Explaining Geolocations Predicted by Label Propagation

Haiying Huang, Yu Hou, Cheng Lu, Zuer Wang, Yuhan Shao
UCLA
{hhaiying, yuhou316, lucheng9, zw85, yuhan17}@g.ucla.edu
UID: 804757410, 105711952, 405726283, 905688919, 305522696

1 Introduction

Geolocation inference for social media users has been an essential component for applications ranging from disaster management (e.g COVID 19) to recommendation systems to public opinion analysis. Current Geolocation models fall into two categories: *text-based* and *(social) network-based*. Text-based models try to locate user based on his/her tweet texts and profile data and are typically driven by NLP algorithms such as Topic models, RNN, and LSTM. Network-based models, which we focus on in this paper, utilize one's social network (e.g. a tweet mention graph) and infer one's location from a small set of labeled users whose location are known using label propagation (LPA).

Despite the great advance in performance, it remains a question how/why a network-based model assigns a specific location tag to an individual. Motivated by a recent line of work on explaining/interpreting neural networks, we decide to investigate the problem of explaining a geolocation tagger. The main contributions of our project are: first, we propose *Explained Label Propagation* (**Ex-LPA**) algorithm, which combines Vanilla label propagation with some book-keeping technique to explain each decision it makes. More specifically, the algorithm not only predicts the class of a new user but also keep track of the contribution of each labeled user for the algorithm to come to this decision. Second, we adapt our Explained-LPA to Tweet Geolocation under two circumstances 1) multi-class classification when the target is region names and 2) regression when the target is GPS coordinates and evaluate its performance on GEOTEXT. For the regression case, we choose Spatial label propagation proposed by [3] as a baseline. Third, we apply some noval $selection\ method$ in Spatial LPA and cross-compare their performance.

2 Explaining Label Propagation

2.1 Problem Statement

We aim to quantitatively explain the rationals of each prediction made by Label Propagation (**LPA**) algorithms, in the context of tweet geolocation task. We propose to measure the *cumulative* feature importance, that is, how much each labeled node contributes to a particular prediction during the whole propagation process. Then, we can easily identify the most influential input for a particular prediction. We will illustrate the problem setup using the following example.

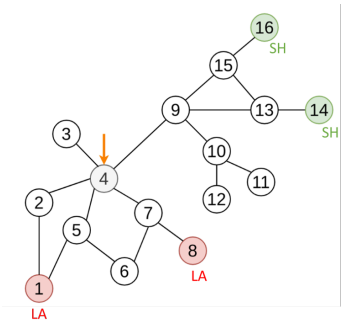
Suppose we have a social network in Figure 1a in which a small set of users $\{n1, n8, n14, n16\}$ are location tagged. And we want to infer the location of user n4. We run LPA over the network, which outputs a label distribution for n4 in Table 1b. Since class LA has a higher prediction score than class SH, node n4 will be tagged with LA. Notice that LPA learns this label distribution by iteratively propagating label from a node to its neighbor. Therefore, to explain this process, we could ascribe the final prediction score $\mathbb{P}(\cdot)$ of a node back to its neighbors and recursively back to each labeled node. In this example, the score $\mathbb{P}(n4 = \text{LA})$ must originate from the two labeled nodes whose class is LA, i.e. n1 and n8, and n1 probably makes more contribution since it is closer to n4 in the network. By integrating some bookkeeping technique into LPA, we can quantify their contribution in the form of a feature importance matrix as shown in Figure 1c.

Before we proceed to the technical details of *Explained Label Propagation*, note that:

1. Our approach for explaining LPA is analogous to previous work on analyzing the feature importance for SVM [1], Random Forest [6], and CNN [8].
2. Despite we limit our discussion to tweet geolocation here, our methodology can be easily adapted to any LPA model, including variants of LPA with nonlinear propagation rules, such as Spatial Label Propagation [5].

2.2 The Explainable Algorithm

In this section, we formally present *Explained Label Propagation* (**Ex-LPA**) which not only predict the classes for unobserved nodes but also explain the predictions it made. First, let us introduce some notations. Consider an undirected graph $G = (\mathbf{V}, \mathbf{E}, \mathbf{W})$ and a label set $\mathcal{Y} = \{1, 2, \ldots, m\}$. A small set of nodes $\mathbf{X}_L \subseteq \mathbf{V}$ are labeled according to $\mathbf{Y}_L$, which contains the ground truth labels. The remaining nodes $\mathbf{X}_U = \mathbf{V} \setminus \mathbf{X}_L$ are not labeled. For short, let



(a)

Class	Prob
LA	0.7
SH	0.3

(b)

Class	n1	n8	n14	n16
LA	0.7	0.3	0	0
SH	0	0	0.5	0.5

(c)

Figure 1: Explaining LPA a) Social network; b) Prediction for node n4; c) Feature importance for node n4. As we can see, node n1 is the most influential input to the decision for node n4 tagged with class LA.

$n = |\mathbf{V}|, n_l = |\mathbf{X}_L|$ and $n_u = |\mathbf{X}_U|$. Just like vanilla LPA, we want to estimate label distribution $\mathbf{Y}_U$ for $\mathbf{X}_U$. We define a transition probability matrix $\mathbf{T} \in \mathbb{R}^{n \times n}$ where $\mathbf{T}_{ij}$ represents the probability of jumping from node i to node j:

$$\mathbf{T}_{ij} = \mathbf{P}(i \to j) = \frac{w_{ij}}{\sum_{j' \in \text{neigh}(i)} w_{ij'}}$$

and define a label distribution matrix $\mathbf{Y} \in \mathbb{R}^{n \times m}$ where each row $\mathbf{Y}_i$ represents a soft labeling for node i.

