Extracting a Graph Model by Mapping Two Heterogeneous Graphs

[1]Asad Feroz Ali, [2] Zohaib Jan

[1,2]Shaheed Zulfiqar Ali Bhutto Institute of Science and Technology, Karachi Pakistan

[1]AsadFerozAli@gmail.com
[2]Zohaib.Jan@szabist.edu.pk

Abstract—With the development of wireless communications, several studies have been performed on Location based Services due to their numerous applications. Amongst those recommendations, Travel Planning and Recommendations are few of the active topics. When it comes to movement patterns and mobility, human beings are restricted in motion due to social, financial and geographical constraints. Using check-in data from a former location based social network namely Gowalla and airport flights and route data from openflights.org, authors aim to extract a graph model from time series data of user check-ins. In this study, authors have identified patterns of location (latitude, longitude) visits using directed weighted graph. In addition to it, we have mapped airports to identified maps using latitude, longitude and used routes. This has helped to identify nearest airport routes probably used by Gowalla users. Hence, by mapping two heterogeneous graphs i.e. the Gowalla check-in data and openflights.org airport and flights data, we have tried to extract the most commonly used airport routes travelled by Gowalla users.

Keywords—Location, Timeseries, Social network analysis, Multivariate timeseries

I. INTRODUCTION

Although human movement involves a lot of freedom and variation, researchers have identified interesting patterns from observing publically available data provided by location based social networks [1]. These patterns exists primarily because of the social, geographical, financial and time constraints that people face each and every day which limits and restricts their movement. Gowalla is a type of social network which can be classified as a (LBSN) i.e. Location-based Social Network where people having similar interests and commonalities connect together based on geographical locations. Unlike other social networks, people primarily use LBSNs' to share their experiences at various places that they have visited with their friends and other users of the network. Due to time restrictions, people prefer to visit locations like restaurants, cafes, parks and hotels which their friends have visited in the past. Feedbacks given by social contacts on various locations also helps guide which location a person would prefer.

In addition to this, people prefer locations which are easier to reach during their daily commute from work to home and vice versa. People would prefer a place which falls on the route so they don't have to discard extra time and fuel to visit a particular place. Apart from its online connectivity, people also get additional benefits from using the social network. Restaurant and café owners could offer users complementary drinks or desserts upon checking into their café or for posting a review. To prove all these reasons which restrict and shape human mobility, restricted data is available due to the privacy policy of various users because people wouldn't like a scientist or a researcher to identify and map their movements by mining their previously visited locations on social networks as illustrated in figure 1. However; in 2010, Gowalla made this data available publically which had check-ins data for 18 months from 2009-2010 [2]. This data did not included exact places that people visited but mentioned latitude and longitude co-ordinates of check-in locations. The exact check-in locations would have been really helpful because if in a building there is a bank on 2nd floor and a café on 5th floor, co-ordinates data won't help us identify whether a user visited the bank or the café.

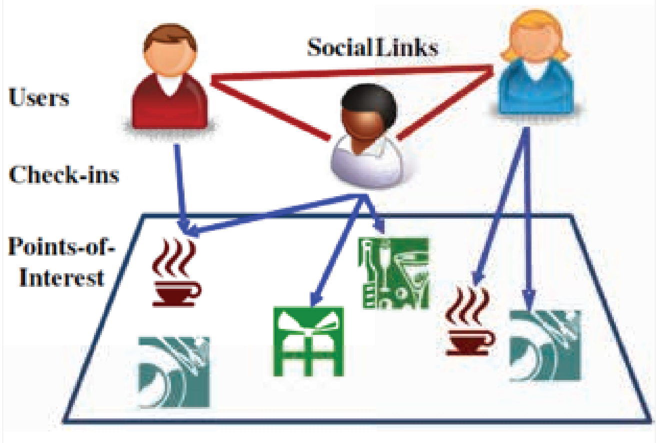

Fig. (1). A Location-based Social Network

Openflights.org [3] is an open source project which provides an interesting platform to share one's flights and trips with friend as well as calculate statistics like kilometers travelled and etc. For data mining enthusiasts, it provides a comprehensive lists of airports, airlines and routes data which spans around the globe. This study aims to map airport locations to the check-ins data so these co-ordinates would help identify if a user used an airplane to reach a particular route or some other mode of transport. Identifying this pattern can have many applications in the real world. Location based networks could offer special deals to people based on their location data if the pattern helps them identify that a user would be using an airplane route for movement to another location. This research is an empirical research and the focus would be to extract a graph model from check-ins data and identify useful and interesting patterns of human mobility by mapping the check-ins graph to the airline routes graph and predict which users chose the airline path.

