NUS National University of Singapore | School of Computing

CS5228 Knowledge Discovery and Data Mining.

Personalised Events Recommendation

Group - 16 Project Report

NAME	MATRIC ID
Jayakumar Alagappan Meenakshi	A0134431U
Krishna Kannan	A0134475A
Thang Trung Nguyen	A0137648U

1 Introduction

With the increase in the number of events published all the time in the Event-Based Social Networks (EBSN), it has become increasingly hard for the users to find relevant events according to his/her preferences. Natural solution to this problem is a Recommender System. In our case of Events Recommendation, the scenario is different from conventional recommender systems. The problem here is an intrinsic new problem and lack of availability of collaborative data. Existing social networking platforms like 'Meetup' and 'Plancast' have gained a large traction because of the ability to connect people based on the events they have attended. But with the increase in number of users and events, users were not able to get event invitations in which they were really interested in. Therefore we have considered the important features of EBSN such event lifetime, co-participation, locality and sparsity in building our model to solve the issues in the existing Events Recommender Engines.

2 Background

Recommender Systems

A Recommender System aims to predict the opinion of a user over a given item. Predicting with high accuracy is very important when recommending items to a user that involves the interest of the user. The interest can be derived from various factors and attributes related to the user. Thus generally a recommender system involves in estimating the probability of a user liking a given content or an item or an object. The recommender systems are classified as follows.

Content Based Recommendation

These recommendation systems primarily recommend items or content to the user based on the user's past preferences. Let us take a case where the user uses an e-commerce application. Over the course of time the e-commerce application collects the usage patterns and preferences of the user ; and based on those historical data, the recommendation model is built and the recommendations are made.

Collaborative Recommendation

The recommendation system first finds the similar users and collects the information about his/her preferences and usage patterns; and based on those data, the model is built and the recommendations are made to the user. This is possible with the explosion of available social data. The model leverages on the social data and enriches the model; and can be used to overcome certain hurdles of cold start problem that is frequent in conventional recommender engines.

Hybrid Approach

Certain recommendation models take the best of both the worlds i.e. models and build a hybrid model that has the positive characteristics of the above mentioned models.

3 Related Work

Collaborative future event recommendation

In this paper the authors[1] propose a model that does not rely upon the feedback. The events almost all the time may not have feedbacks. The proposed model recommends events to the user based on the user's past preferences for events and combines it with other user's preferences and likes. The similarity between the users is based on the deducing a ranking function predicting the interests of the individual users and the degree to which users share these factors.

Event-based Social Networks: Linking the Online and Offline Social Worlds

The authors [2] observed that in certain social networking platforms, there exists a new form of social network which they term as Event Based Social Networks. They use these social networks to observe the properties of the user such as degree distribution and how the location is playing an important role in social interactions. They also study the interplay between the online and offline interactions of the users. Online interactions can be found based on the expression of similar interests towards a particular content and offline interactions are measured with the help of location based social networks and a feature used in LBSN called Check-In. With these factors taken into account the authors proposed a recommender system which recommends events to the users in EBSN.

An Event Distribution Platform for Recommending Cultural Activities :

In this paper [3], the authors proposed on how the enrichment of events data with the general metadata available on the internet helps in improving the personal recommendations, and content based filtering. They propose a recommender system which makes use of the enriched data with content based filtering and collaborative filtering. This is one of the recommender systems that uses hybrid approach.

Social Recommendations for Events

Outlife Recommender [4] is proposed in this paper. Basically the recommender takes into account the user's past preferences and based on that, the events to be recommended are chosen. But instead of recommending it to a particular user, it chooses a group of users or group of friends closer to the user and recommends the chosen event to the group. In better words, the recommendations are made based on the preferences of the group members. The group of friends are composed automatically by the Outlife Recommender which takes the social interaction behaviour between the users.

Hybrid Event Recommendation using Linked Data and User Diversity

In this hybrid approach [5], the authors propose a system which is built over the semantic web. They use the content based recommendations along with the enriched data with the help of other linked data. This helps in overcoming the problem of data sparsity. Along with this, the system incorporates a collaborative filtering to include the social information which can heavily factor the decision making of whether to attend a particular event or not. This system demonstrates the importance of including both the content based and collaborative filtering systems. They also have a higher accuracy compared to other systems.

4 Understanding the problem and dataset

4.1 Data

The data were collected from an Events Recommendation Application. The application contains 16.92M distinct users. But the demographic information is available for only 38K users. The demographic information consists of the user's language territory, birthyear, gender, date of joining the application, location and timezone.

