Influence Maximization in Location-Based Social Networks

Ivan Suarez, Sudarshan Seshadri, Patrick Cho
CS224W Final Project Report

Abstract

The goal of influence maximization has led to research into different influence propagation models and influence probability generation models. However, most research done in influence maximization do not take into account an important factor in human interaction: location. By analyzing how influence spreads in a location-based social network, we aim to contextualize some of these influence propagation models and influence probability generation models in a real world network that explicitly defines user activity in terms of location. Through this paper, we aim to answer some of the following questions: How does the propagation of influence depend on how often an individual checks into a specific location? If the social network is unknown, are we still able to extract meaningful results? Finally, how does the propagation of influence depend on the distance between users and locations?

Introduction and Prior Work

Social networks play an important role in the spread of information, ideas, and influence among its members [3]. The importance of these social networks have become even more prominent with the rise of large-scale social networks that span very large distances. With the rise of platforms such as Facebook, friendships in social networks are no longer limited by face-to-face interactions. This phenomenon leads to an important question: how should influence propagation in large-scale social networks be interpreted for brick-and-mortar stores, whose customer bases, unlike social networks, are generally geographically limited?

The spread of influence in social networks was first analyzed by Kempe et. al in their paper titled "Maximizing the Spread of Influence through a Social Network" [3]. In their paper, Kempe et al. give an overview of five different models of influence propagation and show that finding the optimal initial set for all these models is NP-hard. The paper then describes a simple hill-climbing algorithm for influence maximization. This algorithm can be used to find a set of k users to initially activate, and the expected influence set size from this set of k users is within a factor of $(1 - \frac{1}{e})$ of the optimal solution.

However, even though the hill-climbing algorithm is faster than finding the optimal solution, the hill-climbing algorithm can still be quite slow when dealing with a large number of nodes. In [4], Leskovec et al. introduce the CELF algorithm, which is an optimization of the hill-climbing algorithm. CELF exploits the following fact from submodularity: given two sets $A \subseteq B$ which are subsets of the nodes V, the marginal gain of adding a node s into B is no greater than that of adding it into A. Hence, for two nodes $u, v \in V$ and sets $A \subseteq B \subseteq V$, suppose that the marginal gain of u for A is greater than that of v for A. When considering B, if the marginal gain of u for B is greater than that of v for A, we need not compute the marginal gains of v for B. This avoids unnecessary computation. The above idea leads to a dramatic speed up in runtime.

One key assumption that [3] and [4] both make is that the social network's influence probabilities are known beforehand. In general, these influence probabilities are unknown and have to be inferred from data. In their paper titled "Learning Influence Probabilities in Social Networks" [2], Goyal et al. describe techniques to learn influence probabilities in social networks. They do so by analyzing an action log that specifies a specific action by a specific user at a certain time. The authors provide a variety of ways to model the influence a user will have over another user. We investigate the simplest model: static models, in which the influence probability is time invariant.

Influence maximization and learning influence probabilities have been explored in the context of social networks. However, these influences are often only accurate for online activities. For offline activities such as visiting a brick-and-mortar store, Alice's influence on Bob depends heavily on the distance from Bob to the store. To explore this problem, we use a location-based social network dataset that was also explored in [1]. In [1], the authors present a model aimed at better predicting human mobility by using the following facts: humans tend to visit nearby locations periodically in both a temporal and spatial fashion. In rarer cases, humans visit far locations in a sporadic fashion spatially and the locations visited are, in part, influenced by the friendships that they have.

Throughout this paper, we denote an action by a user visiting a location.

Data Collection Process

We use a location-based social network dataset from Brightkite that can be found at http://snap.stanford.edu/data/loc-brightkite.html for our project. The dataset includes a social graph and an action log. The action log specifies a user checking into a certain location at a specific time. The action log also includes the latitude and longitude of a specific location. However, the dataset does not include the time at which a user joins the social network. This data is required for learning influence probabilities as it would be incorrect to give influence credit to a friend that was made after doing an action. There are two ways to correct this problem. The first way would be to assume that all friendships were made at the beginning of time. The second way would be to assume that a user only joins the social network upon doing his first action and makes all his friends (who have done an action before him/her) upon joining the social network. For ease of computation, we have selected the prior assumption. Moreover, the dataset does not provide the home location of all users. This missing data would hinder analyses on spatial effects on influence probabilities. To overcome this problem, we infer user home locations by taking the weighted average of a user's check-in locations.

