Event Prediction with Community Leaders

Jun Pang*†, Yang Zhang†

*Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg
†Faculty of Science, Technology and Communication, University of Luxembourg

***Abstract*—With the emerging of online social network services, quantitative studies on social influence become achievable. Leadership is one of the most intuitive and common forms for social influence; understanding it could result in appealing applications such as targeted advertising and viral marketing. In this work, we focus on investigating leaders' influence for event prediction in social networks. We propose an algorithm based on events that users conduct to discover leaders in social communities. Analysis on the leaders that we found on a real-life social network dataset leads us to several interesting observations, such as that leaders do not have significantly higher number of friends but are more active than other community members. We demonstrate the effectiveness of leaders' influence on users' behaviors by learning tasks: given a leader has conducted one event, *whether* and *when* a user will perform the event. Experimental results show that with only a few leaders in a community the event predictions are always very effective.**

I. Introduction

Social influence is a phenomenon that people change their behaviors because of their social relations. It has been widely accepted and studied in the area of sociology [1], [2], [3], [4]. Due to the lack of data, most previous research on social influence are constrained at qualitative level. During the past decade, online social network services (OSNs) have gained a huge success. Nowadays, OSNs become an important part of people's daily life. Leading actors in this business own large quantities of users and data. For example, Facebook has more than one billion active users and Twitter users publish 500 million tweets every day. In OSNs, people specify their profiles, articulate social relationships and share activities. With these huge amount of data, quantitative studies of social influence become achievable.

There are several forms of social influence such as conformity, obedience, peer pressure, leadership, etc. Among them, leadership is one of the most common and intuitive forms for social influence, and it has been defined as a social influence process of organizing a group of people to achieve a common goal [4]. There are two constrains in the leadership definition. First, according to [3], we recognize that "leadership operates in groups". Thus it is important to study leaders in the context of social groups or social communities. In a broader context, the authors of [5], [6] argue that the community is the most useful level of resolution for studying social networks. Second, we notice that leader may not be unique, following Merton's monomorphic leadership theory [1]. This indicates that a community often may have several leaders. We take these two observations in our following investigation.

Understanding leader's behaviors can result in appealing applications, including viral marketing, targeted advertising and recommendations. For example, a shop can promote its new products by providing the leaders of a community some free samples or coupons, and counts on these leaders to attract more customers. Moreover, by only focusing on leaders instead of all customers, the marketing cost will decrease as well. On the other hand, exploiting leaders to predict other users' future behavior can also cause privacy threat. Not many research have focus on quantitatively modeling and understanding the phenomenon of leadership (see more discussions in Section VII). In this paper, we aim to mine leaders from social communities and demonstrate their effectiveness in some real-life scenarios, for instance, *event prediction*.

Our contributions in this work are summarized as follows.

- We propose an algorithm for discovering leaders (Section III). The algorithm is based on the solutions of [7] and [8], which aim to quantify the influence of users by checking the events they conducted in the past. Different from their solutions, we take into account important features of the events. More specifically, we focus on the popularity of the event, and use it to adjust each user's influence score.
- By analyzing behaviors of the leaders in a real-life social network dataset, we have found out that our leaders don't have significantly higher number of friends than other community members, but they are more active in terms of the number of conducted events (Section IV).
- We further define an event prediction task which aims to predict whether and when a user will perform one event in the future after a leader (Section V, VI). Our predictor has a strong performance and outperforms the one that is based on the leaders discovered by a state-of-the-art algorithm, namely the LeaderRank algorithm [9].

