TweetHood: Agglomerative Clustering on Fuzzy *k*-Closest Friends with Variable Depth for Location Mining

Satyen Abrol[1], Latifur Khan[2]

Department of Computer Science, University of Texas at Dallas
800 W Campbell Road, Richardson, Texas, USA 75083
[1]satyen.abrol@student.utdallas.edu
[2]lkhan.utdallas.edu

Abstract— **According to a recent report by research firm ABI Research, location-based social networks could reach revenues as high as $13.3 billion by 2014 [1]. Social Networks like Foursquare and Gowalla are in a dead heat in the Location War. But, having said that it is important to understand for privacy and security reasons, most of the people on social networking sites like *Twitter* are unwilling to specify their locations explicitly. This creates a need for software that mines the location of the user based on the implicit attributes associated with him. In this paper, we propose the development of a tool TweetHood that predicts the location of the user on the basis of his social network. We show the evolution of the algorithm, highlighting the drawbacks of the different approaches and our methodology to overcome them. We perform extensive experiments to show the validity of our system in terms of both accuracy and running time. The experiments performed demonstrate that our system achieves an accuracy of 72.1% at the city level and 80.1% at the country level. Experimental results show that TweetHood outperforms the gazetteer based geo-tagging approach.**

Keywords— **Twitter, Location based Services, Agglomerative Clustering, and Gazetteer**

I. INTRODUCTION

Since its inception in the mid 90s, social networks have provided for a way for users to interact, reflecting of social networks or social relations among people, e.g., who share interests and/or activities. Initially, critics regarded social media as a fad, a temporary fashion. But as of 2010, social media has demonstrated exponential growth, making it the most popular activity on the internet. Twitter has recorded a 1,500 percent increase in the number of registered users, and over 70,000 applications have been developed according to the company. From 500,000 messages per quarter in 2007, the company has grown to 4 billion messages per quarter in 2010. As the increase in popularity of social networking is on a constant rise, new uses for the technology are constantly being observed. At the forefront of emerging trends in social networking sites is the concept of "real time" and "location based". So what makes location based social media services so important? We briefly describe three situations where knowing the location of a user has a direct impact on him.

Privacy and Safety: Gowalla is a location-based social networking website which allows users to 'check-in' at places in their local vicinity, either through a dedicated mobile application or through the mobile website. As a reward users will sometimes receive items from check-ins. This is achieved through the use of dedicated applications available on smartphones or by logging on to Gowalla's mobile website.

Posting such an update, publishes your current location to the user, and can prove to be an invitation to get your house robbed. In addition to this, knowing the location of the user makes it easier for spammers to attack the user in a more personalized manner.

Trustworthiness: Social media may be a source but is it a reliable one? Time in partnership with CNN discuss the impact of *Twitter* on the coverage of developments after Iran elections [3]. On June 12th, Iran held its presidential elections between incumbent Ahmadinejad and rival Mousavi. The result, a landslide for Ahmadinejad, has led to violent riots across Iran, charges of voting fraud, and protests worldwide. Even as the government of that country was evidently restricting access to opposition websites and text-messaging, but on *Twitter*, a separate uprising took place, as tweets marked with the hash tag *#cnnfail* began tearing into the cable-news network for devoting too few resources to the controversy in Iran. US State Department officials reached out to *Twitter* and asked them to delay a network upgrade that was scheduled for Monday (June 15th) night. This was done to protect the interests of Iranians and assess the sentiment of the public about the Iran elections. In such cases, it was important for the US State Department to be able to trust the location of the user.

Another major area where trustworthiness of the user location is important is when companies use social media to obtain location based opinion of products. Companies all over the world, from Burger King to Ford Fiesta, are using the social media to target customers [2].

Advertising and Marketing: Here, we talk about two broad aspects on how location based social media can affect companies in doing business in a better manner. Social networks connect people at low cost; this can be beneficial for entrepreneurs and small businesses looking to expand their contact bases. These networks often act as a customer relationship management tool for companies selling products and services. Companies can also use social networks for

978-0-7695-4211-9/10 $26.00 © 2010 IEEE
DOI 10.1109/SocialCom.2010.30

advertising in the form of banners and text ads. Since businesses operate globally, social networks can make it easier to keep in touch with contacts around the world.

Having highlighted the importance of location of the user in social media, it is important to understand that it is not provided explicitly by the users for a number of reasons. Some of the users are concerned about their privacy and security, others do not find any incentive in sharing the location. The end result is the impeded growth of location based services in the present world scenario.

The social networking companies have shifted their focus from building relationships to identifying temporal and spatial patterns in messages. Twitter has very recently introduced a new feature whereby users can find location based trending topics as shown in Fig 1. But this is still not sufficient as firstly, it only covers the users who mention their locations explicitly and secondly, the topic of search is limited to the trending topics.

