(c) Wikipedia



(d) Youtube

Fig. 1: Edge classification accuracy of six different methods on four real-world networks.

edges in a partially labeled graph, and the default value is 50%. In our experimental studies, we will also examine the performance of the proposed edge classification methods *w.r.t.* different parameter settings.

E. Performance Results

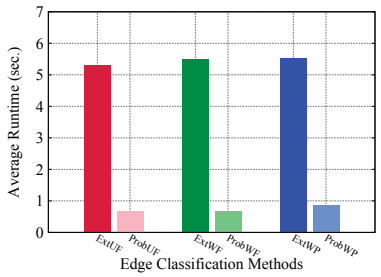
In this section, we report the main results and findings in our experimental studies for edge classification in real-world networks. We consider six different edge classification algorithms on four real-world networks, and examine the accuracy for edge classification, together with time and space cost of different algorithms. We also report how our proposed methods can be fine tuned by regulating the algorithmic parameters.

First of all, we examine the classification accuracy by applying different edge classification methods in four networks, and the results are illustrated in Figure 1. We notice that the proposed k-nearest neighbor based edge classification method can offer high-accuracy classification results in real-world networks. In binary-label networks including Epinion, Slashdot, and Wikipedia, the classification accuracy is at least 74% for all methods. In the multi-label network, Youtube, the classification accuracy is in the range of $[14\%, 35\%]$, which is still high considering there exist 107 edge labels to be classified in the network.

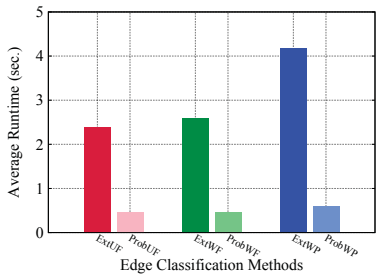
We then compare the relative performance of different edge classification algorithms, and recognize the following interesting findings. (1) In all the four networks, the weighted methods (ExtWF and ProbWF) have higher classification accuracy than the unweighed counterparts (ExtUF and ProbUF), indicating that the consideration of edge weights for the assignment of edge labels is beneficial during edge classification. Meanwhile, the classification methods in fully labeled graphs have higher accuracy than the corresponding methods in partially labeled graphs (ExtWP and ProbWP), because more edge label information within the neighborhood of vertices can be employed for similarity computation and edge classification, while unlabeled edges are less indicative in edge classification. (2) The exact methods (ExtUF, ExtWF, and ExtWP) are consistently more accurate than their probabilistic counterparts (ProbUF, ProbWF, and ProbWP). However, the gaps of classification accuracy are not significant, which are within 8% across all four networks, and on the Slashdot network, this accuracy gap is only within 2%. It indicates that the probabilistic methods can provide edge classification results with very close accuracy to the exact methods in real-world networks. However, probabilistic methods are significantly faster and more space-efficient, making themselves as practical and feasible edge classification solutions in large-scale, dynamic networks, as will be demonstrated in the following experimental studies. The fact that the accuracy of probabilistic methods does not deteriorate significantly is important because it implies that these probabilistic approaches can be used in a variety of network settings.

We then consider the runtime cost of different edge classification methods in real-world networks, and the experimental results are illustrated in Figure 2. It can be witnessed that, for exact edge classification methods (ExtUF, ExtWF, ExtWP), the average time of classifying an edge is in several seconds, except for the Wikipedia network, where exact methods can accomplish the classification of one edge within 0.12 seconds because that dataset is fairly small. However, in the largest Youtube network, it takes more than 10 seconds to classify an edge by average, which is very time-consuming. It indicates that, the exact methods cannot be used as efficient, realtime edge classification solutions, especially in large networks. This can be a significant problem in real-world, large-scale settings.