II. Preliminaries

Notations. A user is denoted by u and set $f(u)$ represents u's friends. A social network is formalized as a social graph G. According to [3], "Leadership operates in groups", which means that leadership is about influencing a community of people who are engaged in a common goal or purpose, thus in this paper, we focus on leaders at the community level. A community is a set of users, represented by C. We use $f(u, C)$ to denote u's friends in community C. Each community forms a graph denoted by $G_C = (C, E_C)$ where E_C represents all the edges among users in C. We use e to represent an event. Each user can conduct several events. The events that a community C performs are denoted by $E(C)$. Similarly, $E(u)$ is the set of all events that u conducts. The fact that u performing e is stored as an event log defined as $\langle u, e, t\rangle$ where t is the time when this event is conducted. We use $\mathit{Elog}(C, e)$ to denote the event logs of C regarding the event e. Similarly, $\mathit{Elog}(u, e)$ is the event log of u on e. Moreover, $\mathit{Elog}(u, e).t$ represents the time of the event log. If u conducts e before another user $u' \in f(u)$, we assume that u influences u'. This influence formalization is similar to [8] in which the authors argue that it is more inclusive than other ways such as retweeting in Twitter. A user may conduct an event due to the social influence through his friends. Meanwhile, if he keeps on performing the same event after, then this is more related to the event itself rather than the social influence. Therefore, we make a simplification that a user performs each event at most once. In [10], the authors have made the same assumption.

Following Merton's monomorphic leadership theory [1], a person who is a leader in one field may be a follower in another field. Thus we regulate that for every specific event a community has one leader, while the community in general could have multiple leaders. We use $\ell_{C,e}$ to denote the leader of C on e and $L(C)$ to denote C's leader set.

Community Detection. In this work, we use the Infomap algorithm to discovery communities since it is one of the most widely used algorithm [11]. Note that our work is not restricted by the chosen community detection algorithm.

III. Leader Discovery

As described in Section II, we are dealing with leaders at the community level. Each community can have several leaders while there is only one leader on one event. To discover the leaders of a community, we first find the leader for each event that the community conducts and then find the leader set of the community based on the discovered leaders on events. Our leader discovery algorithm is thus partitioned into two steps: (1) for each C, we find $\ell_{C,e}$ for all $e \in E(C)$; (2) we find the leader set $L(C)$ from all the discovered $\ell_{C,e}$.

The procedure to discover $\ell_{C,e}$ is described in Algorithm 1, it is inspired by the works in [7] and [8]. We aim to build an *influence tree* for C for a given e. The event log $\mathit{Elog}(C, e)$ is sorted chronologically (line 1). Then $\ell_{C,e}$ is the first user who has performed e in C (line 2), this is simply due to the fact that he is the first one who introduces the event e to his community C. The work of [8] has adopted the same formalization. Moreover, $\ell_{C,e}$ is taken as the root of the tree (line 3). We then iterate the items in $\mathit{Elog}(C, e)$. For a user, if any of his friends is already in the influence tree, then we add him into the tree as well (lines 4-8). This suggests that he is influenced by his friend to perform this event as we mentioned in Section II. Here, we exploit the influence assumption that a user gets influence on a certain event from his friend who conducts that event most recently, i.e., the "last influence" assumption proposed in [10], [8] (line 6). In the end, the influence score of the leader is the number of nodes in the influence tree divided by the total number of users in the community (line 10). Note that the leader himself is not counted as a member who he has influenced. Therefore, we exclude him when calculating the influence score (line 10).

Algorithm 1 Discovering $\ell_{C,e}$

```
Input: $C$ and $\mathit{Elog}(C, e)$
Output: $\ell_{C,e}$ and $\mathit{infscore}(\ell_{C,e}, e)$
 1: sort $E(C, e)$ in a chronological order
 2: $\ell_{C,e} \leftarrow$ the user of the first event log
 3: add $\ell_{C,e}$ as the root of inftree
 4: for each $\langle u, e, t\rangle \in \mathit{Elog}(C, e)$ do
 5:     if $\exists u' \in f(u, C)$ s.t. $u'$ is already in inftree then
 6:         add $u$ as a descendant of his friend
            who is added to the inftree most recently
 7:     end if
 8: end for
 9: $\mathit{tree\_size} \leftarrow$ the number of nodes in inftree
10: $\mathit{infscore}(\ell_{C,e}, e) \leftarrow (\mathit{tree\_size} - 1)/(|C| - 1)$
11: return $\ell_{C,e}$ and $\mathit{infscore}(\ell_{C,e}, e)$
```