However, Ex-LPA has an additional goal: determine the contribution of $\mathbf{X}_L$ to the estimated $\mathbf{Y}_U$ for X_U. To do this, the algorithm maintains a *feature importance* matrix $\mathbf{W} \in \mathbb{R}^{n \times m \times n_l}$, where $\mathbf{W}_{ick}$ represents the contribution of labeled node $k \in \mathbf{X}_L$ to the prediction score $\mathbf{Y}_{ic}$ for node i getting class c. The algorithm will propagate $\mathbf{W}$ side by side with $\mathbf{Y}$. It proceeds as follows:

1. Initialize $\mathbf{Y}$ and $\mathbf{W}$. For labeled node $i \in \mathbf{X}_L$ with class c , initialize $\mathbf{Y}_i = \mathbf{e}(c) \in \mathbb{R}^m$ and $\mathbf{W}_{ic} = \mathbf{e}(i) \in \mathbb{R}^{n_l}$ since the decision of a labeled node solely comes from itself. For unlabeled node $i \in \mathbf{X}_U$, initialize $\mathbf{Y}_i$ uniformly and $\mathbf{W}_i = \mathbf{0}$.
2. Propagate for one step: $\mathbf{Y} \leftarrow \mathbf{TY}$ and $\mathbf{W} \leftarrow \mathbf{TW}$.
3. Clamp ground truth $\mathbf{Y}_L$ in $\mathbf{Y}$ and $\mathbf{W}_L$ in $\mathbf{W}$. Repeat step 2 until convergence.

Let us look into step 2. For each node i, the updated label distribution $\mathbf{Y}_i$ is a weighted average of its neighbor's labels with respect to $\mathbf{T}$:

$$\mathbf{Y}_{ic} = \sum_{j \in \text{neigh}(i)} \mathbf{T}_{ij} \mathbf{Y}_{jc} \tag{1}$$

To keep consistency with the definition of $\mathbf{W}$, the updated $\mathbf{W}_i$ should also be a weighted average with respect to $\mathbf{T}$:

$$\begin{aligned} \mathbf{W}_{ick} &= \mathbf{W}(\text{node i gets label c due to node k}) \\ &= \sum_{j \in \text{neigh}(i)} \mathbf{P}(j \to i) \mathbf{W}(\text{node j gets label c due to node k}) \\ &= \sum_{j \in \text{neigh}(i)} \mathbf{T}_{ij} \mathbf{W}_{jck} \end{aligned} \tag{2}$$

This consistency guarantees that when the algorithm converges, $\mathbf{W}$ accurately maintains the feature importance as we defined. Step 3 is also critical to prevent the ground-truth information in $\mathbf{Y}$ and $\mathbf{W}$ from "fading" away. The pseudo-code for Ex-LPA is provided in Alg 1.

2.3 Towards nonlinear Propagation rules

Our current Ex-LPA assumes the linearity of update rule for a node's labeling distribution Y based on its neighbors. However, in some circumstances, the update rule (sometimes called select method) may not be linear to take into account continuous labels, such as GPS coordinates as in Spatial-LPA [4]. In this case, we can no longer naively propagate the feature importance matrix $\mathbf{W}$ linearly as in equation 2. Instead, we propose to take the first-order approximation of the used select method. We define this approach as Explain-Generic Label Propagation (Ex-Generic-LPA) which we leave for future work.

Algorithm 1 Explain-Label-Prop(G, X_L, Y_L)

Input:
  G - network structure
  $\mathbf{X}_L, \mathbf{Y}_L$ - labeled dataset
Output: label distribution matrix $\mathbf{Y}$ and feature importance matrix $\mathbf{W}$
Initialize $\mathbf{Y}$ with $\mathbf{Y}_L$, $\mathbf{W}$ with $\mathbf{X}_L$ and $\mathbf{Y}_L$
$\mathbf{T} \leftarrow$ transition probabilities from G
for iteration $t = 1, 2, 3, \ldots, N$ **do**
    $\mathbf{Y} \leftarrow \mathbf{TY}$
    $\mathbf{W} \leftarrow \mathbf{TW}$
    $\mathbf{Y} \leftarrow \text{clamp}(\mathbf{Y}, \mathbf{Y}_L)$
    $\mathbf{W} \leftarrow \text{clamp}(\mathbf{W}, \mathbf{X}_L, \mathbf{Y}_L)$
end for
return $\mathbf{Y}$, $\mathbf{W}$

3 Implementation and Experiment

3.1 Explained Label Propagation

We apply our Explained label propagation (**Ex-LPA**) algorithm on the Tweet Geolocation Task and evaluate its performance. We consider two scenarios based on the type of geolocation labels:

1. When Geolocation labels are discrete classes e.g. region name, we formulate the task as a multi-class classification problem. We apply vanilla Label propagation to predict the class for new users and use **Ex-LPA** in Alg 1 to explain its decisions.
2. When Geolocation labels are GPS coordinates, e.g latitude and longitude, we formulate the task as regression problem. We apply Spatial label propagation [3] which is specifically tailored for propagating spatial labels. Jurgens proposed many different heuristics for selecting a node's location tag base on its neighbors such as *Geometric Median*, *Oja's simplex Median*, *Most Frequent Location* and we will try to implement them and evaluate their performance. Moreover, these select methods are nonlinear so we will use **Ex-Generic-LPA** to explain its decisions.