Summary Statistics of Dataset

We start by doing some simple analysis on the dataset. We notice all three distributions in Figures 1 through 3 follow cumulative distributions of power laws. Figure 1 is simply the cumulative degree distribution, and has a typical shape for a social network. Figure 2 denotes number of locations versus the number of unique user visits, which tells us that locations with many unique visitors are increasingly rare. In particular, we note that 88% of the locations in the dataset were only visited by one unique visitor. This foreshadows the sparsity of useful data in the Brightkite dataset. Figure 3 denotes number of unique locations a user visits versus the number of users, which tells us that users who visit a large number of unique locations are increasingly rare. In particular, about 30% of users visit only one unique location.

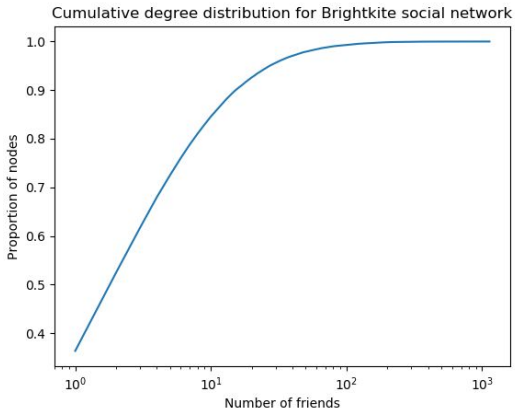

Figure 1: Cumulative Degree Distribution for Brightkite Social Network

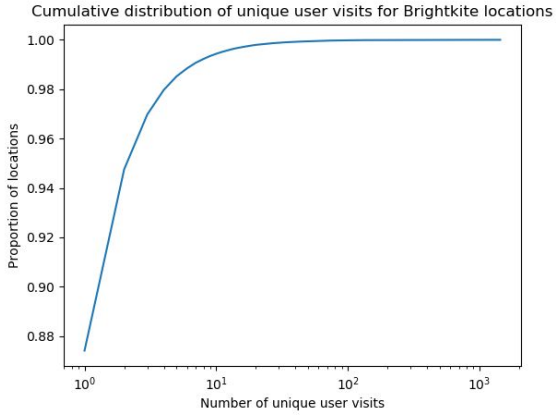

Figure 2: Cumulative Distribution of Unique User Visits per Location

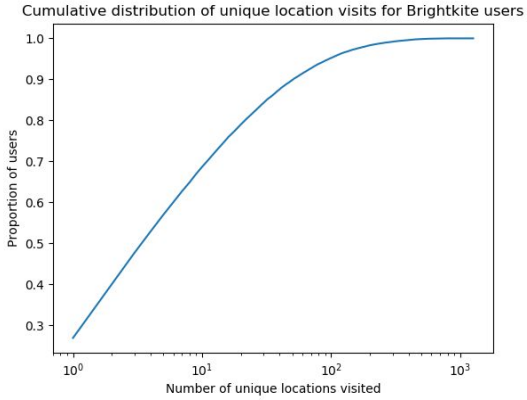

Figure 3: Cumulative Distribution of Unique Location Visits per User

Mathematical Formulations

Joint Influence Probability is Monotone and Submodular

Similar to [3], we make the assumption that the probabilities of neighbors influencing a certain user are independent of each other. That is,

$$P_u(S) = 1 - \prod_{v \in S(u)} (1 - p_{vu})$$

where *S(u)* represents the active (influenced) neighbors of u and p_{vu} represents the influence probability of v to u. Consider the effect of adding a user w to set S:

$$\begin{aligned} P_u(S \cup \{w\}) &= 1 - (1 - p_{wu}) \prod_{v \in S(u)} (1 - p_{vu}) \\ &= 1 - (1 - p_{wu})(1 - P_u(S)) \\ &= P_u(S) + (1 - P_u(S))(p_{wu}) \end{aligned}$$