After getting $\ell_{C,e}$ for all $e \in E(C)$, we perform Algorithm 2 to discover $L(C)$. The basic idea is to sum up each user's influence scores on all the events that he is the leader of and choose the ones with high scores as leaders. However, we make a subtle modification on the influence score of each event here. We notice that some events performed by a user are less socially driven, such as taking public transportation. For a leader of this kind of events, his influence score should be weighted less. On the other hand, one viewing some artist's work is probably through the recommendation of the leaders in his community. Influence scores of this kind should value more. Therefore, we first adjust each event's influence score by an event factor (lines 2-4), then sum up all the influence scores of each user who is the leader for at least one event (lines 5-7). After getting the highest influence score (line 8), we add all users whose influence scores are close to the highest score into $L(C)$ (line 10). Here, ϵ is between 0 and 1 and it can be set according to an application scenario.

IV. Leader Analysis

In this section, we first detect leaders on a social network dataset with event logs. Then, we focus on the characteristics of the discovered leaders.

A. Discovering Leaders

Dataset. Location-based social networks (LBSN) have gain a huge popularity. Like other social network services, a user can establish social relationships. Moreover, when a user visits a place, he can directly share that location, called *check-in*. In this work, we treat users' check-ins at different locations as events that they conduct. Therefore, in LBSN dataset we have both the social graph and event logs for users.

We conduct our experiments using the dataset [12] collected from Gowalla, a popular LBSN back in 2011. The social graph of the dataset has 196,591 users with 950,327 edges. It also contains more than 6 million check-ins. Due to the sparsity

Algorithm 2 Discovering $L(C)$

```
Input: $\ell_{C,e}$ and $infscore(\ell_{C,e}, e)$ for all $e \in E(C)$
Output: $L(C)$
1: $L(C) \leftarrow \emptyset$
2: for $e \in E(C)$ do
3:     $infscore(\ell_{C,e}, e) \leftarrow infscore(\ell_{C,e}, e) \times evefac(e)$
4: end for
5: for $u \in \{\ell_{C,e} \mid e \in E(C)\}$ do
6:     $infscore(u) = \sum infscore(u, e)$
7: end for
8: $maxscore \leftarrow \max\{infscore(u) | u \in \{\ell_{C,e} | e \in E(C)\}\}$
9: $L(C) \leftarrow \{u \mid infscore(u) \geq \epsilon \times maxscore\}$
10: return $L(C)$
```

of location data, in this paper we only focus on the data from 11 major metropolises around the world including New York, L.A., Bay Area, Dallas, San Antonio, Houston, Chicago, Seattle, Tokyo, London and Stockholm. We further filter out users who have less than 10 check-ins and 5 friends. As we have the assumption that each user performs an event once, we only keep the earliest check-in of each user at each of his locations. The above constraints totally leave us 10,355 users with 528,266 check-ins at 164,851 locations. After performing the Infomap algorithm, 877 communities are detected, thus the average community size is around 12.

Leader discovery. In our leader discovery algorithm, we propose to take the event properties to adjust each leader's influence scores. Recall the example in Section III, events like going to a supermarket is certainly less socially driven, compared to eating at a restaurant. One of the major differences between the above two kinds of places is their popularity. The former one (supermarket) normally attracts lots of people while the latter one (restaurant) is relatively more private. To quantify the popularity of a location, we adopt the notion of *location entropy* as proposed in [13]. When a place is popular, its location entropy is high. Moreover, instead of directly using location entropy, we exploit its inverse, i.e., location diversity [14] defined as $locdiv(loc) = \exp(-locent(loc))$. The other parameter ϵ for deciding how many leaders should be included in a community is set to 0.9. In the end, 1,571 leaders are found for 877 communities.