II. PREVIOUS WORK

Eunjoon *et al*. [1] has studied human mobility patterns using the Gowalla, Britekite and cellphone data to identify user movements in location based social networks. They had 2 main hypothesis to answer which were whether the mobility has effect on friendship and vice versa and whether user check-ins are periodic. They explored that people have strong patterns as they commute from work and home and social relationships shape their mobility. They found that the distribution follows a power law cuts off at 100km distance and all three datasets showed similar behavior. Moreover, they explored a high probability of friendship among groups of users who has several similar check-ins.

In the domain of trip recommendation, Eric *et al*. [4] proposed an efficient trip planning approach called Trip-Mine which proposes trip planning with travel and time constraints. The idea of trip mine is to suggest the users', efficient trips in terms of time and reviews so when the user queries the system for a trip and specifies their time constraints, the system calculates various trips according to the time and reviews of various people and suggests the user the best options within their specified constraints.

Jia-Dong *et al*. [5] presented a time aware location recommendation namely TICRec which utilizes temporal influence co-relation. They treat the time of check-ins as continuous variable rather than characterizing it in discrete time slots because they identified that pattern of visiting various location defers at various hours of the day as well as during the weekdays and weekends. This means that if a user queries their system at 9 pm for an interesting place to visit nearby, the system would not show all possible restaurants and locations nearby according to reviews but the system would consider the places which are mostly visited during the queried time and hence would present the results where people generally visit during that time.

Hsun-Ping *et al*. [6] suggested a time sensitive route recommendation called TimeRouter which recommends time sensitive trips to the user. Their system considers several factors such as the popularity of a location, the visiting order of location i.e. people usually visit restaurant or café after a gym rather than vice versa, the time of visiting a place and time duration it takes to move from one location to another. They used Gowalla dataset to carry out experiments on their recommendation system.

Tim *et al*. [7] presented TimeMachine. An automated algorithm to generate a time line of relations and events for several entities in a knowledge database. They developed a couple of orthogonal quality criteria for their timeline that an ideal timeline would satisfy.

Yasushi *et al*. [8] presented several pattern detection and recognition techniques for time-series data. They emphasize the importance of large scale tensor analysis, automatic mining of data and nonlinear modeling. It is actually a form of a tutorial where they provide brief and intuitive overview of several important tools that researchers can use to understand and find patterns in large scale time series data that they have collected.

III. EXPERIMENTAL EVALUATION

This section describe the different set of processes that were carried out on the different data sets to extract the graph model.

A. Dataset Description

Authors used a publicly available large-scale real check-in dataset which was crawled from Gowalla between Feb 2009 and Oct 2010 as to make it a part of Stanford Network Analysis Platform dataset library. The statistics for the data is shown in the table 1. The Gowalla check-in data has 6,442,892 check-ins made by 107,092 users at 1,280,969 different locations worldwide.

Table 1. Statistics of Gowalla Dataset

Number of users	107,092
Number of locations	1,280,969
Number of check-ins	6,442,829
Average number of check-ins per location	5.03

The check-in times are provided in UTC time zone which are time stamps that lists the check-in day and time at a particular venue. The exact venues are not listed. Instead each particular venue is given a venue id (VID) and these VIDs' are present in the check-in data. Each check-in has latitudes and longitudes co-ordinates which are associated with vid such that if the same co-ordinates appeared elsewhere i.e. if another user checks-in at the same location which has the same co-ordinates then that check-in has the same vid.

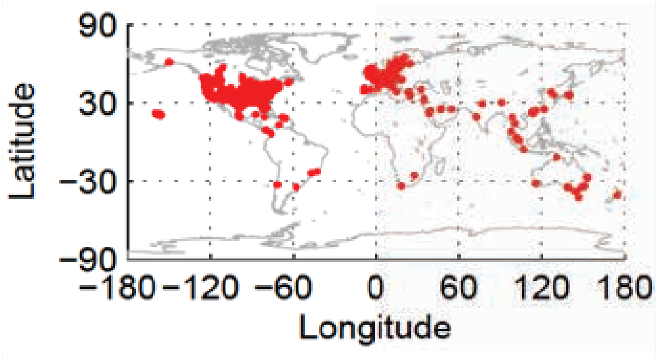

Fig. (2). Distribution of Gowalla check-in locations on a world map.