This equation shows that $P_u(S)$ increases monotonically as the size of set S increases, since $(1 - P_u(S))(p_{wu})$ is nonnegative. Furthermore, if S is a subset of T,

$$\begin{aligned} & P_u(S \cup \{w\}) - P_u(S) - P_u(T \cup \{w\}) + P_u(T) \\ &= (1 - P_u(S))(p_{wu}) - (1 - P_u(T))(p_{wu}) \\ &= (P_u(T) - P_u(S))(p_{wu}) \geq 0 \end{aligned}$$

where the last step is based on monotonicity of P. The first equation gives us a way to quickly update $P(S)$ incrementally when a new neighbor w becomes active without going through all other previously active neighbors.

Algorithms

Learning Influence Probabilities for Static Models

In this algorithm, we want to learn the influence probabilities based on the assumption that influence probabilities stay constant over time. Two models, the Bernoulli and Jaccard models, are as follows. Let A_v store the number of locations that user v visits, let A_{v2u} store the number of times that user u visits a location that user v has already visited, and let $A_{v|u}$ store the number of locations either user v or user u visits. By the Bernoulli method: $p_{vu} = \frac{A_{v2u}}{A_v}$. By the Jaccard method: $p_{vu} = \frac{A_{v2u}}{A_{v|u}}$.

As an extension, we can also allocate partial credits to neighbors. Concretely, if k neighbors of user u did a specific action before user u did the same action, then we can split the credit equally among these k neighbors.

The provided algorithm in [2] efficiently calculates the influence probability for each pair of users. It first sorts the action log by action followed by timestamp. Then, for each action, we iterate through the timestamps while maintaining a list of users who have already done the action. In this way, we can efficiently calculate the influence probabilities for the Bernoulli and Jaccard model either with or without partial credits. In particular, for partial credits, we iterate through all neighbors and find the number of activated neighbors, N. The partial credit from v to u is the reciprocal of N.

Evaluation of Influence Probabilities

In [2], there is a basic evaluation function, which uses a test set to evaluate the accuracy of the p_{vu} found from the learning phase. This is done by sweeping a threshold θ from 0 to 1, and if $p_{vu} > \theta$, predict that v influences u to perform an action. The test set is used to categorize each prediction as either a true positive, false positive, true negative, or a false negative. Sweeping θ produces an ROC curve.

Finding Expected Size of Influence Set

Since finding an influence set of k users is a random process, it would not make sense to pick k users that maximizes possible spread of influence among the rest of the users. Rather, what we aim to do is find a set of k users that maximizes the expected, not best case scenario, spread of influence. Hence, we require an algorithm that finds the expected size of the k initial users' influence set.

Given an initial set of k users, influence probabilities among users, and a number of trials to run, the algorithm essentially computes an influence set for the given k users for each trial and then returns the average size of the influence sets among all trials.

Using CELF for Finding the Set of k Users to Initially Influence

Once we have the estimated probabilities of influence for each pair of users v, u in the network, we now implement CELF, as given in [4], to greedily find a set of k users to initially influence in order to cause the largest cascade effect. We note that this algorithm leads to a massive speedup in the greedy influence maximization algorithm, as expected.

Inferring the Social Network Graph from Check-in Data

Suppose a restaurant owner knows which customers have visited the restaurant via their transactions, but does not know the friendship relationships between customers. In order to perform influence maximization, some sort of estimated social network graph G' is required. We experimented with different ways of generating G' from check-in data only.

For our first attempt, we notice that when computing p_{vu} for each user, the algorithm from [2] stipulates that there must be an edge (v,u) in the social network graph G. We remove this requirement and compute p_{vu} for all users such that v and u had at least one visited location in common. Then, to infer G', we sweep a threshold t from 0 to 1. For a fixed value of t, we add edge (v,u) to G' if and only if $p_{vu} \geq t$. After testing each computed G'_t against the actual graph G, we find we either had an extremely large false negative count, or a zero true positive count. Therefore we reject this method. We hypothesize that this method fails because we assume that users who visit the same place are friends in the underlying social network graph. This is not necessarily true and is most often not the case. For example, you would not typically deem everyone who goes to your favorite coffee shop as your friends.