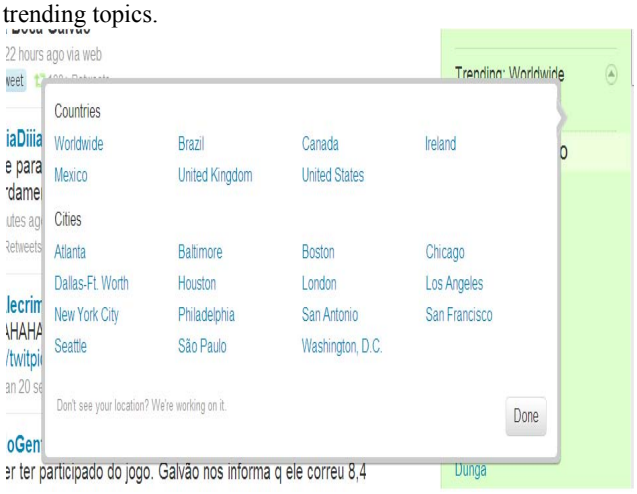

Fig. 1 The new feature by Twitter to provide location based trending topics

But to mine the location of the user is not an easy task in itself. As we shall later show, traditional text based location extraction techniques do not perform well in the domain of social networks. The presence of multiple locations mentioned in the text makes it difficult to identify a single location which serves the purpose of the page focus for the messages. Additionally, we observe a lack of relationship between the location of the user and the location mentioned in the text. Secondly, the high running time associated with such mining techniques makes them unusable in practical scenarios.

In this paper we address the identification of the location of the user on social networking sites based on the location of his closest friends. We demonstrate the development of TweetHood, a tool that outperforms the typical gazetteer based approach in both accuracy and running time.

The research paper is organized as follows. Section 2 surveys the related work in this domain and points out the novelty in our approach. Section 3 discusses the challenges faced in identifying the location of the user. Section 4 talks about location identification from text. Section 5 describes the various algorithms proposed by us in tackling the problem. Section 6 discusses the experiments and the observations on various approaches and their time complexities. We conclude in section 6, by giving a few pointers for the future work.

II. SURVEY OF PREVIOUS WORK

Substantial efforts have been spent on location identification and geo-tagging of documents and web pages. Social Networking on the other hand is still a very new field of computer science and little or no previous work has been done towards identifying the location of users based on their social activity. In this section we do a brief survey of the previous works on geo-tagging documents based on the content.

Most of the research can be broadly classified into two approaches. One, involving the concepts of Natural Language Processing and the other using data mining approach. Most of the work done using NLP techniques consists of input text that is structured and well edited. Li et al. [6] combined these two methodologies and used typical 5-step approach, first short listing the keywords appearing in the gazetteer and then applying NLP techniques to remove non geo terms. A precision of 93.8% was reported using their approach.

Mehler et al. [8] developed a model for estimating and evaluating spatial significance of entities using NLP techniques. Liu et al. [9] do a similar geo-analysis of the impact of the location of the source on the viewpoint presented in the news articles. Sheng et al. in [10] discussed the need for reordering the search results (like food, sports, etc.) based on user references obtained by analysing user's location.

Amitay *et al.* [7] present a way of determining the page focus of web pages using the gazetteer approach after using techniques to prune the data. They are able to correctly tag individual name place occurrences 80% of the time and are able to recognize the correct focus of the pages 91% of the time [7]. But they have a low accuracy for the geo/non-geo disambiguation.

It will be intuitive to think that location can be obtained from IP address of the user. But private information like IP address is not accessible to common users.

It is vital to understand here that identifying the location mentioned in documents or messages is very different from identifying the location of user from the messages posted by him. That is, even if page focus of the messages is identified correctly, that may not be the correct location of the user. E.g. People express their opinions on political issues around the world all the time. The recent catastrophic earthquake in Haiti, led to many messages having references to the country. Another example is that of the volcano in Iceland that led to flights being cancelled all around the world. In addition to this, the time complexity of text based geo-tagging messages is very large making it unsuitable for real time applications like opinion mining. Thus, as we as shall show in our experiment section, the geo-tagging of user messages is an inaccurate method to identify the location of the user.

Tweethood makes an important contribution in this field. The tool identifies the location of the user based on his social

network using a highly efficient algorithm that ensures high accuracy and low time complexity.

III. CHALLENGES IN LOCATION EXTRACTION

As discussed previously, a lot of efforts are being made on the part of the social networking companies to incorporate location information in the communication. Twitter very recently acquired Mixer Labs, a maker of geolocation Web Services, to boost up its location based services campaign and compete with the geo savvy mobile social networking sites like Foursquare and Gowalla.

But still, the efforts are not paying dividends simply because of several security and privacy reasons. And there is no incentive for users. We conducted an experiment and found that out of 1 million users on Twitter; only 14.3% actually share their location explicitly. Hence explicitly mentioned locations are rare and in certain cases untrustworthy. That leaves us with the question; can the location be mined from implicit information associated with the users like the content of messages posted by them and nature of their social media network?

