Charu C. Aggarwal, Peixiang Zhao, and Gewen He
Florida State University
IBM T J Watson Research Center

Edge Classification in Networks

ICDE Conference, 2016

Introduction

- 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.

Applications

- The edge classification problem has numerous applications in a wide array of network-centric scenarios:
  - A social network may contain relationships of various types such as friends, families, or even rivals. A subset of the edges may be labeled with the relevant relationships. The determination of unlabeled relationships can be useful for making targeted recommendations.
  - An adversarial or terrorist network may contain unknown relationships between its participants although a subset of the relationships have been known already.
  - In some cases, the edges may have numerical quantities associated with them corresponding to strengths of relationships, specification of ratings, likes, or dislikes.

Contributions

- We formulate the edge classification problem and the edge regression problem in graph-structured networks.

- We propose a structural similarity model for edge classification.

- 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;

- 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.

Problem Definition

- We 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 $A_u = A \setminus A_l$.

Formal Definitions

- 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 label l_{uv}, 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,

- 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 $A_u = A \setminus A_l$.

Overview

- 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.

Overview of Steps

- Consider an edge $(u, v) \in A_u$, which needs to be classified within the network $G = (N, A)$.
  - The top-k most similar vertices to u, denoted by $S(u) = \{u_1 \ldots u_k\}$, are first determined based on a structural similarity measure discussed slightly later.
  - The top-k most similar vertices to v, denoted by $S(v) = \{v_1 \ldots v_k\}$, are determined based on the same structural similarity measure as in the case of vertex u;
  - 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).

Similarity Computation

- How are 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.

- We consider Jaccard coefficient for vertex-wise similarity quantification.

Similarity Computation

- Let $I^-(u) \subseteq N$ be the set of vertices incident on the vertex u belonging to the edge label -1.

- 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:

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

Similarity Computation (Contd.)

- 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 $J^{-}(u,v)$.

- The most effective way to achieve this goal is to use the fraction of vertices belonging to the two classes.

Weighted Jaccard

- 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 $f^-(u,v) = (f^-(u) + f^-(v))/2$.

- Therefore, the overall Jaccard similarity, $J(u,v)$, of the vertices u and v is defined by the weighted average:

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

Differential Weight

- 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.

Sparse Labeling

- 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.

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

- Here, μ is a discount factor less than 1, which is used to prevent excessive impact of the unlabeled edges on the similarity computation.

Numerical Labels

- 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.

- 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, $I^-(u)$.

- 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.

Classification with Numeric Labels

- 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.

Challenges

- 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.

Basic Approach

- Use synopsis structure to summarize the graph
- Synopsis structure can be efficiently updated in real time
- Synopsis structure can be leveraged to efficiently perform classification

Broad Approach

- In order to achieve this goal, we will consider a probabilistic, min-hash based approach, 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.

Min-Hash Technique

- The use of the min-hash index is standard in these settings
- Details of the use of the min-hash index in paper
- Theoretical bounds provided in paper

Experimental Results

- Experimental studies demonstrate the effectiveness and efficiency of our proposed edge classification methods in real-world networks.

- We consider a variety of effectiveness and efficiency metrics on diverse networked data sets to show the broad applicability of our methods.

Data Sets

- Epinions (binary)
- Slashdot (binary)
- Wikipedia (binary)
- Youtube (multiple labels)

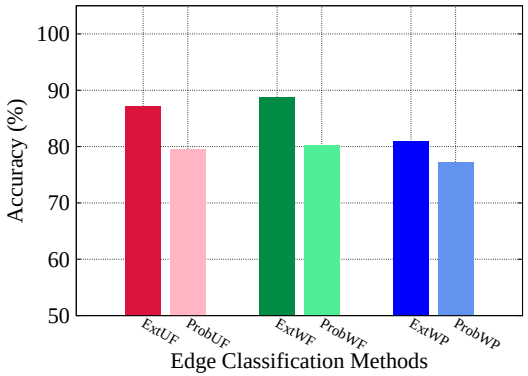
Metrics

- **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;

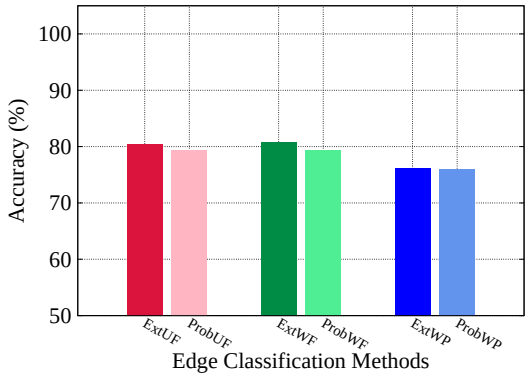
- **Time:** we gauge the average time consumed for edge classification for the $1,500$ edges selected randomly from within the networks;

- **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.