Dataset we evaluate our Geolocation models on the public dataset GEOTEXT, from which we build the @-mention graph. If possible, we will also try to crawl a larger @-mention network from Tweet API.

Evaluation Metrics Geolocation labels can be coordinates, e.g latitude and longitude or discrete classes, e.g. region name. The former case can be viewed as regression so we will use *mean squared loss*, the latter case can be viewed as classification, so we will use *0-1 loss*. Moreover, instead of predicting the most-likely label c for user i, our Ex-LPA can also explain this decision in terms of contributions, i.e. how much a labeled node (users whose location is known as ground-truth) contributes to this node i being labeled class c.

3.2 Vanilla Label Propagation

3.2.1 Data Preprocessing

We extract the user ids, coordinates and the raw text of user's tweets from the GEOTEXT dataset. As a result, 9475 nodes, 17110 edges, 46006 mentions and 122872 dangling users were extracted. Dangling users are users that are mentioned by others but do not appear in the dataset. Since we only care about bidirectional mentions, we simply discard those dangling users. User's tweets are then further analysed to build a mention graph, stored as a dict(user -> dict(user -> number of bidirectional mentions). User's geolocation is assumed to be the coordinates corresponding to the user's first tweet. In order to do classification, the coordinates are converted to discrete location label using the reverse geocoder package. The label distribution matrix Y, the weight matrix W and the probabilistic transition matrix T are then built using the extracted data.

3.2.2 Algorithm

We formulate the geolocation predication task as a classification problem. Our goal is to predict labels for unlabeled nodes using vanilla label propagation algorithm, which propagate labels of labeled node to unlabeled data. The problem is set up as follow. A graph $G = (V, E)$ is used to represent to social network of all users, where each node is a user and an edge between two nodes indicates friendship between these two users. The weight on an edge is determined by

the number of times these two users mention each other. Let n be the number of users. Let c be the number of labels, i.e. locations. Define a $n \times c$ label distribution matrix Y, where each row is the label distribution of a node. Define a $n \times n$ weight matrix W, where W_{ij} is the weight between node i and j. Compute the probabilistic transition matrix T by row-normalize the weight matrix W:

$$T_{ij} = \frac{W_{ij}}{\sum_{k=0}^{n-1} W_{ik}}$$

The algorithm consists of three main steps:

1. $Y \leftarrow TY$
2. row-normalize Y
3. clamp the ground truth label

The above steps are repeated until convergence. It can be mathematically proved that the algorithm will always converged to a fixed point[9]. We implemented the vanilla label propagation as described above. The label distribution matrix Y is partially filled with the ground truth label using one hot encoding. The label distribution will be uniform for nodes with ground truth label masked out.

3.2.3 Evaluation Method

We will evaluate vanilla label propagation on the GEOTEXT dataset using accuracy as metric. After convergence, a label distribution will be estimated for each node. We will find the entry with the highest value, take the corresponding label as the prediction and compare it to the ground truth label. Since oracle label propagation is the "best case scenario" version of vanilla label propagation. We will use the results from oracle label propagation as an upper bound of accuracy to interpret the performance of vanilla label propagation.

3.3 Spatial Label Propagation

3.3.1 Data Preprocessing

First, we built the @-mention graph, which is a dictionary from users to the other dictionary that stores the users' neighbors and the number of bidirectional mentions. We ignored the dangle users in our constructed @-mention graph since they do not have any @-mentions from other users and do not mention others as well. Then, we built $data\_rows$ read from the raw text files. It has 6 columns in each row, which are "$user\_id$", "timestamp", "latitude", "longitude", "raw_text", "mentions". After that, we built $user\_to\_coordinates$, which is a dictionary from user id to the list of latitude and longitude. We plotted our dataset as a graph with 8, 9, and 10-mentions respectively as below.