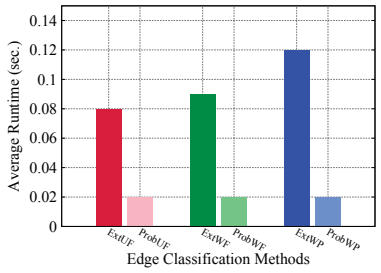
On the other hand, all the probabilistic methods (ProbUF, ProbWF, ProbWP) can successfully classify every edge almost in real time (within 0.5 seconds even in the largest



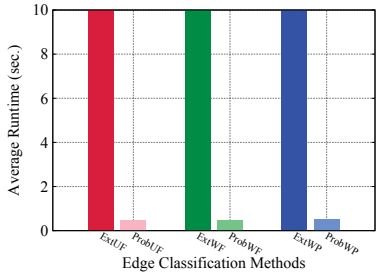
(a) Epinion



(b) Slashdot



(c) Wikipedia



(d) Youtube

Fig. 2: Average runtime cost of six different methods on four real-world networks.

Youtube network). In comparison with their exact counterparts, the probabilistic edge classification methods have achieved between 4 and 34 times speedup across all four real-world networks. This is mainly because the probabilistic methods take advantage of min-hash structures for Jaccard coefficient estimation, which can be achieved efficiently (in $O(d)$ time, where d is the number of min-wise hash functions).

An important observation is that relatively small values of d suffice to achieve robust classification results, and there is often little need for selecting large values of d. Another interesting observation is that, the runtime cost for different probabilistic methods are very close. That is, in the ProbWF method, the consideration of edge weights in edge classification does not take up significant time, and the time for ProbWP to build min-hash structures for unlabeled edges is also marginal, compared with the majority time consumed for min-hash based computation in probabilistic methods. This implies that one can often use the best probabilistic methods in different networked settings with little additional expense in terms of the running time for edge classification.

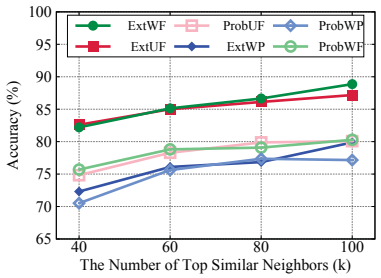
TABLE I: The overall memory consumption of the probabilistic edge classification method, ProbUF, in the four real-world networks.

Networks	Epinion	Slashdot	Wikipedia	Youtube
Space (MB)	30.17	18.80	1.63	2.01

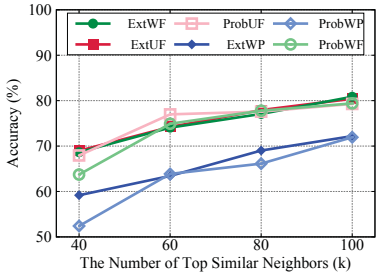
We further examine the space cost of the min-hash based probabilistic methods in real-world networks, as shown in Table I. This is particularly useful in investigating the effectiveness of the probabilistic approaches in space-constrained settings, such as graph streams. Note that the reported results are the *overall* memory consumption for min-hash structures in the probabilistic method, ProbUF, in the four real-world networks (we recorded very similar space cost for the other two probabilistic methods, ProbWF and ProbWP, and thus omit these results in the table). It can be witnessed that probabilistic methods only take very small memory footprints even in large networks, because the space cost for min-hash structures is $O(d|V|)$, which is only related to the number of vertices of networks. So the min-hash based probabilistic methods are also scalable in classifying edges for large networks. In comparison to the exact methods that require an explicit materialization of the whole networks before edge classification is performed, the probabilistic methods only need succinct, memory-resident min-hash structures. As a result, these probabilistic methods can even be applied in extremely large, disk-resident graphs or fast evolving graph streams where an explicit storage of the underlying graphs becomes impossible [22].