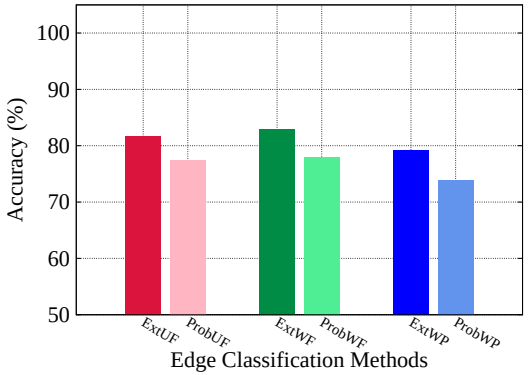
Accuracy



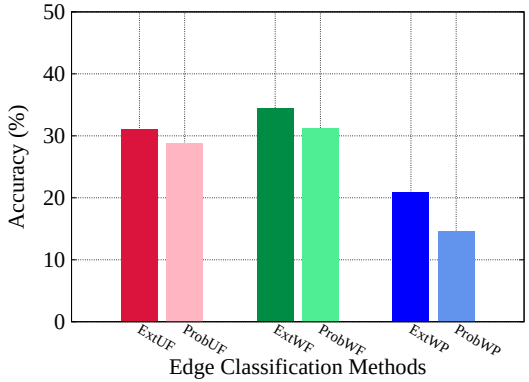
(a) Epinion



(b) Slashdot

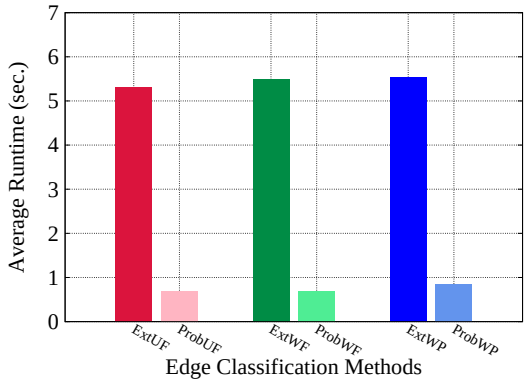


(c) Wikipedia

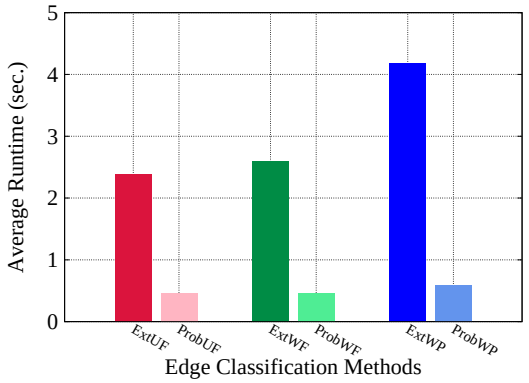


(d) Youtube

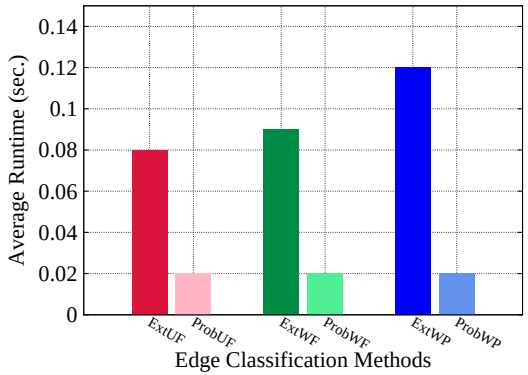