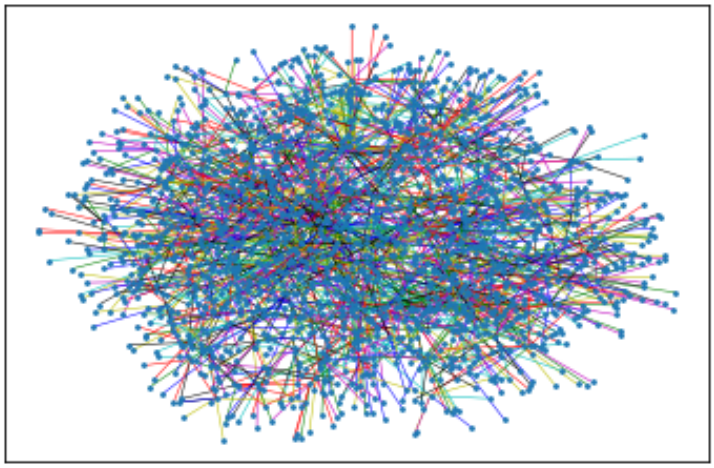


Figure 2: 8-mentions graph

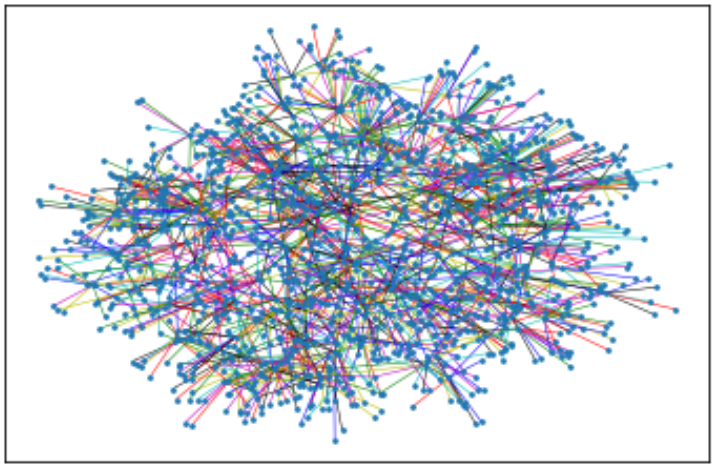


Figure 3: 9-mentions graph

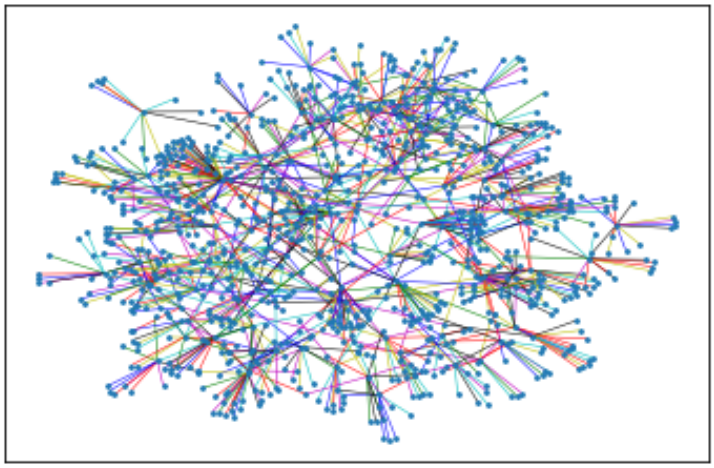


Figure 4: 10-mentions graph

3.3.2 Algorithm

Spatial label propagation is a semi-supervised, iterative approach for spatially inferring labels for items connected in a network. Selecting the closest member in an individual's social network can give significant indication of the individual's location. In this case, the actual location labels are known for only a small number of persons in the network, and these labels could be used as a source of ground truth information to estimate the geolocation of others. The geometric configurations of the neighbors can be used to select the current node's new label in each iteration of the procedure. We implement our spatial label propagation follow the pseudocode below:

The choice of the selection method is crucial for spatial label propagation. However, there are some problems that need to be solved when we consider the location inference. Firstly, when we have a network, it is unclear which neighbor should be selected, we need to figure out an efficient method which selects the neighbor wisely and gives us an accurate result. Secondly, the social network can be sparse, in which case there are lots of users who have only a few neighbors. In order to solve these problems, we decided to compare and contrast two location selection methods: geometric median and geometric mean.

Algorithm 2 Spatial Label Propagation

Data: U, L and N
Let E be the current mapping from user to location;
Initialize E with L;
while convergence criteria is not met **do**
    Let E' be the next mapping from user to location;
    for $u \in U -$ domain(L) **do**
        Let M be a list of locations;
        for $n \in N(u)$ **do**
            if $E(n) \notin \emptyset$ **then**
                add $E(n)$ to M;
            end if
        end for
        if $M \notin \emptyset$ **then**
            $E'(u) =$ select(M);
        end if
    end for
    E = E'
end while

3.3.3 Geometric median

For geometric median, we construct our set of located Twitter users by restricting it to those with at least five GPS-tagged tweets and these tweets should occur within a 15km geographic radius. Each user is then assigned a single location using the geometric median of their GPS locations and the distance is calculated using the formula as shown in formula [3]. To calculate the geometric median of a list of 2D coordinates, we iterate all the coordinates and we add the geodesic distance to all other coordinates. Then, the median of the coordinates will be the one who has the minimum geodesic distance. To make our algorithm more accurate, we introduce the weight and give more weight to more @-mention. We consider using the following equation to calculate the geometric median:

$$Y = argmin_{Y \in NY} \sum_{Y' \in NY} \text{great-circle-dist}(Y, Y') \tag{3}$$

For geometric median, the predicted coordinate of a node is always one of the true coordinates, not a distribution, so the total number of label configurations is fixed, by pigeonhole principle, it will always converge to one of a previous state after many iterations. We expected that the model converges to one single label configuration after many iterations, but in practice, it can oscillate between two configurations. Our geometric median algorithm follow the pseudocode below:

Algorithm 3 Geometric Median

Data: points
for $p \in points$ **do**
    for $p_2 \in points$ and $p_2 ! = p$ **do**
        if p_2 is weighted **then**
            Let n be the number of weight
            duplicate p_2 n times
        end if
        calculate the sum of distance of p2 to p
    end for
    get the minimum sum distance
end for
Result: the point with the minimum sum distance

3.3.4 Geometric mean

To calculate the geometric mean of a list of [latitude, longitude] on the Earth's surface,we decide to transform the position on the sphere to (x, y, z) in a 2D plane. As shown in formula [4], [5], [6]: [2]

$$x = \cos(\text{latitude}) * \cos(\text{longitude}) \tag{4}$$
$$y = \cos(\text{latitude}) * \sin(\text{longitude}) \tag{5}$$
$$z = \sin(\text{latitude}) \tag{6}$$

Specifically, we converted latitude and longitude, which are in radians, to Cartesian coordinates in 2D plane for each location. And then for each location point, we duplicate the location by its weight, which is the number of mentions. After that, we computed the average of all these Cartesian coordinates, and finally converted the average coordinates to latitude and longitude. To better visualize, we shown in the Figure 5.