Figure 2 shows the distribution of the check-ins on a world map. To map this check-ins data to the airline data and find the commonly used routes, authors have used the publicly available OpenFlights data which provides the Airports, Airlines and Routes data from around the globe. Several airlines operate in various cities such many of them lead to the same other city like the other airline. Since, it is not possible to find the specific airlines used by the user so the Airline data has not been utilized.

The Airport and Routes dataset gives details for several parameters as listed in the tables 2 and 3.

Table 2. Details of the Airports Database provided by Openflights.org

Airport Id	Unique Identifier for an airport
Name	The name of an airport
City	The city where an airport is present
Country	The country where an airport if present
IATA/FAA	3 letter IATA/FAA code for an airport
ICAO	4 letter ICAO code for an airport
Latitude	The latitude for an airport in decimal degrees
Longitude	The longitude for an airport in decimal degrees
Altitude	The altitude in feet
Time zone	The time zone offset in terms of UTC
DST	Code for daylight savings time
Tz Database time zone	Time zone in tz Olsen format

Table 3. Details of the Routes Database provided by Openflights.org

Airline Code	IATA or ICAO code for the airline
Airline Id	Unique airline Id from the airlines table
Source Airport	IATA or ICAO code for the source airport
Source Airport Id	Unique airport id from the airports table
Destination Airport	IATA or ICAO code for the destination airport
Destination Airport Id	Unique airport id from the airports table
Codeshare	Yes or No value
Stops	The number of stops that a flight takes
Equipment	Plane type used by the flight

B. Evaluation Technique Description

The check-ins data for Gowalla, Airports and Routes were read by the R [9] program. The first step was to analyze where the check-ins were made i.e. based on the latitudes and longitudes provided for each of the check-in, Authors aimed to find the city and country for each of the check-in. Once the city and country for a check-ins was found, the next step was to identify which of those check-ins were made near the airport. The Latitudes and Longitudes co-ordinates provided in the openflights airport data did not points towards the waiting area or the lounge of an airport but instead just points anywhere inside the boundary of the whole airport area. In case of London airport, the co-ordinates points on the runway. Naturally people won't check-in on the runway but rather in the airport lounge, waiting area or the cafeteria while waiting for their flights. For this purpose, authors tried to use the check-ins which were made around within 2km of radius of the airport co-ordinates. In many cases, this 2km radius around the airport could point to some one's office or check-in made in a restaurant near an airport as well because in busy cities, Airports are compactly situated in the main city areas. To find the actual airport users, authors would then find the next check-in made by the user which is in another city near an airport. This would filter the airport check-ins from non-airport check-ins.

For step 1, we have to process each of the 6.44 million check-ins and find the city and country for a check-in and then for step 2, we would have to look through the airports database to find if any of the check-in was near 2km of any of the airports present within the city where the check-in was made. To find the city and country of the check-in, geo location services, which offer their APIs for data processing, could be used. The API could be used to query through the check-ins data and find the city and country. Two such services which offer geo location API are Google Maps and Open Street Map. Each have their limitations. Google Maps has a limitation of 2500 free requests per day up to 100,000 requests per day for paid subscribers and Open Street Map has limitation of 15,000 requests per day. The number of unique locations in the Gowalla check-ins data is 1,280,969 so we created a separate table of these unique locations / venues and then to find the cities and countries of their presence. This method speeds up the process because five times less queries are present in the venues table which have to be processed as compared to the original number of queries in Gowalla dataset. In spite of that it would take several days to find the cities and countries because of the daily API limitations.

Another approach which is the one that we have used to solve this problem is to use the airport table to find the location of check-ins. This method is computationally expensive in terms of calculations but it wouldn't require the usage of geo location service providers' API. For this process, the distance for each of the 1.28 million venues was calculated with each of the 8107 airports. This requires calculating the distance 10.38 billion times. By creating a matrix such that the rows of matrix represents each of the venues and columns represent the distance to each of the 8107 airports for each venues. Once the matrix is created it could be used to find the minimum value of distance amongst 8107 columns for each of the rows. This distance would help us identify which airport is the closest to a venue. This means that the city and country of the venue would be same as the city and country of that closest airport. Moreover, by evaluating whether that minimum value is greater than or less than 2km, we can calculate the answer to the second step and find which of the check-ins were made near an airport.