Our second attempt was to calculate the probability of an edge (v,u) in G' as: $f_{vu} = \frac{|Visited(v) \cap Visited(u)|}{|Visited(v) \cup Visited(u)|}$ where $Visited(v)$ is the set of locations v visited after a threshold date. Again, we add an edge (v,u) to G' if and only if $f_{vu} \geq t$ for a threshold parameter t. Unfortunately, this method gave very few true positives. This is because many people who are friends in the social network may not necessarily check in to the same set of locations. As we noted earlier, 88% of users checked in to only one unique location. We believe this method would work better on a dataset in which most users performed a sufficiently large number of unique actions.

Our third attempt was to let G' be a fully connected graph. However, this led to nearly any target set of one user activating the entire graph. This method gives far too much influence to any one user.

Our final attempt was, for each user v, randomly sampling five edges (v,u) such that v and u visited at least one location in common. We sampled five edges because this is the closest integer estimate of the average degree for each user. This method performed better than the previous attempts. We refer to the "Blind Greedy" algorithm as one in which we run the CELF algorithm after inferring G' through this random sampling of edges. We note that the inferred G' resembles an Erdős-Rényi random graph. Although Erdős-Rényi random graphs are generally not accurate models for social networks, this assumption worked reasonably well for our purposes.

Results

After implementing the learning algorithm described in [2], we also implemented the basic evaluation function given in [2]. Our implementation produced the following ROC curve, which we have

plotted in Figure 4. We calculated p_{vu} in two ways: first with the Bernoulli method, and second with the Jaccard Index method.

Comparing our ROC curves with those found in [2] after basic evaluation shows that our true positive rate is considerably lower than we would have wanted. This may be due to the nature of the dataset we are using. Specifically, location check-ins make for sparse data when compared to other possible actions in a social network, such as joining a group in a social network platform.

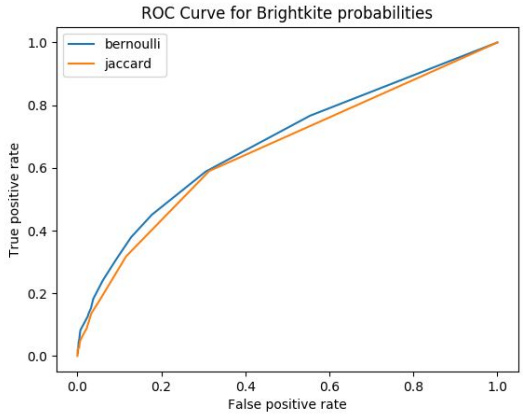

Figure 4: ROC curves for Brightkite influence probabilities

After estimating influence probabilities with the Bernoulli method, we apply the CELF algorithm given in [4] to compute the target set of k users to initially influence under the two following scenarios. First, with the underlying social network graph ("Greedy") and second without the social network graph ("Blind Greedy", as explained in the algorithms section "Inferring the Social Network Graph from Check-in Data"). We compare the results with those obtained from picking k users based on each of four metrics separately: highest degree, highest betweenness centrality, highest clustering coefficient, and random choice.

Figures 5 and 6 display the results of CELF with the Greedy and the Blind Greedy methods. For all of the algorithms used, we created a splice of the social network where we only include users that are located within 500 kilometers of a particular location and assume that the probability of influence among users is independent of where they are relative to the center location. We chose 500 kilometers mainly because the data at hand is very sparse in terms of location check-ins, so using a range smaller than 500 kilometers would often result in having fewer than a thousand users in the graph.