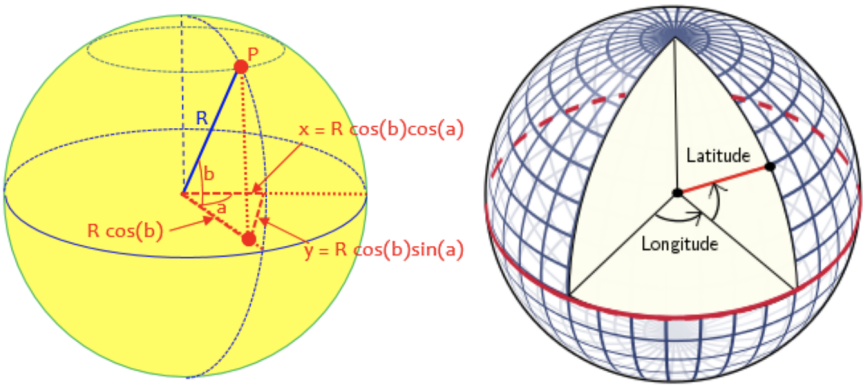


Figure 5: Convert between Cartesian Coordinates and latitude and longitude

Our geometric mean algorithm follow the pseudocode below:

Algorithm 4 Geometric Mean

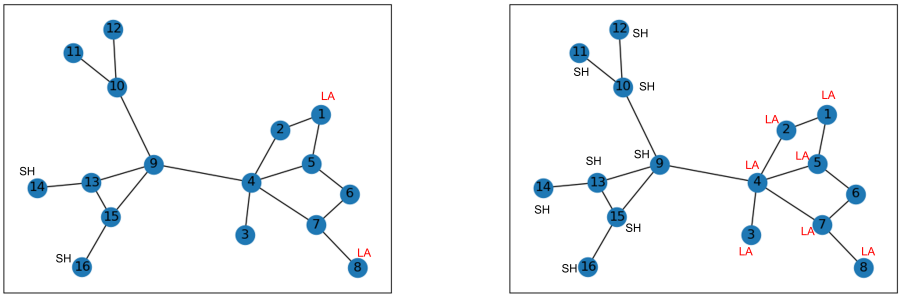
1: **Data**: Coordinates, Weight
2: Initialize x, y, z with 0.0, and initialize $points$ with the weighted coordinates
3: **for** p in $points$ **do**
4: Let lat and lot be the latitude and longitude of p
5: Calculate x, y, z by lat and lot
6: **end for**
7: Compute the means of x, y, z on 2D plane
8: Transform the means of x, y, z to the latitude and longitude
9: **Result**: mean of latitude and longitude

4 Evaluation

4.1 Geo-Classification

First, we run our vanilla-LPA and Explain-LPA model on small handcrafted network to verify the correctness of our implementation. We use the network in Figure 6a where nodes n1 and n8 are tagged LA and nodes n14 and n18 are tagged SH. We set uniform edge weights. After running for 86 iterations, Ex-LPA classify the nodes into two communities using the most-likely label. The decision boundary clearly cut through the edge (n4, n9) and is very intuitive as shown in Figure 6b. The computed feature importance matrix is included in Figure 1c which suggest n1 is most influential for classifying n2-n5 and n14 is most influential for classifying n13. We manually inspect the network and thinks this automatic explanation does make sense.

Next, We evaluate our LPA model the GEOTEXT datast. We randomly masked out different ratio of nodes as testing nodes and use the remaining nodes as seeds to run the propagation. We report the accuracy against train-test ratio in Table 1. As a result, we only achieved 7 to 8% accuracy when we masked out 60 percent of the ground truth nodes on the GEOTEXT dataset, as shown in the table below. The results surprised us at first. However, as we kept investigating further, we believed that the algorithm is working as intended and the problem lies in the assumption of vanilla LPA, as we justified using an argument on “Oracle LPA".



(a) Before LPA

(b) After LPA

Node	Y-pred	Contribution			
		n1	n8	n14	n16
n2	LA	0.84	0.16	0	0
n3	LA	0.67	0.33	0	0
n4	LA	0.67	0.33	0	0
n5	LA	0.75	0.25	0	0
n6	LA	0.42	0.58	0	0
n9	SH	0	0	0.5	0.5
n10	SH	0	0	0.5	0.5
n11	SH	0	0	0.5	0.5
n12	SH	0	0	0.5	0.5
n13	SH	0	0	0.63	0.34
n15	SH	0	0	0.34	0. 36

(c) Feature importance analysis

Figure 6: Performance of our Explain-LPA on handcrafted network.

Train/test ratio	Accuracy
Oracle	0.103
20%	0.0987
40%	0.0873
60%	0.0721
80%	0.0571

Table 1: Performance of our Explain-LPA on GeoText. Although the performance is minimal, it is not the fault of our implementation because the accuracy of oracle LPA is only around 10%!