Since a matrix of 1.28 million rows and 8107 columns would not possible to keep in memory of a laptop computer which was used for this evaluation, we decided to process 20,000 rows of venues at a time i.e. at any given time, there would be 20,000 rows and 8107 columns in the matrix and all the rows of venue table were processed in chunks of 20,000 rows. Another important thing is that the matric column headers pointed to the Airport table's Airport Id. Hence, after performing distance function on each of the rows for all the airports, the matrix was kept in memory until minimum distance for all the rows of data was evaluated. This minimum distance was used to look up in the Airports table and it would point to the index or the airport id of the airport's table row and hence using that row, the nearest city and airport were extracted. After this step, the distance to the airport can be checked to check whether it is within the 2km range. For this, another column was added to the table titled as "Near Airport". This would store logical values of yes or no for each of the venue id's and helps to identify if a venue is near the airport.

In the next step, all the venues with a value of "No" in the "Near Airport" field were removed from the venue table. This venue table was then merged with Airports table to find the name of Nearest Airports, Cities and Countries for the remaining entries present in the venue table. Then in the next step this newly merged venue table was scanned to find the list of venue ids present in the table. That list was compared with the list of venue id present in the Gowalla check-ins table. Then from the Gowalla check-ins table, all those venue Ids were kept that were present in the Venue Table and the remaining entries were removed since they were not near the airport. The Gowalla dataset was then merged with the venue table based on venue Ids. Hence for each of the check-in, the city of check-in, the country and the nearest airport of that check-in was obtained.

In the Gowalla merged table, only those entries are present which are near the airport. But for confirmation that the person indeed did make a trip from one city to another and checked-in on airports at both the locations, several filters were applied to remove the irrelevant rows of data. Irrelevant rows of data means that there is a possibility that a user was near 2 km of 2 airports but the user was in a shopping mall near airport in both the cases or in a train in both the cases rather than being near an airport. Hence for further refinement of the dataset, another column was added to the table to mark with values of Yes (Y) or No (N) for whether a row is irrelevant. Those filters are as follows:

1. Delete check-ins of same cities sandwiched between other rows. For each of the rows of the dataset compare the current row with previous and next row to check if those entries are for the same user. If they are for the same user then check for the cities. If the cities are same then a trip across cities wasn't made and all such entries should be marked as Y and deleted. This step reduced around 65% of rows from the merged table.

Pseudo Code: The data frame "gowallaMerged" contains only those check-ins of users which are near the airport, "vid" is venue Id for a particular location, "uid" is user Id for a particular Gowalla user, "utc" is time in UTC time zone for a particular check-in made by a Gowalla user at some venue, "city" is the name of the city where the check-in was made, "irrelevant" is a column in the data frame which identifies whether a check-in is relevant or not, "airportType" identifies whether the check-in was made near a source airport or destination airport and is relevant for mapping on the graph.

```
a. [Initialize] Set nrow = Number of Check-ins +in gowalla Merged data frame.
b. [Initialize Counter]
c. Repeat Steps below until counter is equal to nrow
    I. IF current uid is equal to the next uid AND current uid is equal to the previous uid AND current city is equal to the next city AND current city is equal to the previous city
            SET irrelevant to Y
    ELSE
            SET irrelevant to N
    [Increment Counter] Increment Counter by one
[End of Step 2 loop]
d. DELETE all the check-in rows marked as Y in the irrelevant column.
```

2. Delete last check-ins of all such users whose second last and last cities are the same. For each of the rows of dataset compare the user id and find the last check-in entry for each of the user. Once the last check-in of a user is spotted, compare the cities for the last and the second last check-in and if they are the same then trip across cities wasn't made and all such last entries should be marked Y and deleted from the merged table.

Pseudo Code

```
a. [Initialize] Set nrow = Number of Check-ins in gowalla Merged data frame.
b. [Initialize Counter]
c. Repeat Steps below until counter is equal to nrow
    I. IF current uid is not equal to the next uid AND
    current uid is equal to the previous uid AND current
    city is equal to the previous city
            SET irrelevant to Y
    ELSE
            SET irrelevant to N
  [Increment Counter] Increment Counter by one
[End of Step 2 loop]
d. DELETE all the check-in rows marked as Y in the irrelevant column
```

3. After deletion of check-ins the resulting table may have many single entries such that only one check-in exists for each of the user i.e. only one entry for a user exists in the merged table. To remove such rows all the check-ins should be scanned and compared such that for any given row, the previous uid is different from the next uid which suggests that it is the only entry for that user which is of no use and should be marked as Y and deleted.