B. Who are the Leaders?

After discovering the leaders, we are interested in whether these leaders are different from other members in communities. For every leader, we focus on three metrics including the number of his friends, the number of his friends in the community and the number of his check-ins. The results are listed in Table I. We can see that the leaders we found have much more check-ins than members in the communities, meaning that they are more active. On the other hand, leaders and members in a community have a similar amount of friends. This indicates that the leaders are not necessarily those users who know more people than others in their communities.

	Our algorithm		LeaderRank	
Average value	leader	mem.	leader	mem.
# of check-ins	153.30	72.79	117.94	76.61
# of friends	6.33	5.87	10.67	5.19
# of friends in com	3.96	3.88	6.40	3.50
closeness	0.10	0.05	0.12	0.04
betweenness	0.28	0.12	0.58	0.07

TABLE I: Feature comparison between leaders and community members w.r.t. two leader discovery algorithms

Each community C forms a graph G_C, we further calculate two centrality metrics, i.e., closeness and betweenness, for nodes in each community. As shown in Table I, both closeness and betweenness of leaders are much larger than those of community members. In [15], the authors discovered that users with high centrality scores are more influential, the centrality metrics give us an indirect evidence of it.

We are also interested in whether the leaders we found are different from the leaders discovered by algorithms based on graph structure. We exploit a state-of-the-art leader discovery algorithm LeaderRank [9] to find leaders as well. As shown in Table I, LeaderRank leaders have twice as many friends as members in communities, while using our algorithm these two values are similar between leaders and members. Moreover, the differences of centrality metrics are even larger. Especially, betweenness of leaders by LeaderRank is 8 times larger than members, while it is only around 2 times with our algorithm. Leaders from both algorithms have more check-ins than community members. However, the difference in our case is much higher (153.80-72.79 v.s. 117.94-76.71), i.e., our leaders are more active than leaders discovered by LeaderRank.

V. Event Prediction

So far, we have seen some characteristics of the leaders we found, next we want to demonstrate the leaders' influence on community members. In this section, we study the problem of predicting a user's behavior, i.e., check-in at a location, using his leader's information.

A. Problem Definition

Event prediction problem. For a community C with its leader set $L(C)$ and an event $e \in E(C)$, given a member u's closest leader $\ell \in L(C)$ that has conducted e, we predict whether u will perform e in the future.

The event prediction problem can be naturally formalized as a binary classification. Next, we describe the features we consider and utilize machine learning techniques to solve it.

B. Feature Description

The features we use are related to four aspects including the leader, the user, the relationship between them as well as the community itself. Therefore, we group our features into four categories. The detailed description is listed in Table II.

Leader features. The first feature related to the leader is his influence on community members. For example, if ℓ has

