Detection of Nearby Friends

The key insight is to leverage tweets from **"co-location friends"** for improving accuracy. Co-location friends are OSN friends that also live near the target user in real world. However, not all friends in Twitter are co-location friends so they should be classified and treated discriminatingly.

1 Empirical Study

Intuitively, co-location friends of the target user are typically neighbors, classmates, colleagues, etc. These people tend to know each other due to physical proximity. As a result, they probably follow each other on line and form community in Twitter, which implies that the topology of social network may reveal useful hints on identifying co-location friends.

Figure 1 shows a real world example from Twitter. All nodes of red circle are users lived in Oklahoma City, OK while the nodes of blue box are users in other cities or users whose locations are unknown.

As can be seen from this figure, most co-location friends form a tightly connected component in the graph. In other words, if two nodes are co-location friends, they have a higher probability to establish a link between each other.

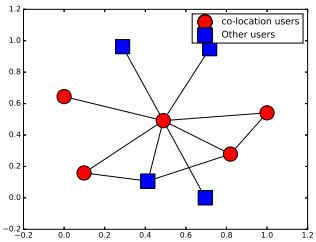

Figure 1: A Real World Example of Co-location User.

Let P_{local} denote the probability to establish a link between two co-location nodes and $P_{nonlocal}$ denote the probability to establish a link between two nodes, at least one of which is not co-location node. Statistical results of P_{local} and $P_{nonlocal}$ from real world data are shown in Figure 2. As one can see from the figure, P_{local} are significantly larger than $P_{nonlocal}$.

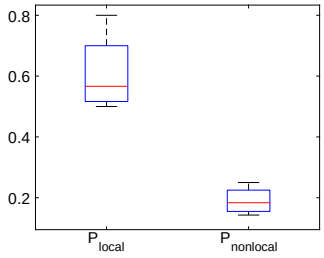

Figure 2: Statistical Results of P_{local} and $P_{nonlocal}$.

2 Co-location Analysis

Based on the above observation, it's possible to infer the labels of nodes (co-location or not) according to the topology. In other words, we aim at assigning each node with a proper label(local or nonlocal), which maximize the total probability for forming the observed topology structure.

Let $G(V, E)$ denote the graph of one-hop social network of the target user and $U \subset V$ is the set of co-location users.

We first derive the probability to produce an existing topology given U. For simplicity, different links are assumed to be formed independently.

Let's consider the existing links among two local nodes. Since establishing a link between two local nodes are P_{local}, the probability to form all this kind of links are:

$$\prod_{\substack{(i,j)\in E \\ i\in U \wedge j\in U}} P_{local} \tag{1}$$

In addition, the missing links between two co-location nodes also provides opposing evidences for current assignment of labels and therefore, should be taken into consideration. The probability to observe all this kind of missing links is

$$\prod_{\substack{(i,j)\notin E \\ i\in U \wedge j\in U}} 1 - P_{local} \tag{2}$$

Similarly, consider the links between two nodes, at least one of which is not a co-location node. Then the probabilities for these kinds of existing and missing links are

$$\prod_{\substack{(i,j)\in E \\ i\notin U \vee j\notin U}} P_{nonlocal} \tag{3}$$

and

$$\prod_{\substack{(i,j)\notin E \\ i\notin U \vee j\notin U}} 1 - P_{nonlocal} \tag{4}$$

respectively.

The probability for producing the whole network can be gained by calculating the product of all four probabilities above, which is shown in Equation 5.

$$P(U) = \prod_{\substack{(i,j)\in E \\ i\in U \wedge j\in U}} P_{local} \prod_{\substack{(i,j)\notin E \\ i\in U \wedge j\in U}} 1 - P_{local} \prod_{\substack{(i,j)\in E \\ i\notin U \vee j\notin U}} P_{nonlocal} \prod_{\substack{(i,j)\notin E \\ i\notin U \vee j\notin U}} 1 - P_{nonlocal} \tag{5}$$

Our goal is to maximize $P(U)$ by choosing a proper U, which is shown in Equation 6.

$$\underset{U}{\operatorname{argmax}} P(U) \tag{6}$$

In practice, we adopt the log form of probability $P(U)$ to avoid underflow in computation.

Note that to solve this optimization problem is computationally expensive because one have to enumerate all possible combination of node labels. However, the number of combination grows exponentially as the number of nodes increases. For example, there will be 2^n possible label settings if the number of friends is n.

To solve the problem efficiently, we propose the following Local Probability Maximization Algorithm(LPMA) (shown in algorithm 1). At the beginning, each node is assigned with a random label. Then, for each node in the graph(except the target user, which is regarded as a local node by definition), we check whether changing its label will increase the global probability if labels of other nodes remain unchanged. If so, its label will be flipped. This process iterates until convergence(i.e., no label changes in one iteration).

If the size of the social network is n, the computation complicity for the brute force approach is $O(2^n)$ since it has to check all possible 2^{n-1} combinations of labels. And for the LPMA algorithm, its computational complicity is $O(tn)$, where n the number of nodes and t is the number of iteration.

A case study of our algorithm is shown in Figure 3, where we use the social network mentioned in the empirical study as input. Nodes of magenta circle are co-location users computed by our algorithm while green boxed ones are other users. As can be seen from this figure, the co-location users detected by our approach match well with the real users who live in the same city with the target user. In fact, our approach is able to detect most of the co-location users with only 1 misclassification.

All tweets from the target user and detected co-location friends will be fed into the content analyzer. The result will be regraded as refined statistical location.

Algorithm 1: Local Probability Maximization Algorithm(LPMA) for co-location user detection

Input : $G(V, E)$, P_{local}, $P_{nonlocal}$, Target User t
Output: U

Randomly initialize U;
$U = U \cup t$;
while *true* **do**
    isConverge = true;
    bestProb = P(U) ;
    for $n \in V/\{t\}$ **do**
        if $n \in U$ **then**
            newU = $U/\{n\}$;
            currentProb = P(newU);
        else
            newU = $U \cup \{n\}$;
            currentProb = P(newU);
        end
        if *currentProb* >*bestProb* **then**
            U = newU ;
            bestProb = currentProb;
            isConverge = false;
        end
    end
    if *isConverge* **then**
        break;
    end
end
return U

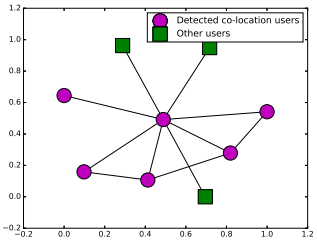

Figure 3: A Case Study of Detected Co-location Users.