Pseudo Code

```
a. [Initialize] Set nrow = Number of Check-ins in gowalla Merged data frame.
b. [Initialize Counter]
c. Repeat Steps below until counter is equal to nrow
    I. IF current uid is not equal to the next uid AND
    current uid is not equal to the previous uid
            SET irrelevant to Y
    ELSE
            SET irrelevant to N
  [Increment Counter] Increment Counter by one
[End of Step 2 loop]
d. DELETE all the check-in rows marked as Y in the irrelevant column
```

4. The next few steps deal with identifying the source and destination airports. So a new column was added to the table to identify whether that entry or check-in is a source airport or destination airport. After going through the conditions below we will delete all destination rows which appear next to each other and only leave the rows in such an an order that one is a source row and one is the destination row. So after scanning through each of the check-ins for all users, if a check-in has a different city when compared with a previous check-in but same city for the next check-in for the same user, then that entry can be marked as destination airport.

Pseudo Code

```
a. [Initialize] Set nrow = Number of Check-ins in gowalla Merged data frame.
b. [Initialize Counter]
c. Repeat Steps below until counter is equal to nrow
    I. IF current uid is equal to the next uid AND current
    uid is equal to the previous uid AND current city is not
    equal to the previous city AND current city is equal to
    the next city
            SET airportType to destination
  [Increment Counter] Increment Counter by one
[End of Step 2 loop]
```

5. The next step is calculating the time difference for entries having different city check-ins for the same user. The longest flight between 2 cities is less than 20 hours at most so keeping a time of 24 hours between check-ins to confirm travel between two cities, each of the check-ins should be checked for each user. For any particular user, if the next check-in is in a different city and the time difference is greater than 24 hours than that entry would be marked as destination airport.

Pseudo Code

```
a. [Initialize] Set nrow = Number of Check-ins in gowalla Merged data frame.
b. [Initialize Counter]
c. Repeat Steps below until counter is equal to nrow
    I. IF current uid is equal to the next uid AND city is not
```

```
        equal to the next city and the time difference for utc
        between the two values is greater than 24 hours
                    SET airportType to destination
    [Increment Counter] Increment Counter by one
[End of Step 2 loop]
```

6. Since step 2 deleted entries of same cities for the last check-in of a particular user, now we can compare the last entry with the second last entry again to see if it is different and if it is then it can be marked as destination airport.

Pseudo Code

```
a. [Initialize] Set nrow = Number of Check-ins in gowalla Merged data frame.
b. [Initialize Counter]
c. Repeat Steps below until counter is equal to nrow
        I. IF current uid is not equal to the next uid AND
        current uid is equal to the previous uid AND current
        city is not equal to the previous city
                    SET airportType to destination
    [Increment Counter] Increment Counter by one
[End of Step 2 loop]
```

7. In condition 1 sandwiched entries were removed and so in the current merged table sandwiched entries are not present but several entries are there such that for each of the user id, the first check-in and the next check-in are in the same city and all such check-ins are irrelevant and are marked as Y and deleted.

Pseudo Code

```
a. [Initialize] Set nrow = Number of Check-ins in gowalla Merged data frame.
b. [Initialize Counter]
c. Repeat Steps below until counter is equal to nrow
        I. IF current uid is equal to the previous uid AND
        current city is equal to the city
                    SET irrelevant to Y
    [Increment Counter] Increment Counter by one
[End of Step 2 loop]
d. DELETE all the check-in rows marked as Y in the irrelevant column
```

8. The next step is calculating the time difference between two entries and marking all such entries for the a particular user id as source airports in which the next check-in is in a different city and the time difference is less than 24 hours. If in n consecutive entries all the cities are different for the same user id and the time difference is less than 24 hours as well then all those entries would be marked as source airport and could identify stops between travelling.