Running Time



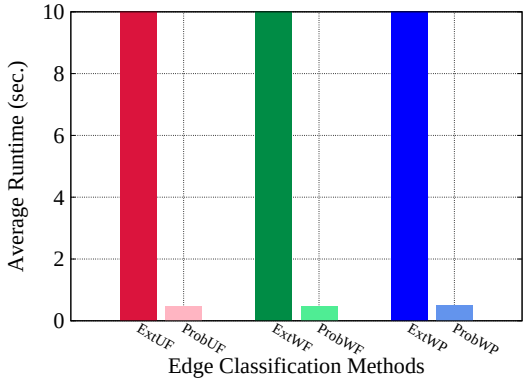
(a) Epinion



(b) Slashdot

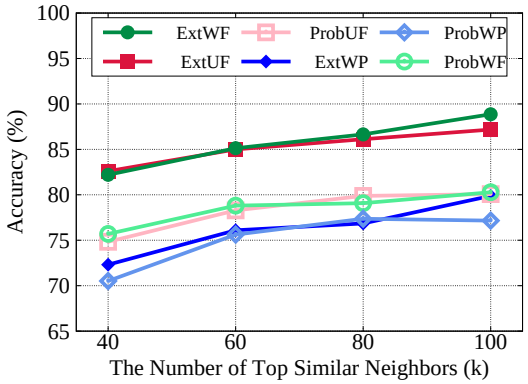


(c) Wikipedia

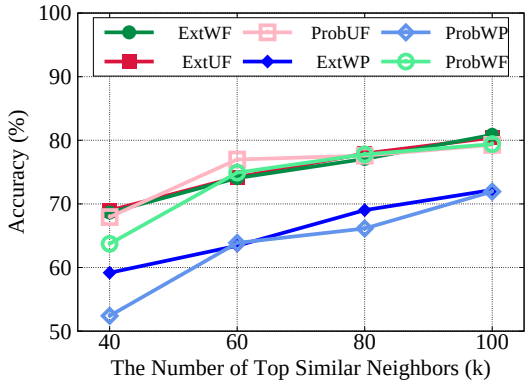


(d) Youtube

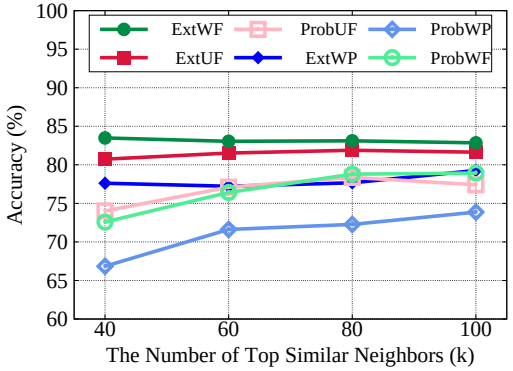
Effect of the number of nearest neighbors