In the final set of experiments, we examine the performance of our edge classification methods by tuning important algorithmic parameters. First of all, we consider the parameter k, the number of top similar neighboring vertices selected during edge classification, and examine how k affects the edge classification accuracy, and the results are demonstrated in Figure 3 for all the four networks. It is clear to note that a larger value of k can often lead to higher accuracy for different edge classification methods. This is mainly because the edge label is determined by the dominant label of edges within the vicinity of vertex pairs, and a larger set of top similar vertices (regulated by k) can enhance the accuracy for this majority voting strategy. However, when k is increased to a large value, this accuracy improvement becomes insignificant. This is especially true for probabilistic methods.

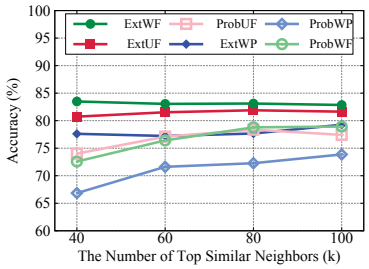
We then examine another important parameter, d, the number of min-wise hash functions in our probabilistic edge classification method, ProbUF (Experimental results and trends for ProbWF and ProbWP have been found similar to those of



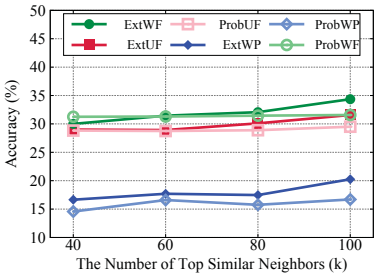
(a) Epinion



(b) Slashdot



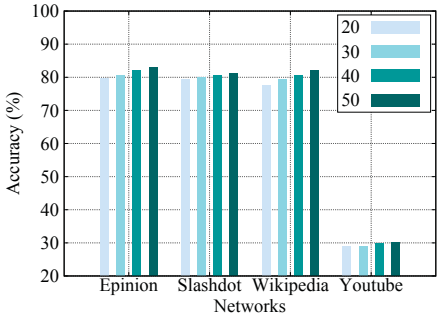
(c) Wikipedia



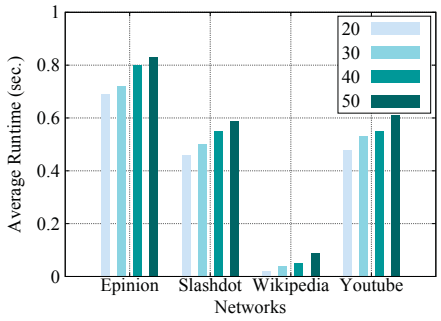
(d) Youtube

Fig. 3: Edge classification accuracy of six different methods in the four real-world networks when the number of top similar neighbors, k, varies.

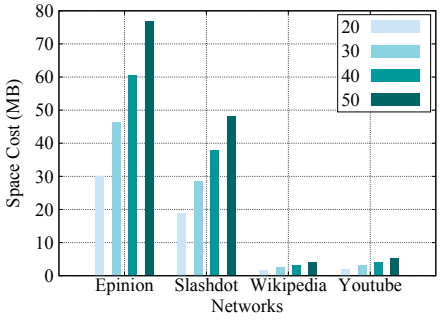
ProbUF, and thus are omitted for the interest of brevity), and the experimental results are shown in Figure 4. By varying the number of min-wise hash functions, d, in the min-hash structures from 20 up to 50, we recognize that the edge classification accuracy of ProbUF in the four different networks is improved slightly (Figure 4(a)), while both the time and space costs grow proportionally (Figure 4(b) and Figure 4(c)), because more min-wise hash functions and corresponding min-hash indexes are created and updated during the course of edge classification. This indicates that a moderate value of d, say 20, is typically good enough to support effective probabilistic edge classification in real-world networks. In the meantime, the resultant min-hash structures are both cost-effective and