		Leader Features
1	$\mathit{infscore}(\ell)$	influence score of ℓ, i.e., $\sum\{\mathit{infscore}(\ell_{C,e}) \mid \ell_{C,e} = \ell\}$
2	$\mathit{infratio}(\ell)$	influence ratio of ℓ, i.e., $\mathit{infscore}(\ell)/\sum\{\mathit{infscore}(\ell_{C,e}) \mid e \in E(C)\}$
3	$\mathit{ci\_cnt}(\ell)$	the number of check-ins conducted by ℓ, i.e., $\lvert E(\ell)\rvert$
4	$\mathit{size}(f(\ell))$	the number of friends ℓ has in general, i.e., $\lvert f(\ell)\rvert$
5	$\mathit{size}(f(\ell, C))$	the number of friends ℓ has in C, i.e., $\lvert f(\ell, C)\rvert$
6	$\mathit{nb\_ratio}(\ell)$	the ratio of the number of ℓ's friends in C over the number of his friends in general, i.e., $\lvert f(\ell, C)\rvert/\lvert f(\ell)\rvert$
7	$\mathit{avg\_nb\_deg}(\ell)$	the average number of friends of ℓ's friends, i.e., mean$\{\mathit{size}(f(u)) \mid u \in f(\ell)\}$
8	$\mathit{closeness}(\ell)$	closeness of ℓ in the community graph G_C
9	$\mathit{betweenness}(\ell)$	betweenness of ℓ in G_C
10	$\mathit{eccentricity}(\ell)$	eccentricity of ℓ in G_C, i.e., max$\{\mathit{dist}(\ell, u) \mid u \in C\}$
		User Features
11	$\mathit{ci\_cnt}(u)$	the number of check-ins conducted by u, i.e., $\lvert E(u)\rvert$
12	$\mathit{size}(f(u))$	the number of friends u has in general, i.e., $\lvert f(u)\rvert$
13	$\mathit{size}(f(u, C))$	the number of friends u has in C, i.e., $\lvert f(u, C)\rvert$
14	$\mathit{nb\_ratio}(u)$	the ratio of the number of u's friends in C over the number of his friends in general, i.e., $\lvert f(u, C)\rvert/\lvert f(u)\rvert$
15	$\mathit{avg\_nb\_deg}(u)$	the average number of friends of u's friends, i.e., mean$\{\mathit{size}(f(u')) \mid u' \in f(u)\}$
16	$\mathit{closeness}(u)$	closeness of u in G_C
17	$\mathit{betweenness}(u)$	betweenness of u in G_C
18	$\mathit{eccentricity}(u)$	eccentricity of u in G_C, i.e., max$\{\mathit{dist}(u, u') \mid u' \in C\}$
		Leader-user Features
19	$\mathit{isfri}(u, \ell)$	whether u and ℓ are friends or not, $\mathit{isfri}(u, \ell) = 1$ when they are friends
20	$\mathit{dep2cnt}(u, \ell)$	the number of 2-depth paths between u and ℓ, i.e., $\lvert\{u' \mid u' \in f(u) \cap u' \in f(\ell)\}\rvert$
21	$\mathit{dist}(u, \ell)$	the length of the shortest path between u and ℓ
		Community Features
22	$\mathit{ci\_cnt}(C)$	the number of check-ins conducted by all users in C
23	$\mathit{size}(C)$	the size of the community C, i.e., $\lvert C\rvert$
24	$\mathit{density}(G_C)$	density of $G_C = (C, E_C)$, i.e., $2\lvert E_C\rvert/(\lvert C\rvert\lvert C-1\rvert)$
25	$\mathit{diameter}(G_C)$	diameter of G_C, i.e., max$\{\mathit{dist}(u, u') \mid u, u' \in C\}$
26	$\mathit{radius}(G_C)$	radius of G_C, i.e., min$\{\mathit{dist}(u, u') \mid u, u' \in C\}$
27	$\mathit{avg\_path\_len}(G_C)$	average length of shortest paths in G_C, i.e., mean$\{\mathit{dist}(u, u') \mid u, u' \in C\}$

TABLE II: Feature description

influenced a lot of members, then the chance that the user follows him is high as well. Recall in our leader discovery algorithm, only users with high influence scores are considered as leaders of a community, we simply use this value to represent the influence power.

Each community normally conducts several events and leaders for these events are often different. If the leader ℓ we exploit, i.e., the closest leader who has conducted the event, has a significant higher influence score than others in $\{\ell_{C,e} \mid e \in E(C)\}$, then it is evident that ℓ is stronger in influencing community members than other leaders. We include *influence ratio* of ℓ as one feature, and it is defined as

$$\mathit{infratio}(\ell) = \frac{\mathit{infscore}(\ell)}{\sum \mathit{infscore}(\{\ell_{C,e} \mid e \in E(C)\})}.$$

Previous works [8], [16] show that the number of events a user conducts and the number of his friends are strong indicators for predicting his influence. We take these corresponding data of the leader as features. In [17], the authors propose a metric called neighbor ratio, it is defined as $|f(u, C)|/|f(u)|$. The results in [18] show that the neighbor ratio is correlated with the user's influence. Thus we take neighbor ratio as another feature. As stated in [15], users with high centrality scores are considered more influential than others. Also in Section IV, we have already shown that the leaders we found have higher value regarding closeness and betweenness. We add these two metrics as well as the average neighbor ratio and eccentricity into our leader related feature set.