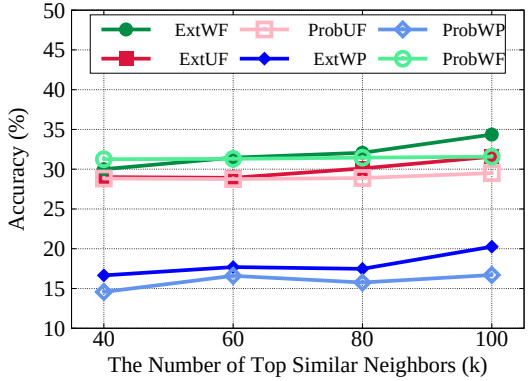
(a) Epinion



(b) Slashdot

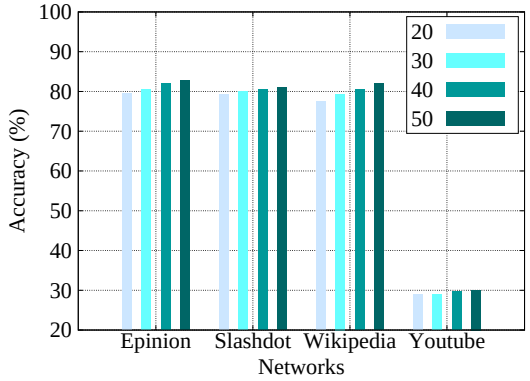


(c) Wikipedia

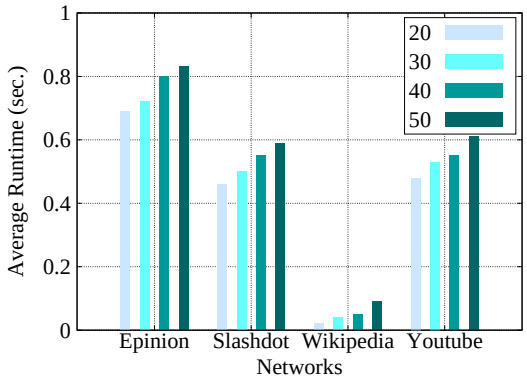


(d) Youtube

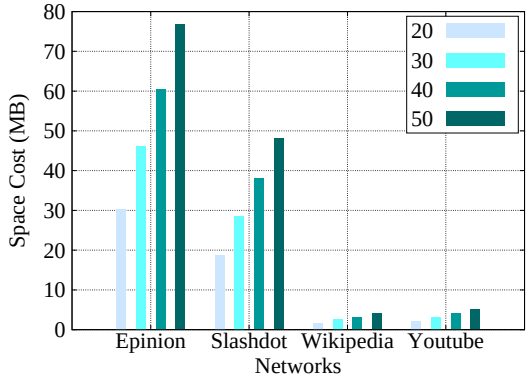
Effect of hash functions



(a) Accuracy

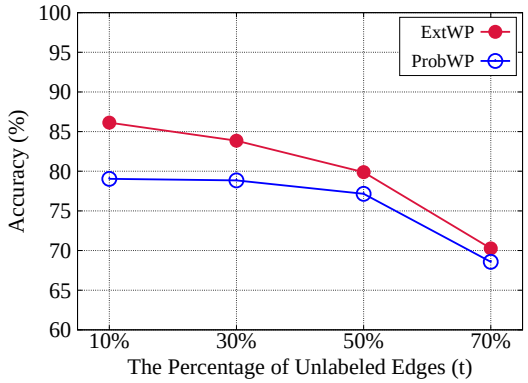


(b) Runtime

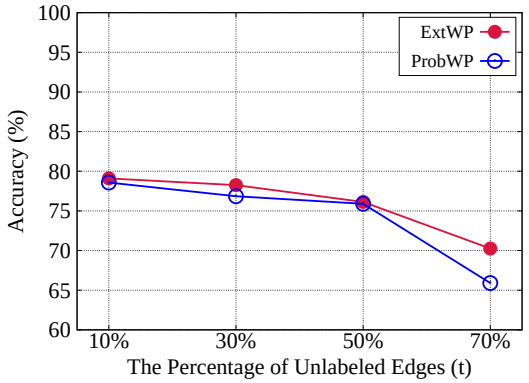


(c) Space

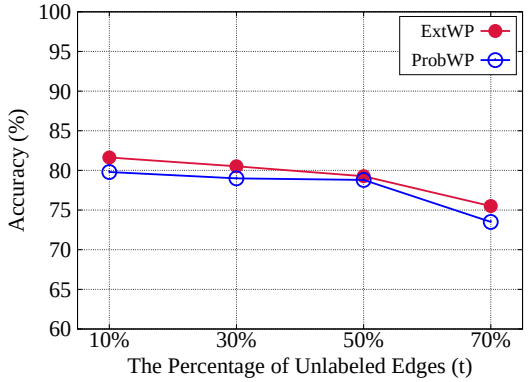
Effect of the number of unlabeled edges



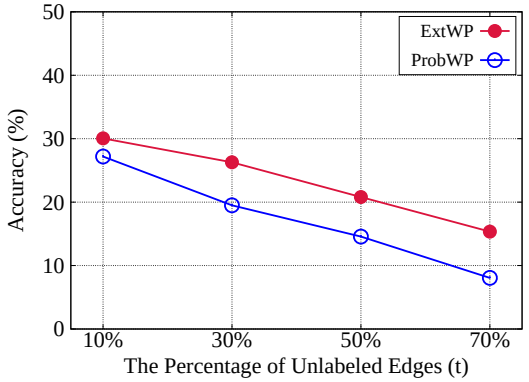
(a) Epinion



(b) Slashdot



(c) Wikipedia



(d) Youtube

Conclusions

- New technique for edge classification in networks
- Uses probabilistic min-hash data structures for efficient classification