(a) Accuracy



(b) Runtime

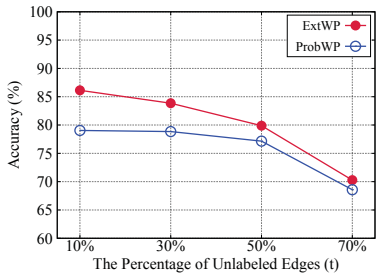


(c) Space

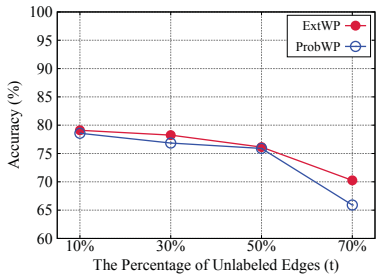
Fig. 4: Edge classification performance of the probabilistic method, ProbUF, in the four real-world networks when the number of min-wise hash functions, d, varies.

efficient to support Jaccard coefficient estimation during edge classification.

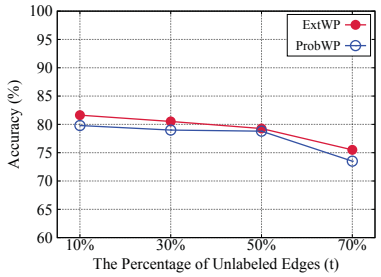
Our final experiment is to examine the edge classification accuracy of two algorithms, ExtWP and ProbWP, in partially labeled graphs in terms of the percentage t of unlabeled edges within networks. This type of experiments provides an idea of how the edge sparsity affects the classification effectiveness. The experimental results are illustrated in Figure 5. In all four real-world networks, it can be clearly seen that, when the percentage of unlabeled edges, t, grows from 10% up to 70%, the edge classification accuracy drops significantly. This is mainly because much fewer labeled edges exist within the neighborhood of vertex pairs for edge label prediction. However, the gap of edge classification accuracy between the exact method, ExtWP, and the probabilistic method, ProbWP, is still close, indicating that even in very sparsely labeled networks, the probabilistic edge classification methods are still good surrogates for their corresponding exact methods.



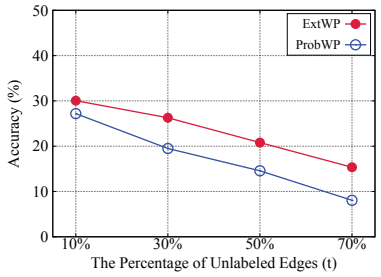
(a) Epinion



(b) Slashdot



(c) Wikipedia



(d) Youtube

Fig. 5: Edge classification accuracy of ExtWP and ProbWP on partially labeled networks when the percentage of unlabeled edges, t, varies.

V. Conclusions

In this paper, we designed an edge classification model in the network setting. The edge classification problem is particularly challenging because of the complexity associated with structural relationships between the incident vertices of an edge to be classified. We designed neighborhood-based methods to perform edge classification in large networks. The main challenge of these exact approaches stems from their high computational complexity. We further proposed probabilistic, min-hash based data structures to infer the relationships among the edges in an efficient and cost-effective way. Such probabilistic edge classification methods are scalable in networks, and can be applied successfully in disk-resident graphs or graph streams for edge classification. Our experimental results have demonstrated that our approaches are able to infer edge labels accurately without compromising on classification efficiency in large networks that can be either disk-resident or stream-like.

VI. Acknowledgement

Research of the first author was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

[1] C. C. Aggarwal and H. Wang, *Managing and Mining Graph Data*. Springer Publishing Company, Incorporated, 2010.

[2] D. J. Cook and L. B. Holder, *Mining Graph Data*. John Wiley & Sons, 2006.

[3] S. E. Schaeffer, "Survey: Graph clustering," *Comput. Sci. Rev.*, vol. 1, no. 1, pp. 27–64, 2007.

[4] P. Yang and P. Zhao, "A min-max optimization framework for online graph classification," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM'15)*, 2015, pp. 643–652.

[5] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: A survey," *Data Min. Knowl. Discov.*, vol. 29, no. 3, pp. 626–688, 2015.