Oracle Label Propagation Vanilla LPA makes a big assumption that the location label of a node can be estimated by looking at the neighbors of that node. In order to evaluate the validity of this assumption. We introduce the oracle LPA, which assume the ground truth labels of all neighbors are known and only propagate for one step. The algorithm is set up similarly as the vanilla LPA. We define Y, W and T the same as we did in vanilla LPA, except making one small adjustment to the weight matrix W. We fill the diagonal of W with zeros:

$$W_{ii} = 0 \quad \forall\, i$$

To enforce the assumption that the ground truth labels of all neighbors are known, Y will have all the ground truth label for all nodes. As the ideal version of label propagation, oracle LPA achieved only 10.3% accuracy on the whole GEOTEXT dataset. This results show that in the vanilla setting, estimating the discrete location label based on one's neighbor on the graph is not feasible on the GEOTEXT dataset. This is **NOT** contradictory to previous work, since previous work often use some variants of LPA such as Spatial LPA or incorporate LPA with text-based information to achieve state of the art result on GEOTEXT.

Analysis We believe that there are several reasons why formulating the geolocation prediction task as a classification problem and using vanilla label propagation to estimate the labels failed:

1. Label loss. Another assumption made by vanilla label propagation is that all the labels must presents in the seed nodes. There are over 9000 nodes and only over 2000 labels. When we mask out 60 percent of the labels, lots of the original labels cannot be found in the remaining nodes, which means we are bound to fail on lots of examples.

2. Label bias. When we convert the coordinate to location labels, the locations are not always city and some small places like Morrisville are also included in the dataset. As a result, the algorithm will tend to favor large city because there are lots of samples from there.

3. Failure to include "close enough" match. Prediction that is very close to user's location but does not match the user's label will be counted as failure.

4. Scale of dataset. The GEOTEXT dataset contains only 9475 users. As a result, each node in social network graph only has around 5 and in most cases under 10 neighbors. Estimating the label based on neighbors is hard on such a small dataset, where only few neighbors are available. However, vanilla label propagation does not scale well with large dataset. The time and space complexity of the algorithm is in $\mathcal{O}(n^2)$, where n is the number of users. Therefore, inference using vanilla label propagation is intractable on large dataset with millions of users.

4.2 Geo-Regression

We use three metric to evaluate our spatial label propagation model: 1) absolute distance 2) mean square error according to latitude and longitude, and 3) mean square error according to Cartesian coordinates. To calculate absolute distance between two coordinates, we used the distance function in geopy and compared the outcome with the threshold. We set the threshold to be 15km, which means it is a true prediction if the distance between the coordinate we calculated and the corresponding ground true coordinate is in [0, 15km].

To calculate mean square error according to latitude and longitude (2D), we used the formula [7] to calculate the error. We set the threshold to be 1, which means it is a true prediction if the error we calculated is smaller than 1.

Similarly, to calculate mean square error according to Cartesian coordinates x, y, z (3D), we first converted [latitude, longitude] to [x, y, z] by formula [4] [5] [6], and then used the formula [8] to calculate the mean square error in 3D. We set the threshold to be 0.01, which means it is a true prediction if the error we calculated is smaller than 0.01.

We first calculated the accuracy using 5 different train / test ratio with 5:5, 6:4, 7:3, 8:2, 9:1, respectively. When the max iteration is 5, the mean and median calculated by three different metrics: distance difference, MSE (Latitude, Longtitude)and MSE (Cartesian Coordinates) are shown in Figure 7 and 8, respectively. From Figure 7 and 8, we conclude that the selection method, median is better than mean. Therefore, we continue to run the median for more iteration, i.e. set max iteration to 6, as shown in Figure 9, which is better than the max iteration 5 case. We also fix the train / test ratio to be 8:2 and run the max iteration range from 1 to 10 for both mean and median selection methods with the three different metrics: distance difference, MSE (Latitude, Longtitude)and MSE (Cartesian Coordinates) are shown in Figure 10 and 11. In order to better visualize the relationship, we also plot the correspond line chart as shown in Figure 12 and 13.

Mean – 5 iterations

Train / Test Ratio	Distance Difference	MSE (Latitude, Longitude)	MSE (Cartesian Coordinates)
5:5	0.4002	0.5869	0.5394
6:4	0.4118	0.5937	0.5500
7:3	0.4187	0.6005	0.5580
8:2	0.4098	0.5962	0.5517
9:1	0.4373	0.6156	0.5694

Figure 7: Accuracy of our Spatial-LPA on GEOTEXT with Geometric Mean Select Method. Stop at 5 iteratons.

Median – 5 iterations

Train / Test Ratio	Distance Difference	MSE (Latitude, Longitude)	MSE (Cartesian Coordinates)
5:5	0.7469	0.8358	0.8242
6:4	0.7463	0.8360	0.8233
7:3	0.7451	0.8388	0.8274
8:2	0.7401	0.8263	0.8170
9:1	0.7556	0.8441	0.8362

Figure 8: Accuracy of our Spatial-LPA on GEOTEXT with Geometric Median Select Method. Stop at 5 iteratons

Median – 6 iterations

Train / Test Ratio	Distance Difference	MSE (Latitude, Longitude)	MSE (Cartesian Coordinates)
5:5	0.7848	0.8631	0.8522
6:4	0.7783	0.8601	0.8484
7:3	0.7696	0.8563	0.8449
8:2	0.7401	0.8263	0.8170
9:1	0.7649	0.8507	0.8428