A. Location Extraction from Raw Messages

Twitter, being a popular social media site, is a way by which users generally express their opinions, with frequent references to locations including cities, countries etc. It is also intuitive in such cases to draw a relation between such locations mentioned in the *tweets* and the place of residence of the user. In other words a message from a user supporting the Longhorns (Football team for University of Texas at Austin) is most likely from a person living in Austin, Texas, USA then from someone in Australia.

But as it is evident, this approach shall prove to be inaccurate in cases where the user talks about news making incidents in other parts of the world. E.g. Haiti was a popular geo-reference in *tweets* after the earthquake. In another case, someone who talks about going to Venice for a vacation is not necessarily Italian.

B. Location Extraction from Social Network of User

Another approach that can be employed makes use of the social network of the user. Here, the social network of the user comprises of followers and friends (or people he is following). This approach gives us an insight on a user's close friends and the celebrities he is following. Intuitively, most of a person's friends are from same country and also, a person is more likely to follow celebrities that are from his own country. In other words, an American's friends are mostly Americans and he has a higher probability of following President Barack Obama than Asian users.

This approach also has certain drawbacks. At the country level, it is not always safe to assume that a person always follows celebrities from his own country. Queen Rania of Jordan advocates for global education and thus has followers around the world. In such cases, judging the location of a user based on the celebrities he is following can lead to inaccurate results.

IV. LOCATION MINING FROM TEXT

In this section we discuss the various approaches, in detail, that can be taken to identify the location of a user on *Twitter*. It is important to understand here that a location concept is typically of the format {City} A/ {State} B/ {Country} C. And for each location depending on the level of detail, either of A, B or/and C can be null.

To determine the location from mining the messages, we devise a score based identification and disambiguation method Location_Identification. Before running the actual algorithm, we perform preprocessing of data, which involves removal of all those words from the messages that are not references to geographic locations. For this, we use the CRF Tagger, which is an open source Part of Speech (POS) tagger for English with an accuracy of close to 97% and a tagging speed of 500 sentences per second [4]. The CRF tagger identifies all the proper nouns from the text and term them as keywords $\{K_1, K_2,\ldots, K_n\}$. In the next step, the TIGER (**T**opologically **I**ntegrated **G**eographic **E**ncoding and **R**eferencing system) [5] dataset is searched for identifying the city names from amongst them. The TIGER dataset is an open source gazetteer consisting of topological records and shape files with coordinates for cities, counties, zip codes, street segments, etc. for the entire US.

Algorithm 1 Location_Identification (User_Messages)

Input: UM: *All Messages of User*
Output: Vector (C, S): Concepts and Score vector

1: for each keyword, K_i //Phase 1
2: for each $C_j \in K_i$ //C_j - Street Concept
3: for each $T_f \in C_j$
4: type = Type (T_f)
5: If (T_f occurs in UM) then $S_{Cj} = S_{Cj} + S_{type}$
6: for each K_i //Phase 2
7: for each $C_j \in K_i$
8: for $T_f \in C_j$, $T_s \in C_L$
9: If ($T_f = T_s$) and ($C_j \neq C_L$) then
10: type = Type (T_f)
11: $S_{Sj} = S_{Cj} + S_{type}$
12: return (C, S)

Algorithm 1 describes the gazetteer based algorithm. We search the TIGER gazetteer for the concepts $\{C_1, C_2\ldots.C_n\}$ pertaining to each keyword. Now our goal for each keyword would be to pick out the right concept amongst the list, in other words disambiguate the location. For this, we use a weight based disambiguation method. In the phase 1, we assign the weight to each concept based on the occurrence of its terms in the text. Specific concepts are assigned a greater weight as compared to the more general ones. In phase 2, we check for correlation between concepts, in which one concept subsumes the other. In that case the more specific concept gets the boosting from the more general concept. If a more specific

concept C_i is part of another C_j then the weight of C_j is added to that of C_i.

E.g. City carries 15 points, state 10 and a country name carries 5 points. For the keyword "Dallas", consider the concept of {City} Dallas/ {State} Texas/ {Country} USA. The concept gets 15 points because Dallas is a city name, and it gets an additional 10 points if Texas is also mentioned in the text. In phase 2, we consider the relation between two keywords. Considering the previous example, if {Dallas, Texas} are the keywords appearing in the text, then amongst the various concepts listed for "Dallas" would be {City} Dallas/{State} Texas/{Country} USA and one of the concepts for "Texas" would be {State} Texas/ {Country} USA. Now, in phase 2 we check for such correlated concepts, in which one concept subsumes the other. In that case the more specific concept gets the boosting from the more general concept. Here, the above mentioned Texas concept boosts up the more specific Dallas concept. After the two phases our complete we re-order the concepts in descending order of their weights. Next, each concept is assigned a probability depending on their individual weights.