[6] E. David and K. Jon, *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, 2010.

[7] M. Newman, *Networks: An Introduction*. Oxford University Press, Inc., 2010.

[8] P. Sen, G. M. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, no. 3, pp. 93–106, 2008.

[9] H. Fei and J. Huan, "Structured sparse boosting for graph classification," *ACM Trans. Knowl. Discov. Data*, vol. 9, no. 1, pp. 4:1–4:22, 2014.

[10] X. Kong and P. S. Yu, "gmlc: a multi-label feature selection framework for graph classification." *Knowl. Inf. Syst.*, vol. 31, no. 2, pp. 281–305, 2012.

[11] X. Kong, W. Fan, and P. S. Yu, "Dual active feature and sample selection for graph classification," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'11)*, 2011, pp. 654–662.

[12] Y. Anava, N. Avigdor-Elgrabli, and I. Gamzu, "Improved theoretical and practical guarantees for chromatic correlation clustering," in *Proceedings of the 24th International Conference on World Wide Web (WWW'15)*, 2015, pp. 55–65.

[13] F. Bonchi, A. Gionis, F. Gullo, and A. Ukkonen, "Chromatic correlation clustering," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'12)*, 2012, pp. 1321–1329.

[14] M. Rocklin and A. Pinar, "Latent clustering on graphs with multiple edge types," in *Proceedings of the 8th International Conference on Algorithms and Models for the Web Graph (WAW'11)*, 2011, pp. 38–49.

[15] P. Gupta, V. Satuluri, A. Grewal, S. Gurumurthy, V. Zhabiuk, Q. Li, and J. Lin, "Real-time twitter recommendation: Online motif detection in large dynamic graphs," *Proc. VLDB Endow.*, vol. 7, no. 13, pp. 1379–1380, 2014.

[16] O. Küçüktunç, E. Saule, K. Kaya, and U. V. Çatalyürek, "Diversified recommendation on graphs: Pitfalls, measures, and algorithms," in *Proceedings of the 22Nd International Conference on World Wide Web (WWW'13)*, 2013, pp. 715–726.

[17] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems (NIPS'14)*, 2014, pp. 2672–2680.

[18] W. Yu, C. C. Aggarwal, S. Ma, and H. Wang, "On anomalous hotspot discovery in graph streams," in *2013 IEEE 13th International Conference on Data Mining (ICDM'13)*, 2013, pp. 1271–1276.

[19] D. E. Difallah, G. Demartini, and P. Cudré-Mauroux, "Pick-a-crowd: Tell me what you like, and i'll tell you what to do," in *Proceedings of the 22Nd International Conference on World Wide Web (WWW'13)*, 2013, pp. 367–374.

[20] S.-H. Yang, B. Long, A. Smola, N. Sadagopan, Z. Zheng, and H. Zha, "Like like alike: Joint friendship and interest propagation in social networks," in *Proceedings of the 20th International Conference on World Wide Web (WWW'11)*, 2011, pp. 537–546.

[21] F. Cacheda, V. Carneiro, D. Fernández, and V. Formoso, "Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems," *ACM Trans. Web*, vol. 5, no. 1, pp. 2:1–2:33, 2011.

[22] A. McGregor, "Graph stream algorithms: A survey," *SIGMOD Rec.*, vol. 43, no. 1, pp. 9–20, 2014.

[23] C. Aggarwal and K. Subbian, "Evolutionary network analysis: A survey," *ACM Comput. Surv.*, vol. 47, no. 1, pp. 1–36, 2014.

[24] C. C. Aggarwal, *Social Network Data Analytics*. Springer Publishing Company, Incorporated, 2011.

[25] D. Zhou, J. Huang, and B. Schölkopf, "Learning from labeled and unlabeled data on a directed graph," in *Proceedings of the 22Nd International Conference on Machine Learning (ICML'05)*, 2005, pp. 1036–1043.