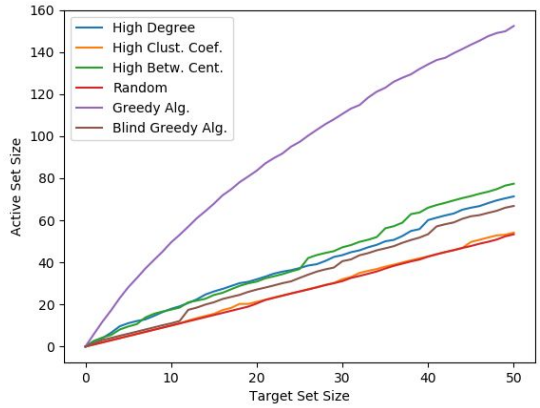

Figure 5: Comparison of methods with network splice centered in New York

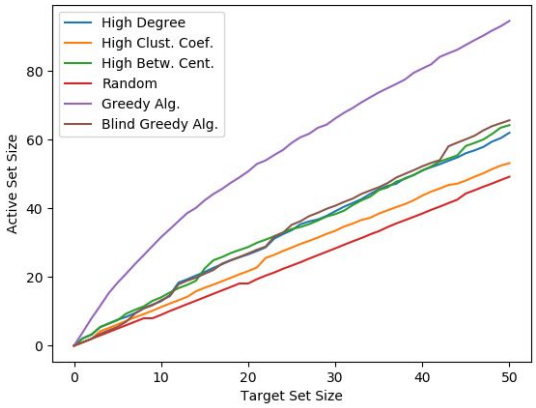

Figure 6: Comparison of methods with network splice centered in Oakland

Using the Greedy method to compute the set of k users to initially influence yields significantly better results than any of the other given methods. The expected active set size after influencing k users is more than double of those from the other methods. In addition, Greedy runs reasonably quickly on graphs with a few thousand nodes, due to the CELF optimization.

Without knowledge of the social network, we see that Blind Greedy performs significantly worse than the Greedy method. Nonetheless, Blind Greedy, which operates without the social network, does enable us to pick out sets of k users that yield higher expected influence set sizes than when picking k users either by random choice or by highest clustering coefficients. This demonstrates that solely having the location check-in data enables us to perform better than naively picking k users at random.

We experimented different ways for estimating probabilities of influence between two users. One such way may be to account for multiple influence: if user v visits location L and later on user u visits location L n times, then the model will give user v credit for the n visits of user u to location L. This is in contrast to the original model of probability estimation where it would give user v credit for only one visit of user u, thus ignoring subsequent visits by user u.

Another way would be to incorporate Laplace smoothing, where we automatically assume that user v has influenced user u into visiting λ times, and that user v has failed to influence user u into visiting λ times. For example, the Bernoulli method estimate of p_{vu} changes from $p_{vu} = \frac{A_{v2u}}{A_v}$ to $p_{vu,\lambda} = \frac{A_{v2u}+\lambda}{A_v+2\lambda}$. This slightly shifts all probabilities towards 0.5, thus preventing any influence probabilities between friends from being either 0 or 1. We used $\lambda = 1$ in our experiments.

Figures 7 and 8 display the results of choosing k users intended to maximize the expected influence set size when estimating probabilities using the idea of multiple influence. Figures 9 and 10 display the results of the listed methods when using Laplace smoothing.

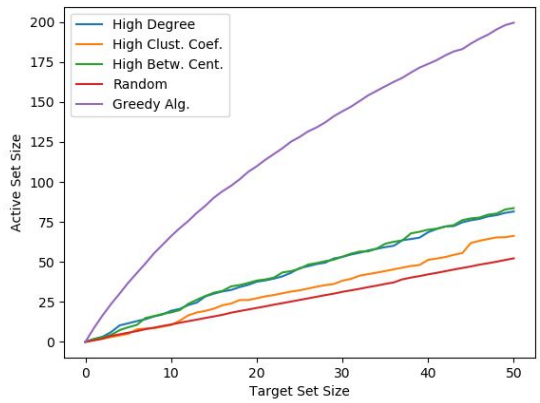

Figure 7: Comparison of methods with network splice centered in New York (multiple influence)