User features. Besides the leaders, the user himself also plays an important role when predicting whether he will conduct the event. We have similar features for users as the ones of leaders, excluding the influence score and influence ratio.

Leader-user features. Considering the relationship between leaders and users, first of all, if the leader ℓ and the user u know each other, then u can get influence from ℓ directly. Intuitively, the probability that u conducts the event could be high. Second, suppose that there exist 2-depth paths between ℓ and u, i.e., they share common friends. According to [19], more common friends two users share, more chances that they trust each other. Therefore, it is more probable for u to follow ℓ on this event. Third, if the distance between them in G_C is short, then the influence can be easily propagated through.

These intuitions are captured through three features including whether ℓ and u are friends ($isfri(u, \ell)$), the number of their common friends which is also known as embededness ($dep2cnt(u, \ell)$) and the distance between them ($dist(u, \ell)$).

Community features. As mentioned before, all users in the community form a graph G_C. In [16], the authors have shown that the graph structure has a big impact on influence propagation. Regarding the community related features, we put the community density as one feature, which is formally defined as $density(C) = \frac{2|E_C|}{|C||C-1|}$. Other community related features include G_C's radius, diameter and average path length.

C. Evaluation

Machine learning techniques. We adopt logistic regression as our classifier. The evaluation metrics we use are ROC curve as well as AUC (area under the ROC curve).

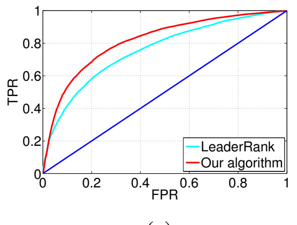
Experiment setup. We extract community and check-in information from the 11 metropolises and only keep the check-ins of each community at each location that is first performed by its leaders. Note that in Section IV we use all check-ins of a community to discover its leaders. However in event prediction, if we do the same, the result will be biased since we already use the events, i.e., check-ins at locations, that we aim to predict for leader discovery. In order to avoid this, we split the locations of each community into half, the first part is used to discover leaders and the second one is used for training and testing. For the second part, we split the training and testing set by communities. Concretely, we use the data from 70% of the communities for training and the data for the rest 30% communities for testing.

Baseline model. As mentioned in Section IV, leaders we found are quite different from the leaders discovered by LeaderRank. Therefore, we want to see which leaders are more effective for event prediction. Since LeaderRank is purely based on network structure, we don't need to split the dataset into two and use one for leader discovery. We establish the same set of features for the leaders of LeaderRank. For $infscore(\ell)$ and $infratio(\ell)$ (features (1) and (2)), we use the leaders' leader ranking score instead.

Results. The ROC curves in Figure 1a has shown that our prediction achieves a good performance with AUC equal to 0.818. It outperforms the baseline model whose AUC is 0.758. This indicates that the leaders discovered by the event based approach are more effective than the leaders by the graph based approach when predicting a member's future behavior. By studying the learned logistic model, important features include leader's influence ratio (2), the number of check-ins (3), betweenness (9), the number of the user's check-ins (11), community size (23) and density (24).

D. Event Popularity and Prediction

As discussed in Section III, we take event's popularity into account when finding leaders. Thus it is natural to ask how much the event's popularity can affect the prediction results. We calculate the average AUC value as a function of location diversity and plot the result in Figure 2. With the increase of location diversity, our predictor's performance grows almost linearly. This indicates that it is easier to predict whether a user will go to a less popular place than a popular one. This demonstrates our intuition that when an event is not popular, leader's influence takes an important role for a member to decide whether to conduct this event.



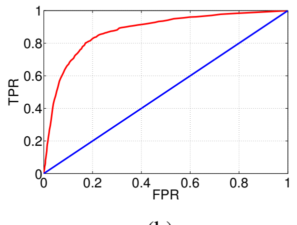


Fig. 1: (a) ROC curve for event prediction (AUC for our algorithm is 0.818, and AUC for LeaderRank is 0.758). (b) ROC curve for time prediction (AUC = 0.897).