[26] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Advances in Neural Information Processing Systems (NIPS'04)*, 2004, pp. 321–328.

[27] S. Bhagat, I. Rozenbaum, and G. Cormode, "Applying link-based classification to label blogs," in *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 Workshop on Web Mining and Social Network Analysis (WebKDD/SNA-KDD '07)*, 2007, pp. 92–101.

[28] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proceedings of the 22Nd International Conference on Machine Learning (ICML'03)*, 2003, pp. 912–919.

[29] Q. Lu and L. Getoor, "Link-based classification," in *Proceedings of the 22Nd International Conference on Machine Learning (ICML'03)*, 2003, pp. 496–503.

[30] S. Bhagat, I. Rozenbaum, and G. Cormode, "Applying link-based classification to label blogs," in *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 Workshop on Web Mining and Social Network Analysis (WebKDD/SNA-KDD '07)*, 2007, pp. 92–101.

[31] M. Bilgic and L. Getoor, "Effective label acquisition for collective classification," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'08)*, 2008, pp. 43–51.

[32] V. R. Carvalho and W. W. Cohen, "On the collective classification of email "speech acts"," in *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'05)*, 2005, pp. 345–352.

[33] T. Zhang, A. Popescul, and B. Dom, "Linear prediction models with graph regularization for web-page categorization," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'06)*, 2006, pp. 821–826.

[34] S. Chakrabarti, B. Dom, and P. Indyk, "Enhanced hypertext categorization using hyperlinks," in *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data (SIGMOD'98)*, 1998, pp. 307–318.

[35] P. Agrawal, V. K. Garg, and R. Narayanam, "Link label prediction in signed social networks," in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI'13)*, 2013, pp. 2591–2597.

[36] S.-H. Yang, A. J. Smola, B. Long, H. Zha, and Y. Chang, "Friend or frenemy?: Predicting signed ties in social networks," in *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'12)*, 2012, pp. 555–564.

[37] G. Bachi, M. Coscia, A. Monreale, and F. Giannotti, "Classifying trust/distrust relationships in online social networks," in *2012 International Conference on Privacy, Security, Risk and Trust (PASSAT'12)*, 2012, pp. 552–557.

[38] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Predicting positive and negative links in online social networks," in *Proceedings of the 19th International Conference on World Wide Web (WWW'10)*, 2010, pp. 641–650.

[39] C. Wang, J. Han, Y. Jia, J. Tang, D. Zhang, Y. Yu, and J. Guo, "Mining advisor-advisee relationships from research publication networks," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'10)*, 2010, pp. 203–212.

[40] C. Diehl, G. M. Namata, and L. Getoor, "Relationship identification for social network discovery," in *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI '07)*, 2007, pp. 546–552.

[41] W. Tang, H. Zhuang, and J. Tang, "Learning to infer social ties in large networks," in *Proceedings of the 2011 European Conference on Machine Learning and Knowledge Discovery in Databases (PKDD'11)*, 2011, pp. 381–397.

[42] C. C. Aggarwal, *Data Classification: Algorithms and Applications*. Chapman & Hall/CRC, 2014.

[43] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, 2009.

[44] S. Pandey, A. Broder, F. Chierichetti, V. Josifovski, R. Kumar, and S. Vassilvitskii, "Nearest-neighbor caching for content-match applications," in *Proceedings of the 18th International Conference on World Wide Web (WWW'09)*, 2009, pp. 441–450.

[45] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher, "Min-wise independent permutations," in *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing (STOP'98)*, 1998, pp. 327–336.

[46] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing (STOC'02)*, 2002, pp. 380–388.

[47] S. Guha and A. McGregor, "Graph synopses, sketches, and streams: A survey," *Proc. VLDB Endow.*, vol. 5, no. 12, pp. 2030–2031, 2012.

[48] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge University Press, 1995.