V. Graph Related Methods

Graph related approaches are the methods that rely on the social graph of the user while deciding on the location of the user. In this section we describe four such methods that show the evolution of the algorithm currently used in TweetHood. Figure 2 shows an undirected graph with a depth d=2 used to represent the social network of a user. Each node in the graph represents a user and an edge represents friendship. The root represents the user U whose location is to be determined, and the F_1, F_2,…, F_n represents the *n* friends of the user. Each friend can himself his own network, like F_2 has a network comprising of *m* friends F_{21}, F_{22},…., F_{2m}.

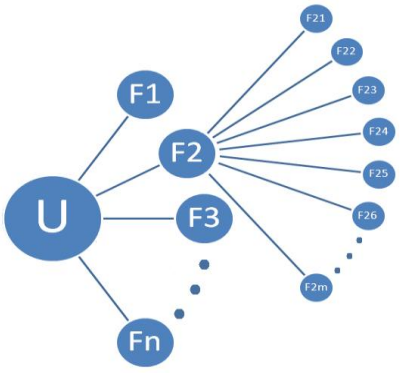

Fig. 2 An undirected graph for a user U showing his friends

A. Simple Majority with Variable Depth

A naïve approach for solving the location identification problem would be to take simple majority on the locations of friends (followers and following) and assign it as the label of the user. Since a majority of friends will not contain a location explicitly, we can go further into exploring the social network of the friend (friend of a friend). For example, in Figure 1, if the location of Friend F_2 is not known, instead of labelling it as *null*, we can go one step further and use F_2's friends in choosing the label for it. It is important to note here that each node in the graph will have just one label (single location) here.

Algorithm 2 Simple_Majority (userId, depth)

Input: User Id of the User and the current depth
Output: Location of the User

```
1: If (Twitter_Location (userId)! =null)
2:         then return Twitter_Location (userId);
3: else If (depth=0)
4:         then return null;
5: else {
6:         All_Friends [] = Get_Friends (userId);
7:         for each friend All_Friends[i]
8:           Location[i] = Simple_Majority (All_Friends[i],
           depth-1);
9:         Aggregate (Location [ ]);
10:        Boost (Location [ ]);
12:        return Max_Location (Location [ ]);
12:    }
```

The algorithm Simple_Majority (userId, depth) is divided into several steps as shown in algorithm 2. In steps 1 and 2, we check for the explicitly specified location, and if it is present, the node is given that label. At Step 3, if the algorithm on being called recursively has reached a depth of 0 and is unable to find a location, the algorithm returns *null* to the calling method. It is important to note here that the above two conditions specify the boundary conditions of the recursive function. If either of the two conditions is not met, then we go try to find the try to determine the location on the basis of the simple majority of the labels of the friends. In Step 6, we collect the list of all friends of the user. Next, for each of the friends we determine the location by recursively calling Simple_Majority with the friend's user id and decreasing the depth by 1. Once, we have the locations for all the friends, in step 6 we perform aggregation of the locations to obtain unique locations. Next, we perform the boosting of the concepts in which a more specific concept is boosted by a more general concept. That is, the state concepts boost all the city concepts in which the city belongs to that state. Similarly, the country level concepts boost the state and city level concepts. Finally, the algorithm chooses the one with the maximum frequency and assigns it to the node.

B. k- Closest Friends with Variable Depth

As *Twitter* has a high majority of users with public profiles, a user has little control over the people following him. In such cases, considering spammers, marketing agencies, etc. while deciding on the user's location can lead to inaccurate results. Additionally, it is necessary to distinguish the influence of each friend while deciding the final location. We further

modify this approach and just consider the *k closest* friends of the user.

Algorithm 3 Closeness (userId, friendId)

Input: User Id of the User and User Id of the Friend
Output: CF, the *Closeness* between the user and the friend

1: CF=0; //initialize
2: All_Friends1 [ ] = Get_Friends (userId);
3: All_Friends2 [ ] = Get_Friends (friendId);
4: CF = Common_Friends (All_Friends1 [ ], All_Friends2 [ ]);
5: If SR > $N_{spammer}$ // spammer
6: then CF = 0;
7: If (Followers (friendId) > $N_{celebrity}$) then

8: $CF = CF * \frac{|All\_Friends1|}{Followers(friendID)}$
9 return CF;

Before we explain *k*_Closest_Friends () algorithm, let us define *closeness* amongst users. *Closeness* amongst two people is a subjective term and we can implement it in several ways including number of common friends, semantic relatedness between the activities (verbs) of the two users collected from the messages posted by each one of them, etc. Based on the experiments we conducted, we adopted the number common friends as the optimum choice because of the low time complexity and better accuracy. Algorithm 3 illustrates the detailed explanation of the *closeness* algorithm. The algorithm takes as input the ids of the user and the friend and returns the *closeness* measure. In steps 2 and 3, we calculate obtain the ids of the friends of both the user and his friend. Next, we calculate their common friends and assign it as CF. But, there are certain cases we need to take care of spammers and celebrities. The algorithm has zero tolerance towards spammers. A spammer is typically identified by the vast difference between the number of users he is following and the number of users following him back. We define the Spam Ratio of a friend as

$$SR(friendId) = \frac{Following(friendId)}{Followers(friendId)} \quad (1)$$

And if SR is found to be greater than a threshold $N_{spammer}$ we identify the friend as a spammer and set CF as 0. Finally, we would like to control the influence of celebrities in deciding the location of the user because of previously mentioned problems. But, it is also important to note here that in certain cases the celebrities he is following are our best bet in guessing the user's location. In step 6 we abbreviate the *closeness* effect a celebrity has on a user.