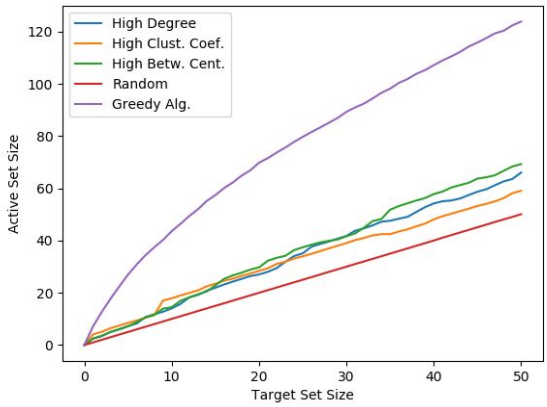

Figure 8: Comparison of methods with network splice centered in Oakland (multiple influence)

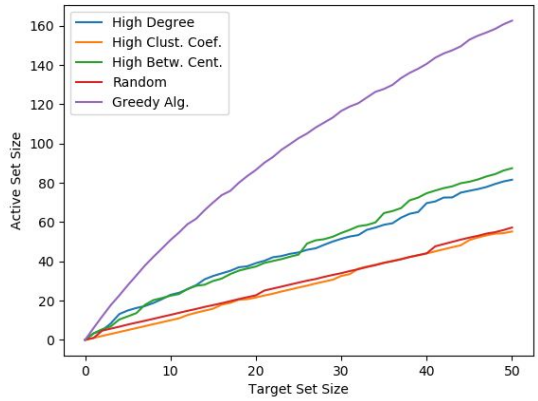

Figure 9: Comparison of methods with network splice centered in New York (Laplace smoothing)

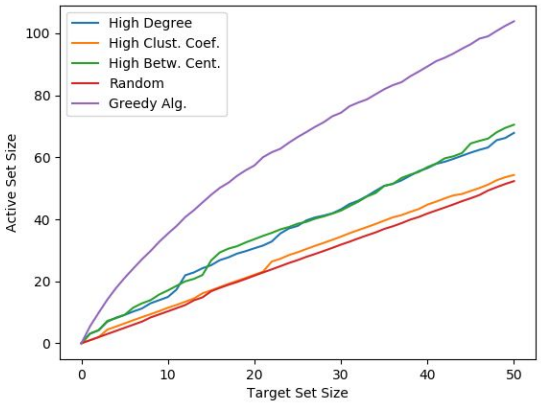

Figure 10: Comparison of methods with network splice centered in Oakland (Laplace smoothing)

As we can see in Figures 7 through 10, modifying our original model of influence probability estimation with either the incorporation of multiple influence or Laplace smoothing changes the sizes of expected activation sets. We notice that influence maximization via the Greedy algorithm is still feasible. Therefore, in scenarios where we deem multiple influence or Laplace smoothing to be important, we may safely incorporate such modifications into the model of influence probability estimation and remain confident that the results will be meaningful.

Although the current model of probability estimation performs well, we have the following concern: is the current model actually reflective of true influence between users? Indeed, we do not take user location into account when computing the best set of k users to initially influence, apart from removing far away users entirely. Hence, one modification may be to compute influence probabilities conditioned on user location. If user v has influence over user u, the influence is likely to be stronger if user v recommends user u to a nearby location as compared to a faraway location. Therefore, influence

probabilities should be dependent on distance between the user to be influenced and the location L to check into.

In order to determine the relationship between the distance from a user to a location, and the likelihood of the user visiting the location, we hypothesize the following relationship:

$$y = \sigma(a + bd + cd^2)$$

where y is the probability of a user visiting a location, σ is the sigmoid function, and d is the distance in kilometers between the user and the location.

We construct the following dataset. For every check-in, we get a training example where d is the distance in kilometers between the user and the location and $y = 1$. These training examples represent all the positive examples. For negative examples, we randomly sample a location and a user. We use a higher sampling bias for locations that were frequently visited and users who visit more locations. For each location and user, if the user has never visited the location, we add it to our dataset with $y = 0$. We sample as many negative examples as there are positive examples so that there will be the same number of examples in each class. Finally, we fit a logistic regression model on 70% of the dataset, leaving the rest for testing. We achieve a test accuracy of 87.95%.