Figure 9: Accuracy of our Spatial-LPA on GEOTEXT with Geometric Median Select Method. Stop at 6 iteratons

Mean – 8:2 train / test ratio

Iterations	Distance Difference	MSE (Latitude, Longitude)	MSE (Cartesian Coordinates)
1	0.4987	0.6691	0.6286
2	0.4556	0.6373	0.5942
3	0.4284	0.6127	0.5670
4	0.4244	0.6088	0.5656
5	0.4098	0.5962	0.5517
6	0.4158	0.5948	0.5530
7	0.4072	0.5855	0.5471
8	0.4131	0.5915	0.5491
9	0.4032	0.5869	0.5444
10	0.4098	0.5902	0.5491

Figure 10: Iteration-wise performance of our Spatial-LPA on GEOTEXT with Geometric Mean Select Method. As we can see, the algorithm basically converges after 6 iterations.

Median – 8:2 train / test ratio

Iterations	Distance Difference	MSE (Latitude, Longitude)	MSE (Cartesian Coordinates)
1	0.7374	0.8276	0.8176
2	0.7606	0.8395	0.8302
3	0.7401	0.8263	0.8170
4	0.7599	0.8395	0.8302
5	0.7401	0.8263	0.8170
6	0.7599	0.8395	0.8302
7	0.7599	0.8395	0.8302
8	0.7599	0.8395	0.8302
9	0.7599	0.8395	0.8302
10	0.7599	0.8395	0.8302

Figure 11: Iteration-wise performance of our Spatial-LPA with Geometric Median Select Method. As we can see, the algorithm basically converges after 6 iterations.

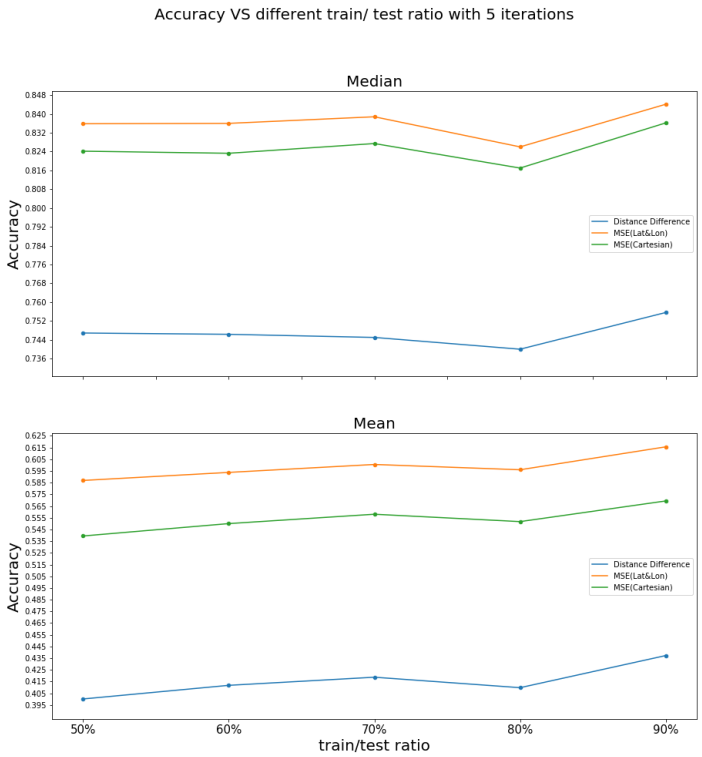


Figure 12: Accuracy of our Spatial-LPA over different train/test ratio. As we can see, the accuracy only slightly improves with more training nodes and quickly plateaued.Stop at 5 iterations.

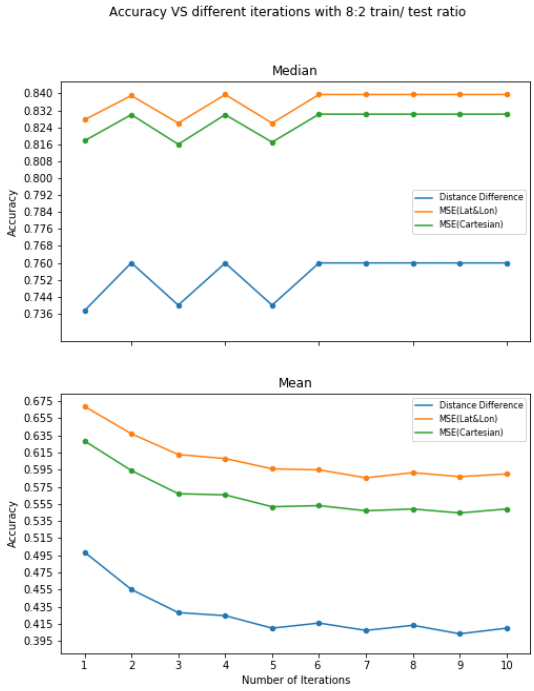


Figure 13: Iteration-wise performance of our Spatial-LPA with 8:2 train/test ratio. As we can see, the algorithm basically converges after 5 iterations.