Algorithm 4 shows the steps involved in the *k*-_Closest_Friends (userid, depth). Steps 1 through 7 remain the same as that of the Simple_Majority (userid, depth). Next, we call the method *k*-CF (userid, AllFriends [], k). The method returns an array consisting of userids of *k* closest friends of the user along with their pair wise *closeness* to the user as described in Algorithm 3. In the next step, for each of the *k* closest friends, we determine the location by recursively calling *k*_Closest_Friends () with the friend's user id and decreasing the depth by 1. Once, we have all locations of *k* closest friends, supported by their individual *closeness* as specified by Algorithm 3 we aggregate and boost the scores of the concepts and the concept with the maximum weight is returned.

Algorithm 4 k_Closest_Friends (userId, depth)

Input: User Id of the User and the current depth
Output: Location of the User

1: If (Twitter_Location (userId)! =*null*)
2: then return Twitter_Location (userId);
3: else If (depth=0)
4: then return *null*;
5: else {
6: All_Friends [] = Get_Friends (userId);
7: k_CloseFriends [ ] [2] = *k*-CF (userId, Friends [ ], k);
8: for each friend k_CloseFriends[i] [ ]
9: Location[i] [1] = k_Closest_Friends (k_CloseFriends [i], depth-1);
10: Location[i] [2] = k_CloseFriends [i] [2];
11: Aggregate (Location [ ] [ ])
12: Boost (Location [ ] [ ]);
13: return Max_Location (Location [ ] [ ]);
14: }

C. Fuzzy k- Closest Friends with Variable Depth

As mentioned previously, in Simple_Majority () and *k*_Closest_Friends (), each node in the social graph has a single label, and at each step, the locations with lower probabilities are not propagated to the upper levels of the graph. The disadvantage of this approach is that firstly, it tells us nothing about the confidence of the location identification of each node and secondly, for instances where there are two or more concepts with similar score, only the location with highest weight is picked up, while the rest are discarded. This leads to higher error rates.

The idea behind the Fuzzy *k* closest friends with variable depth is the fact that each node of the social graph is assigned multiple locations each associated with a certain probability. And these labels get propagated throughout the social network, no locations are discarded whatsoever. At each level of depth of the graph the results are aggregated and boosted similar to the previous approaches so as to maintain a single vector of locations with their probabilities.

Algorithm 5 Fuzzy_k_Closest_Friends (userId, depth)

Input: User Id of the User and the current depth
Output: Location Vector of the User
1: If (Twitter_Location (userId)! =*null*)

```
2:        then return [Twitter_Location (userId), 1.0];
3: else If (depth=0)
        then return [null, 1.0];
4: else {
5:        All_Friends [] = Get_Friends (userId);
6:         k_CloseFriends [ ] [2] = k-CF (userId, All_Friends [
          ], k);
7:        for each friend k_CloseFriends[i] [ ]
8:            Location[i] [1] = k_Closest_Friends
          (k_CloseFriends [i], depth-1);
9:            Location[i] [2] = k_CloseFriends [i] [2];
10:       Aggregate (Location [ ] [ ])
11:       Boost (Location [ ] [ ]);
12:       return Max_Location (Final_Location [ ]);
13:    }
```

Algorithm 5 Fuzzy *k*-Closest Friends approach to determine location

Algorithm 5 shows the steps involved in the algorithm. The initial input to the algorithm is the userid of the user and the maximum depth. In step 1, at any depth of recursion, the algorithm tries to determine the explicitly specified location. If the location is mentioned explicitly then it is return with confidence 1.0. Otherwise on reaching a depth of 0, if the algorithm is not able to find the location it returns *null* with a confidence 1.0. If the location is not mentioned explicitly then the algorithm tries to determine it on the basis of the locations of the *k*-Closed Friends of the user. In step 5 we collect the list of all the friends of the user comprising of the people he is following and the people following him. Next, we call the method *k*-CF (userid, AllFriends [], k) described in the *k*-_Closest_Friends () algorithm. In the next step, for each of the *k* closest friends, we determine the list of locations, each associated with a probability, by recursively calling *k*-_Closest_Friends () with the friend's user id and decreasing the depth by 1. Once, we have all locations-probability distribution of *k-closest* friends, supported by their individual *closeness* as specified by Algorithm 3, we aggregate and boost the scores of the concepts as discussed in Simple_Majority () algorithm. The method finally returns a vector of location concepts with individual probabilities.