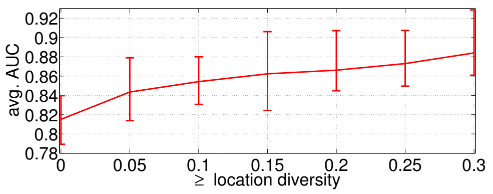


Fig. 2: Avg. AUC as a function of location diversity.

VI. Time Prediction

We have shown that a user's closest leader in a community can be used to effectively predict whether he will conduct a given event. In many cases, not only *whether* but also *when* a member will conduct a certain event is important. Recall the example in Section I, a shop gives leaders coupons or free products to expect them to attract more customers for its new products. However, if most of users that follow leaders go to the shop after 2 weeks or even 1 month, then the promotion probably is already over. The marketing strategy won't succeed. On the other hand, if an attacker, such as a stalker, learns when a user will go to a certain place, then the user's privacy is seriously threatened. In this section, we aim to see if it is possible to predict how soon that community members will follow their leaders' influence.

A. Problem Definition

There are several ways to formalize the problem, such as predicting when a member will conduct an event after the leader, i.e., as a regression problem. However, knowing the exact time (or exact time interval) is not that necessary. In most of the cases, a shopping center or a restaurant just needs an approximate time interval. Thus, we formalize the problem as a classification task: instead of knowing the exact time, we focus on whether a user will conduct the event within a certain time period or not. Formally, the problem is defined below.

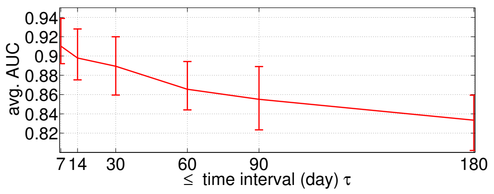


Fig. 3: Avg. AUC as a function of time interval.

Time prediction problem. For a community C with its leader set $L(C)$ and an event $e \in E(C)$, given a member u's closest leader $\ell \in L(C)$ that has conducted e, we predict whether u will perform e within a given time interval τ.

B. Evaluation

We adopt the same features as presented in Table II and use logistic regression again as our classifier. The time interval τ is an application dependent parameter. It can be a day, a week or a month. In the experiment, we set it to 30 days.

The ROC curve is depicted in Figure 1b. Our classifier achieves a surprisingly strong performance with AUC equal to 0.897. For the logistic model that we have learned, the important features are quite similar to the ones for the event prediction model. Due to the page limit, we omit them here.

C. Time Interval and Prediction

We are also interested in whether the time interval has impact on the prediction result. For example, if a user goes to a restaurant 6 months after his closest leader went there, then it is quite possible that this is not a leader driven event at all. Thus, the influence from leaders should decay with the increase of time. We pick 6 values for the time interval τ including 1 week, 2 weeks, 1 month, 2 months, 3 months and 6 months, and execute our classifier with these time intervals, respectively. The results depicted in Figure 3 coincide with our intuition. With the increase of time interval, the AUC value decreases almost linearly.

VII. Related Work

There exist many algorithms on leader discovery. One type of the algorithms are based on the structure of the social graph, solutions include centrality score, PageRank, etc. The LeaderRank algorithm [9] belongs to this type as well. In [9], the authors have shown that LeaderRank is more effective than PageRank and rather robust against manipulations and noisy data. The other type of leader discovery algorithms are based on mining users' past behaviors. Algorithms of [7] and [8], on which we build our solution, fall into this class.

Besides leader discovery, the study on the use of leaders is also meaningful. However to the best of our knowledge, this hasn't been much addressed in the literature. The work in [8] predicts the actual influence scores of each user based on his previous influence scores as well as some attributes. Our work instead has used the leaders to predict community members' behaviors (whether and when) in the future.

VIII. Conclusion

We have shown how to explore social influence of leaders in a community for event prediction of community members. This allows us to answer questions such as whether and when a user will perform an event in an effective way. Leadership is a common social influence form and understanding it could help to build appealing applications. So far, we have mainly focused on a location-based social network dataset. We believe that our research can be applied for other types of event prediction in social networks, and will conduct more studies in the future.