From the results, we can conclude that ,spatial-LPA performs best under the metric of mean square error of [latitude, longitude], and the general accuracy of the geometric median is much better than that of geometric mean. Hence, we could infer that for the geometric median, it could represent an actual location of the user and avoid assigning a user a non-meaningful location from averaging locations. But we found that it may suffer from oscillations of output. To deal with this situation, we currently set the max iterations to be 5, but in the future we would figure out the reason. It is worth mentioning that although geometric means could calculate a more accurate location than the geometric median, the location it computes may not be valid. For example, it may give a location at the Pacific Ocean where no one would be there.

$$\text{error2D} = \sqrt{(\text{trueLattitude - predLattitude})^2 + (\text{trueLongitude - predLongitude})^2} \tag{7}$$

$$\text{error3D} = \sqrt{(\text{trueX - predX})^2 + (\text{trueY - predY})^2 + (\text{trueZ - predZ})^2} \tag{8}$$

Furthermore, we evaluated our method by changing the percentage of train/test nodes and number of max iterations respectively. For the percentage of train/test nodes, we used 70% train and 30% test, 80% train and 20% test (default), and 90% train and 10% test. For the number of max iterations, we used 5 (default), 8, 10. From the figure 7, we can see that for both selection methods, when max iteration is 5, 90% train and 10% test provides the highest accuracy and 80% train and 20% test provides the lowest accuracy in three evaluation methods. From the figure 8, we can see that for geometric mean, 5, 8, 10 iterations result in almost the same accuracy. For geometric median, 8 and 10 iterations overlap and they both provide the highest accuracy in three evaluation methods, which we can conclude that the accuracy will not change anymore after 8 iterations.

5 Discussion and Future Work

In this project, we explored the problem of explaining the decisions made by label propagation models. We proposed the *Explained label propagation*(Ex-LPA) algorithm which not only predicts the class of a new user but also compute a feature importance matrix for human to identify the most influential nodes in the network towards a particular. We have implemented and tested our Explained LPA on the Geo-classification setting using both handcrafted examples and GEOTEXT dataset. For handcrafted examples, we manually examine the computed feature importance matrix and believe it does makes sense, as shown in section 4.1.

For Geo-regression setting, we need to generalize our Ex-LPA for Spatial label propagation. We believe this is possible by taking the first-order approximation of the *select method* but we don't have enough time to implement this idea. Instead, we report the performance of our basic Spatial LPA which is comparable to the state of art, as shown in section 4.2.

In the future, we would like to extend our work on explained-LPA to other version of label propagation algorithm, especially when the update rule is nonlinear such as Spatial-LPA. We may also adapt our approach to the Modified Adsorption version of LPA proposed by [7]. Besides social network information, recent geolocation prediction models often integrate some text-based information into LPA to further boost the performance, which makes the problem of interpretability much more challenging.

6 Roles of team members

Haiying Huang propose and implement Explained Label Propagation. Also implement Spatial Label Propagation together with Yu, Yuhan, Zuer and run the test cases with different train / test ratio and num of iterations.

Cheng Lu data preprocessing, implementation and evaluation of vanilla label propagation.

Yu Hou together with Zuer, Yuhan, and Haiying are responsible to the spatial label propagation, structure and implement the data visualization.

Zuer Wang together with Yu, Yuhan, and Haiying are responsible to the spatial label propagation, run and evaluate the test cases with different train / test ratio and num of iterations.

Yuhan Shao together with Yu, Zuer, and Haiying are responsible to the spatial label propagation, run and evaluate the test cases with different train / test ratio and num of iterations.

7 Link to Github Folder

https://github.com/arthurhaiying/CS245-Geolocation-Predictor.git

References

[1] Yin-Wen Chang and Chih-Jen Lin. "Feature ranking using linear SVM". In: *Causation and prediction challenge*. PMLR. 2008, pp. 53–64.

[2] George Gerdan and Rod Deakin. "Transforming cartesian coordinates X, Y, Z to geographical coordinates ?, ?, h". In: *Australian Surveyor* 44 (June 1999). DOI: 10.1080/00050326.1999.10441904.

[3] David Jurgens. "That's What Friends Are For: Inferring Location in Online Social Media Platforms Based on Social Relationships". In: *Proceedings of the International AAAI Conference on Web and Social Media* 7 (Aug. 2013), pp. 273–282. URL: https://ojs.aaai.org/index.php/ICWSM/article/view/14399.

[4] David Jurgens. "That's What Friends Are For: Inferring Location in Online Social Media Platforms Based on Social Relationships". In: *Proceedings of the International AAAI Conference on Web and Social Media* 7 (Aug. 2021), pp. 273–282. URL: https://ojs.aaai.org/index.php/ICWSM/article/view/14399.

[5] David Jurgens et al. "Geolocation prediction in twitter using social networks: A critical analysis and review of current practice". In: *Ninth international AAAI conference on web and social media*. 2015.

[6] Bjoern H Menze et al. "A comparison of random forest and its Gini importance with standard chemometric methods for the feature selection and classification of spectral data". In: *BMC bioinformatics* 10.1 (2009), pp. 1–16.

[7] Afshin Rahimi, Trevor Cohn, and Timothy Baldwin. "Twitter User Geolocation Using a Unified Text and Network Prediction Model". In: *CoRR* abs/1506.08259 (2015). URL: http://arxiv.org/abs/1506.08259.

[8] Zifan Wang et al. "Towards frequency-based explanation for robust cnn". In: *arXiv preprint arXiv:2005.03141* (2020).

[9] Xiaojin Zhu and Zoubin Ghahramanirh. "Learning from labeled and unlabeled data with label propagation". In: (2002).