D. Agglomerative Hierarchical Clustering

As mentioned previously, at each step of the depth of the recursion, the function returns a vector of location concepts with their individual probabilities. Amongst the different concepts obtained from different concepts of the social graph, there is a *null* concept also, which originates when even at 0 depths we are not able to find a location. It is also important to understand that higher the *maxDepth* (the maximum depth), lesser number of nodes in the upper hierarchy are tagged *null* and hence lower is the score of the *null* concept in the final vector.

At this point we introduce something called the Location Confidence Threshold (LCT). The idea behind LCT is to ensure that when the algorithm reports the possible locations, it does so with some minimum level of confidence.

$$LCT(u, maxDepth) = (1 - \beta(u)^{maxDepth}) \quad (2)$$

As evident, the LCT increases with the increasing value of *maxDepth*, and reaches 1 for higher values of *maxDepth*. This also speaks to our intuition as with increasing depth, the probability of *null* concept also should decrease and we should be more confident in our result. β is a constant whose value lies between 0 and 1 and depends on the social graph of the user. For example, higher the number of labelled immediate friends of the user, lower is the value for β.

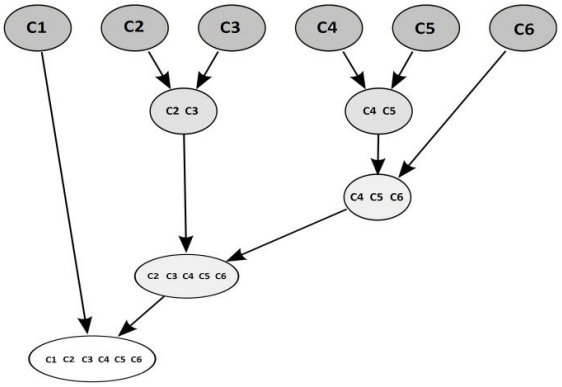

Fig. 3 Illustarion to show the agglomerative hierachial clustering

Till this point we have given little emphasis on the geospatial proximity of the different concepts. That is, we were treating the concepts purely as labels, with no mutual relatedness. Since the concepts are actual geographical cities, we agglomerate the closely located cities and suburbs in an effort to improve the confidence and thus the accuracy of the system. Fig 3 shows the agglomerative hierarchical clustering algorithm. Consider we have *p* concepts $C_1,\ldots,$ Cp each associated with its respective probability.

Initially, we have all concepts present individually as $\{C_1\}$, $\{C_2\},\ldots,$ $\{C_p\}$. If any non-*null* concept has a value greater than the LCT, then the program returns that concept as the location and terminates. Otherwise, at the next step we construct a matrix in which the number in the *i*-th row *j*-th column is an objective function Θ of the distances and cumulative scores between the *i*-th and *j*-th concepts.

$$\Theta_{i,j} = e^{S/T} * d \quad (3)$$

where $S = S_i + S_j$, the combined score of concept clusters C_i and C_j, d is the geographic distance between the two clusters and T is a constant with $0<T<1$.

At the first step of agglomeration, we combine two concepts with highest value of the objective function, Θ and check if the new concept cluster has a combined score greater than the LCT. If not, then we continue the process, constructing the matrix again, but this time some of the concepts are replaced by concept clusters. And we proceed to choose the two concepts clusters that have the maximum value of the objective function Θ. The mean geographic distance between

the a concept cluster A_i and a concept cluster B_j can be defined as

$$d_{AB} = \frac{1}{|A||B|} \sum_{x \varepsilon A} \sum_{x \varepsilon B} d(x, y) \quad (4)$$

Thus at each step of the agglomeration, we choose the two concept clusters with maximum value of the objective function Θ. If the score of the combined bag of concepts crosses the JCT, we return the bag of concepts as the possible location vector and terminate.

VI. EXPERIMENTS AND RESULTS

In this section, we evaluate the quality of the algorithms mentioned in the previous sections and describe how TweetHood outperforms the other approaches.

A. Data

For the experiments, we randomly choose 1000 users from all different countries and cities who explicitly mention their location. But to the algorithms, we do not mention the same. It is important to note here, for uniformity, we ensure that each has at least 50 friends so that 50- closest friends approach can be applied.

Secondly, all processes are run offline i.e. we store all the relevant information about the user like location, friend count, friends ids, etc. on the local machine and then run the algorithm. Hence the entire process is done offline, barring the geocoding process, which is used to convert the explicitly mentioned locations to a standard format.

B. Evaluation Method

Our evaluation is designed with the following goals in mind. First, we aim to compare the accuracy of different approaches both at the city as well as the country level and show the effectiveness of TweetHood. Secondly, we want to show the tradeoff between accuracy and time as a function of depth. Finally, we wish to show the affect the choice of number of closest friends (k) has on accuracy and time. For all experiments we choose the gazetteer based approach discussed in the previous sections as the baseline.