The logistic regression model allows us to predict the probability of a user visiting a location based on the distance between the user and the location. Hence, given a location L, we can now update the influence probabilities on the graph by simply multiplying all edges with this probability. Concretely, we update:

$$p_{vu}' = p_{vu} \cdot \sigma(a + bd + cd^2)$$
$$\text{where } d = dist(user\ u,\ location\ L)$$

Intuitively, this update makes sense. As user u gets further away from location L, user v's influence on u diminishes. We call the original weights P and the updated weights P'.

Figure 11 displays the results of taking user location into account when estimating influence probabilities. We take splices of the social network where we only consider users within n kilometers of the center location of choice (in this case, Oakland). We fix $k = 15$ users to initially influence. Then we vary the value of n and plot the resulting expected activated set size produced by the Greedy algorithm for both P and P'.

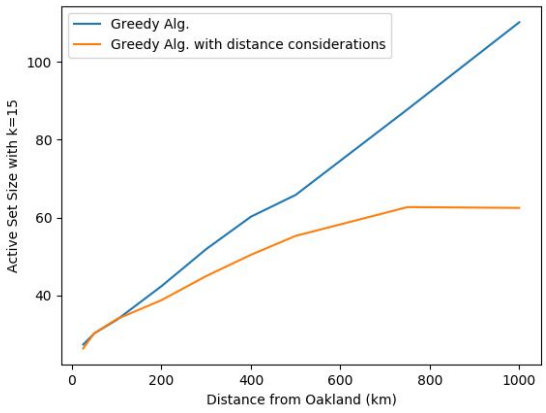

Figure 11: Comparison of the CELF algorithm using P and P' (centered at Oakland).

As demonstrated in Figure 11, taking user location into account has a significant impact on expected activation set size. This is to be expected: it is much less likely for user *v* to influence user *u* to visit location *L* if the distance from user *u* to location *L* is significant (200 kilometers for example). Notice that without distance considerations, the expected activation set size increases linearly with *n*. However, more realistically, there is diminishing value for increasing *n*. This is because users who are very far away from *L* are likely to have influence over other users who are very far from *L* and therefore not likely to visit *L* even if influenced by a friend. Without taking into account how distance changes influence probability, the Greedy algorithm would severely overestimate its effectiveness, potentially impacting the budgeting of an agent wishing to perform location-based influence maximization.

Conclusion

By algorithmically learning influence probabilities in static models to estimate influence probabilities, we find that special considerations such as multiple influence or Laplace smoothing still provide valid models for which to perform influence maximization. In order to augment these models with location-based considerations, our model should not assume that the probability of *u* being influenced to visit location *L* is independent of the distance from *u* to *L*. When dealing with location-based data, it is necessary to consider the distance between *u* and *L*. Although the expected size of the influence sets decrease, this is a more realistic representation of influence probability between any two users *v* and *u*.

When we do not have knowledge of the social network graph, we may still be able to infer meaningful information from the location check-in data alone. Inferring the social network graph from the check-in data allows an agent to perform influence maximization better than just picking random target users. Indeed, when operating a food chain restaurant for instance, the social network graph for customers is often unknown; we would only have the location check-in data. In this case, the data would be when a customer purchases a meal, assuming that we are able to record the information via a rewards program for example.

Individual Contributions

Sudarshan Seshadri - Plot generation, Data preprocessing, CELF implementation, location filtering, Blind Greedy
Ivan Suarez - Influence set calculation, F1 score of inferred graphs, Bulk of report writing
Patrick Cho - Data preprocessing, Learning probability of influence from check-in data, Logistic Regression

References

[1] E. Cho, S. A. Myers, J. Leskovec. Friendship and Mobility: Friendship and Mobility: User Movement in Location-Based Social Networks ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2011.

[2] A. Goyal, F. Bonchi, L.V.S. Lakshmanan. Learning influence probabilities in social networks. In Proc. WSDM, 2010.

[3] D. Kempe, J. Kleinberg, E. Tardos. Maximizing the Spread of Influence through a Social Network. In Proc. KDD 2003.

[4] J. Leskovec et al. Cost-effective Outbreak Detection in Networks. In KDD 2007.