Pseudo Code

```
a. [Initialize] Set nrow = Number of Check-ins in gowalla Merged data frame.
b. [Initialize Counter]
c. Repeat Steps below until counter is equal to nrow
        I. IF current uid is equal to the next uid AND current
        city is not equal to the next city and the time difference
        for utc between the two values is less than 24 hours
                    SET airportType to source
    [Increment Counter] Increment Counter by one
[End of Step 2 loop]
```

9. In the remaining dataset all the rows are either left marked as source airport or destination airport. In this step all those rows are marked irrelevant such that the destination airport is the first entry in the check-in list for a particular user or destination airport is followed by or preceded by another destination airport. All such rows are marked as irrelevant because the travel can only be marked as relevant if a trip is made from source airport to another source airport or from a source airport to a destination airport. All the irrelevant rows are deleted and the remaining dataset is ready to be plotted in a map.

Pseudo Code

```
a. [Initialize] Set nrow = Number of Check-ins in gowallaMerged data frame.
b. [Initialize Counter]
c. Repeat Steps below until counter is equal to nrow
        I. IF current uid is equal to the previous uid AND
        current airportType is equal to destination and previous
        airportType is equal to destination
                    SET irrelevant to Y
    [Increment Counter] Increment Counter by one
[End of Step 2 loop]
d. DELETE all the check-in rows marked as Y in the irrelevant column
```

After these steps only collection of venue trips were present which were made by user to go from one location to the other such that those venues were near airports. Although those venues were in different cities and near airports, it was still possible that a person could have travelled through train or car and could have been near airports while checking-in. For this purpose the Routes tables was utilized. The Routes table has a list of all the routes made by all the flights around the globe. Utilizing that table for each of the trips made by user, it could be identified if Routes table has a record for a flight which serves between the two locations. Suppose user travels from venue A to venue B and the routes table has no entry for such a route being served by any of the airlines of the world, then it means the user made that trip in some other medium of transport rather than air travel. Therefore, each of the trips in the table were checked with the Routes table and all such trips were removed which had no flight record in the Routes table.

Now the check-ins table has all the trips which were verified from the Routes table. There were several entries which were repeated i.e. N number of users travelled from A to B and M number of users travelled from B to C. Hence, to plot on a map, mode was calculated for each of the unique routes to find the routes mostly used by Gowalla users i.e. to highlight all such routes on map where maximum number of user checked-in before travelling through air.

VI. RESULTS

The check-ins table has 90,006 trips made by Gowalla users where they checked-in near the airport. After comparison with the Routes table only 46,018 or 51.12% routes were travelled through airport which means that the remaining 43,988 or 48.88% routes were travelled using other means of transport. Out of those 46,018 routes, 6788 were unique routes. Of these unique routes, 2421 routes i.e. 36% of these routes were only made once. 84.6% of routes were used less than 10 times, 1.78% of these routes were used more than 50 times and 0.43% of these routes were used more than 100 times.

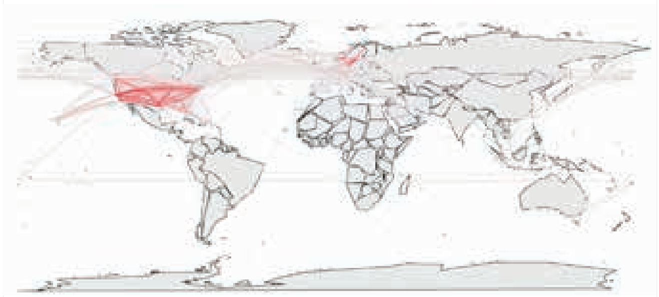

Fig. (3). Map showing airport routes used by Gowalla users. Red shade depicts maximum used routes

Figure 3 shows the map which highlights all the routes made by Gowalla users through airports. The colors are transitioned from grey to red such that least used routes have a shade of grey while most used routes are in shades of red.

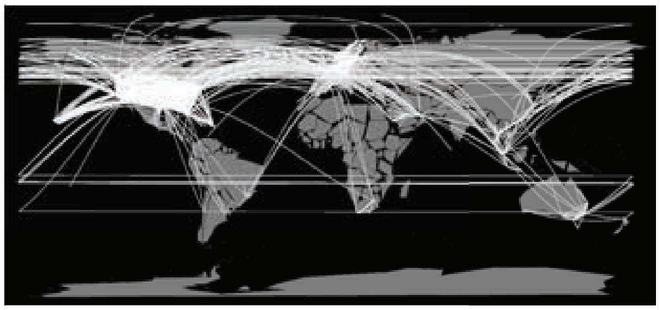

Fig. (4). Map showing airport routes used by Gowalla users. Red shade depicts maximum used routes