C. Experiment Type 1: Accuracy vs Depth

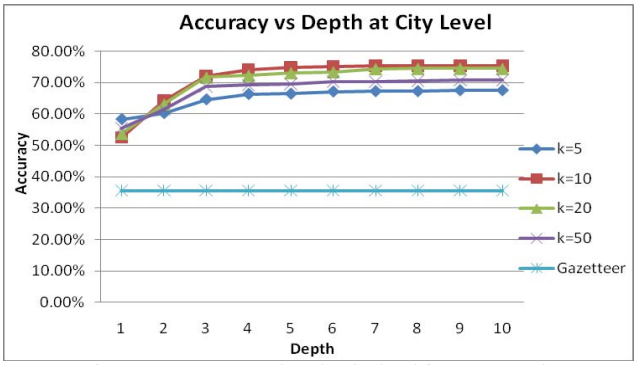

Fig. 4 Accuracy vs Depth at the city level for TweetHood

Fig. 4 shows the accuracy as a function of the depth for the city level location identification for the Agglomerative clustering on Fuzzy *k* closest Friends. We make two key observations, firstly, with the increasing depth the accuracy increases monotonically. This is obvious because, for *null* nodes we are willing to go further and thus eventually find a label. But, the accuracy doesn't increase significantly after depth =3. Secondly, for a major portion choosing k=10 gives us the highest accuracy as compared to the other values of k. The baseline gazetteer based approach has a fairly low accuracy of 35.6% compared to our approach.

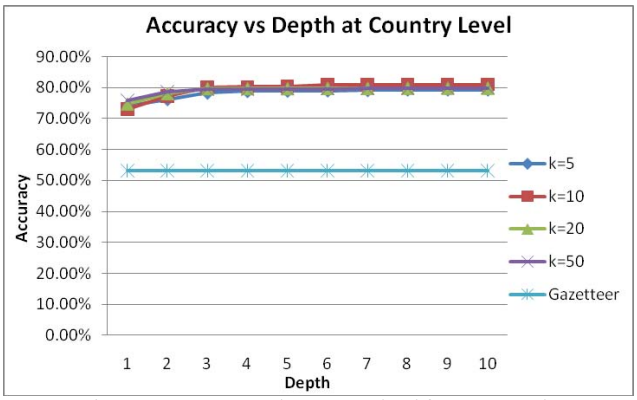

Fig. 5 Accuracy vs Depth at country level for TweetHood

Next, we study the effect of increasing the depth on country level location identification for the Agglomerative clustering on Fuzzy *k* closest Friends. The observations are very similar to the city level identification i.e. for depth greater than 3 the accuracy saturates. But here, the choice of k does not affect the accuracy as significantly as it did for the city level identification. And understandably, the accuracy for country level is higher than for the city level, and at k=10, depth=3 it is found to be 80.1%.

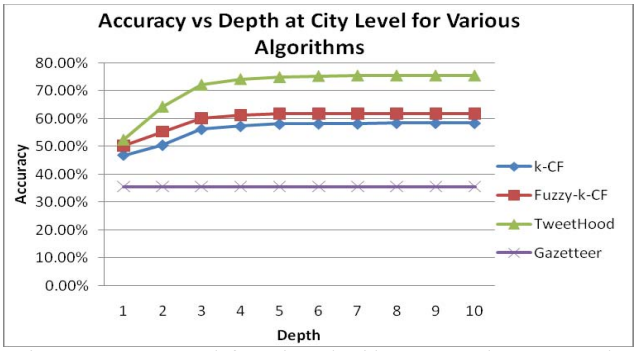

Fig. 6 Accuracy vs Depth for various algorithms compared to TweetHood

Fig 6 shows the comparison of various algorithms proposed by us on the city level location identification. It is important to note here that on the basis of previous experiments we conclude that k=10 is the optimal value for the future experiments. The key observations to make here are that the introduction of agglomeration of concepts actually brings about a great improvement in the accuracy because in some

cases just choosing the maximum value does not produce the correct result, the proximity of concepts and the threshold have to be taken into account.

D. Experiment Type 2: Time Complexity

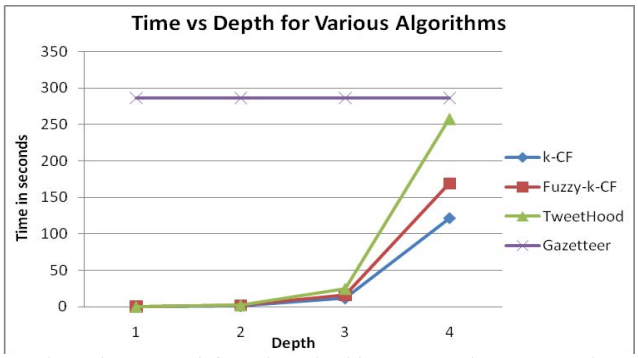

Fig 7. Time vs Depth for various algorithms compared to TweetHood

Now, we discuss the average running time for determination of the location of a single user. First, we compare the execution time for the various algorithms as a function of depth. As expected, the time increases exponentially with increasing depth. The other observation we make is that the time complexity increases as we go from k-closest friends to fuzzy k-closest friends to agglomerative fuzzy k-closest friends. This happens because of the increased overhead in the calculations and additional iterations performed to choose the cluster of concepts. But even then, for depth less than 4 the time is less than that for the baseline gazetteer approach in which the searching of gazetteer proves to be expensive on time.