References

[1] R. K. Merton, *Social Theory and Social Structure*. Simon and Schuster, 1968.

[2] D. Easley and J. Kleinberg, *Networks, crowds, and markets: reasoning about a highly connected world*. Cambridge University Press, 2010.

[3] W. G. Rowe and L. Guerrero, *Cases in leadership*. Sage, 2012.

[4] M. Chemers, *An Integrative Theory of Leadership*. Psychology Press, 2014.

[5] J. Yang, J. J. McAuley, and J. Leskovec, "Detecting cohesive and 2-mode communities indirected and undirected networks," in *Proc. 7th ACM International Conference on Web Search and Data Mining (WSDM)*. ACM, 2014, pp. 323–332.

[6] J. Pang and Y. Zhang, "Exploring communities for effective location prediction," in *Proc. 24th World Wide Web Conference (Companion Volume) (WWW)*. ACM, 2015, pp. 87–88.

[7] A. Goyal, F. Bonchi, and L. V. Lakshmanan, "Discovering leaders from community actions," in *Proc. 17th ACM Conference on Information and Knowledge Management (CIKM)*. ACM, 2008, pp. 499–508.

[8] E. Bakshy, J. M. Hofman, W. A. Mason, and D. J. Watts, "Everyone's an influencer: quantifying influence on twitter," in *Proc. 4th ACM International Conference on Web Search and Data Mining (WSDM)*. ACM, 2011, pp. 65–74.

[9] L. Lu, Y.-C. Zhang, C. H. Yeung, and T. Zhou, "Leaders in social networks, the Delicious Case," *PLoS ONE*, vol. 6, no. 6, p. e21202, 2011.

[10] A. Goyal, F. Bonchi, and L. V. Lakshmanan, "Learning influence probabilities in social networks," in *Proc. 3rd ACM International Conference on Web Search and Data Mining (WSDM)*. ACM, 2010, pp. 241–250.

[11] S. Nilizadeh, A. Kapadia, and Y.-Y. Ahn, "Community-enhanced de-anonymization of online social networks," in *Proc. 21st ACM Conference on Computer and Communications Security (CCS)*. ACM, 2014, pp. 537–548.

[12] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *Proc. 17th ACM Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2011, pp. 1082–1090.

[13] J. Cranshaw, E. Toch, J. Hone, A. Kittur, and N. Sadeh, "Bridging the gap between physical location and online social networks," in *Proc. 12th ACM International Conference on Ubiquitous Computing (UbiComp)*. ACM, 2010, pp. 119–128.

[14] H. Pham, C. Shahabi, and Y. Liu, "EBM: an entropy-based model to infer social strength from spatiotemporal data," in *Proc. 2013 ACM International Conference on Management of Data (SIGMOD)*. ACM, 2013, pp. 265–276.

[15] J. Sun and J. Tang, "A survey of models and algorithms for social influence analysis," in *Social Network Data Analytics*. Springer US, 2011, pp. 177–214.

[16] J. Cheng, L. Adamic, P. A. Dow, J. Kleinberg, and J. Leskovec, "Can cascades be predicted?" in *Proc. 23rd International Conference on World Wide Web (WWW)*. ACM, 2014, pp. 925–936.

[17] A. Mehler and S. Skiena, "Expanding network communities from representative examples," *ACM Transactions on Knowledge Discovery from Data*, vol. 3, no. 2, p. Article No.7, 2009.

[18] M. Genois, C. L. Vestergaard, J. Fournet, A. Panisson, I. Bonmarin, and A. Barrat, "Data on face-to-face contacts in an office building suggests a low-cost vaccination strategy based on community linkers," *CoRR*, vol. abs/1407.7017, 2014.

[19] P. W. Holland and S. Leinhardt, "Transitivity in structural models of small groups." *Comparative Group Studies*, 1971.