Among the routes used more than 100 times, the 3 countries which were present in the routes were United States and European countries Norway and Sweden. In addition to that the countries covered in the routes used 50 or more times includes Saudi Arabia and Denmark in addition to the countries above. Moving down to countries used 10 or more times includes Asian countries Japan, Hong Kong, Thailand and Indonesia as well as New Zealand and Australia. In addition to it, South American Countries such as Brazil and Puerto Rico are also present. For better visibility the black and white map shown in figure 4 highlights the routes like the map in figure 3 but here white routes clearly shows the mostly used routes which can be spotted in United States and parts of Europe and Asia

V. CONCLUSION AND FUTURE WORK

In this paper, authors have tried to extract interesting and useful information by mapping the Gowalla check-ins data to the OpenFlights dataset and have extracted the list of mostly used airport routes used by the Gowalla users. We have used the Airports data to calculate the distance and find the nearest airport for check-ins made by users and then compared the routes dataset from OpenFlights to find which of the routes or trips made by Gowalla users were travelled through airports and which ones were travelled by other means of transport. In the end, we have tried to map the routes on the world map to give a visualization of countries where the mostly used routes are located.

For future work, these routes can be analyzed by friendship data i.e. how many users made a trip as a result of their friends making that trip which would mean peer recommendation increases the likelihood of a user making a particular trip. Moreover, using the time interval through time stamps mentioned in the data and using cell phone signals data for further accuracy, user stay at a particular location can be used to identify the most popular places visited by users and can be used in recommendation system to users who are looking for vacation trips. Cell phone data would be helpful because if a user forgets to check-in the cell phone data could be cross checked with the check-ins to determine the time taken to reach a particular destination as well as the length of stay.

The longer stay of trips would signify that users liked a particular location better than another location. Apart from that smart phones have the ability to allow an application to connect to a new mobile signal tower when a user is travelling from one area to another. Based on the information obtained from the former trips data made by the users, the applications can suggest the user how long the trip will take based to mobile signal towers data in case of lower priced smart phones which are not equipped with GPS feature. In addition to this, the result from trips which were not made by airport could be studied to identify why users chose to travel using other modes of transport.

REFERENCES

[1] E. Cho, S. A. Myers and J. Leskovec, “Friendship and mobility: user movement in location-based social networks,” In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining* (KDD '11), 2011, pp: 1082-1090.
Doi: 10.1145/2020408.2020579

[2] *Gowalla dataset from Stanford*, [Online]. Available: https://snap.stanford.edu/data/loc-gowalla.html. Accessed: 2016-02-03.

[3] *Airpot, airline and route data from Openflights.org,* [online]. Available: http://openflights.org/data.html Accessed: 2016-02-03.

[4] E. H-C. Lu, C-Y. Lin and V. S. Tseng, “Trip-Mine: An Efficient Trip Planning Approach with Travel Time Constraints,” In *Proceedings of the 2011 IEEE 12th International Conference on Mobile Data Management - Volume 01* (MDM '11), 2011, vol. 1, pp: 152-161.
DOI: 10.1109/MDM.2011.13

[5] J-D. Zhang and C-Y. Chow, "TICRec: A Probabilistic Framework to Utilize Temporal Influence Correlations for Time-aware Location Recommendations," *IEEE Transactions on Services Computing*, vol. 9, no. 4, pp: 633-646.
DOI: 10.1109/TSC.2015.2413783

[6] H-P. Hsieh, C-T. Li and S-D. Lin, "Measuring and recommending time-sensitive routes from location-based data," *ACM Transactions on Intelligent Systems and Technology*, vol. 5, no. 3, Article no. 45.
DOI: 10.1145/2542668

[7] T. Althoff, X. L. Dong, K. Murphy, S. Alai, V. Dang and W. Zhang, “TimeMachine: Timeline Generation for Knowledge-Base Entities,” In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15)*, 2015, pp: 19-28.
DOI: 10.1145/2783258.2783325

[8] Y. Sakurai, Y. Matsubara and C. Faloutsos, “Mining and Forecasting of Big Time-series Data,” In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD '15)*, 2015, pp: 919-922.
DOI: 10.1145/2723372.2731081

[9 *R: A language and environment for statistical computing. R Core Team (2015)*, [Online]. Available: URL https://www.R-project.org/.