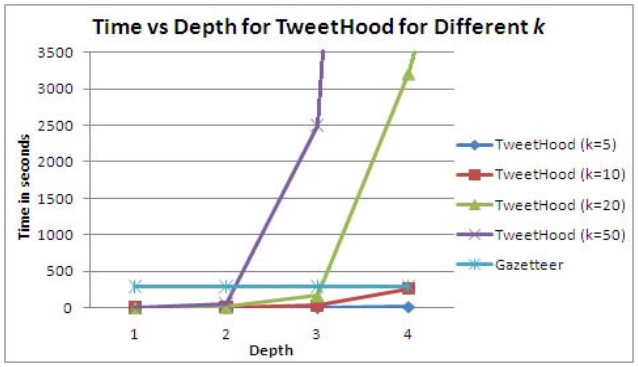

Fig 8. Time vs Depth for different values of *k*

Finally, we discuss the affect of increasing *k* on the time complexity of the algorithm. The increase is still exponential, but with the greater value of *k* the greater is the slope. In fact, we have just shown even for depth =4 the graph for k=50 becomes too large to be considered for practical use.

VII. CONCLUSION AND FUTURE WORK

We present the development of a system that predicts the location of the user based on his social network. We choose the values k=10, d=3 for TweetHood because of its optimal combination of accuracy and time complexity. Our experiments show that we are able to correctly identify the location of the user at the city level with an accuracy of 60.1% and concept of group of cities with 72.1%. The accuracy for country level identification is reported to be as high as 80.1%.

The system performs better than the traditional text based approach, in respect to both time and accuracy and is thus suited for the real time applications. Even though IP address of a user can predict the location with most accuracy, only Twitter has the IP address of the users. On the contrary Tweethood provides a simple yet effective algorithm that can be used to identify the location of user with a public profile.

The accuracy can be improved further by taking other attributes like temporal nature of the friendship into account. Even though time complexity is currently the biggest concern, it can be tackled by using the distributed computing or cloud computing framework. By doing this, we can geo-tag users on the fly and build real time web applications.

The main sources of errors are that firstly, in certain cases the users provide incorrect information about their location e.g. “Rome (I wish)”. Another cause for the errors arises from the ambiguity in place names, e.g. Yorkshire is a historic northern county in UK as well as a city in South Carolina, US. Finally, for the single city identification accuracy is significantly lower than country level because of the nature of society in the 21st century. People keep moving to different places from time to time for jobs, school etc. In such cases, the friends of a person belong to different cities and it becomes difficult for the algorithm to predict the current location correctly.

ACKNOWLEDGMENT

We wish to acknowledge Dr. Mehedy Masud, University of Texas at Dallas for his invaluable help and support.

REFERENCES

[1] ABI Research. [Online].Available: http://www.abiresearch.com [Accessed: May. 1, 2010].

[2] Socialnomics.net, Socialnomics – Social Media Blog. [Online].Available: http://www.socialnomics.net [Accessed: May. 1, 2010].

[3] Time in Partnership with CNN, Iran Protests: Twitter, the Medium of the Movement. [Online]. Available: http://www.time.com/time/world/article/0,8599,1905125,00.html [Accessed: May. 1, 2010].

[4] CRF Tagger, [Online]. Available: http://sourceforge.net/projects/crftagger/ [Accessed: May. 1, 2010].

[5] TIGER gazetteer [Online].Available: http://www.census.gov/geo/www/tiger/ [Accessed: May. 1, 2010].

[6] Li, H., Sihari, R.K., Niu, C., and Li, W. in “Location Normalization for Information Extraction”, 19th International Conference on Computational Linguistics, Aug. 2002,

[7] E. Amitay, N. Har’El, R. Sivan, and A. Soffer. Web-a- Where:geo-tagging web content. In Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 273-280

[8] A. Mehler, Y. Bao, X. Li, Y. Wang, and S. Skiena. Spatial analysis of news sources. IEEE Transactions on Visualization and Computer Graphics, 12(5):765--772, 2006.

[9] J. Liu and L. Birnbaum. Localsavvy: aggregating local points of view about news issues. In LocWeb, pages 33--40, 2008.

[10] C. Sheng, W. Hsu, and M.-L. Lee. Discovering geographical specific interests from web click data. In LocWeb.