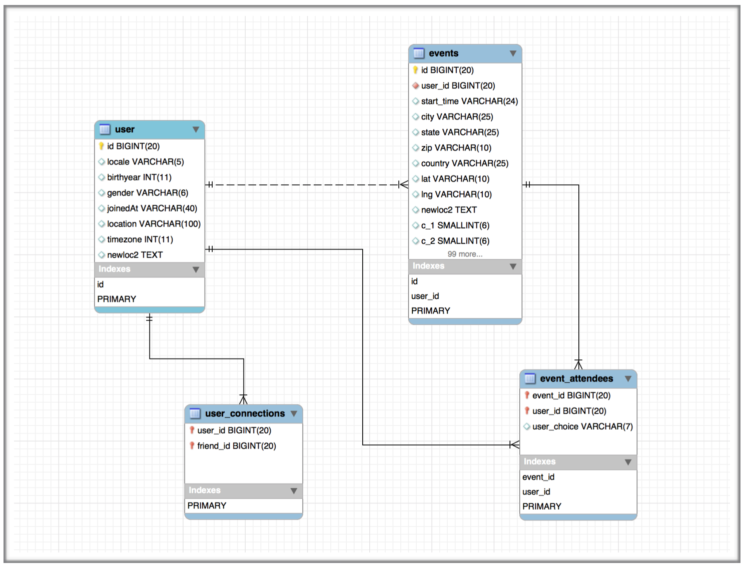
The user connections represent the social network graph i.e. friendship graph among the users. There are around 60M edges in the network.

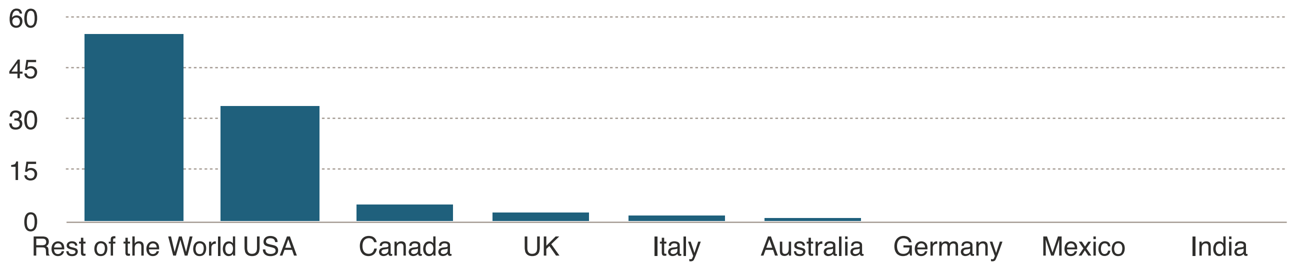
The system constitutes 3M events. The start time and the location information about the event is available. In addition to this information, another 101 data columns representing the most commonly used words appearing in the description or name of the event that is obtained with the application of Porter Stemming is also available. These data columns contain integer values denoting the number of times, the stem word that appears in the event description or name. If the data columns are represented as C1, C2, C3, .. and CO; the data column CN represents the N ranked word in the top 100 most common words used and CO represents the other appearing words in the event's description or name. Say if the value of C1 for an event is 10, then it denotes that the 1st ranked stem word appears 10 times in the name or description of that event.
Event attendance contains the RSVP information of a particular event i.e. response from the invited user. Around 11M recorded responses are available. The application contains 3 year's events data occurring between 2011 and 2014.

Distinct Users	16924627 (~ 16.92M)
User demographic data	38209 (~ 38K)
User Connections	60772806 (~ 60M)
Events	3138211 (~ 3M)
Event Attendance	11246276 (~ 11M)
Events Time Range	2011 - 2014 (3 years)

Data Set

Schema





Geo-Split

4.2 Understanding the problem

The problem of recommending events in an Events Based Network is different from traditional recommender systems problem.

New Event Problem. In those traditional systems, the item to be recommended would have been already rated or consumed by the other users of the system. In Events Based Networks, the events that are to be recommended are futuristic and cannot be rated in advance.

As a result, typical collaborative filtering approaches cannot be applied since there is a lack of collaborative data. It gives rise to the New Event Cold Start issue as these approaches require large piles of existing data about the events to recommend them to the users.

New User Problem. Moreover, these systems wouldn't be able to handle new users in the application. The new user will not be having any historical data in the application and the traditional content based recommendation fails severely in such a case.

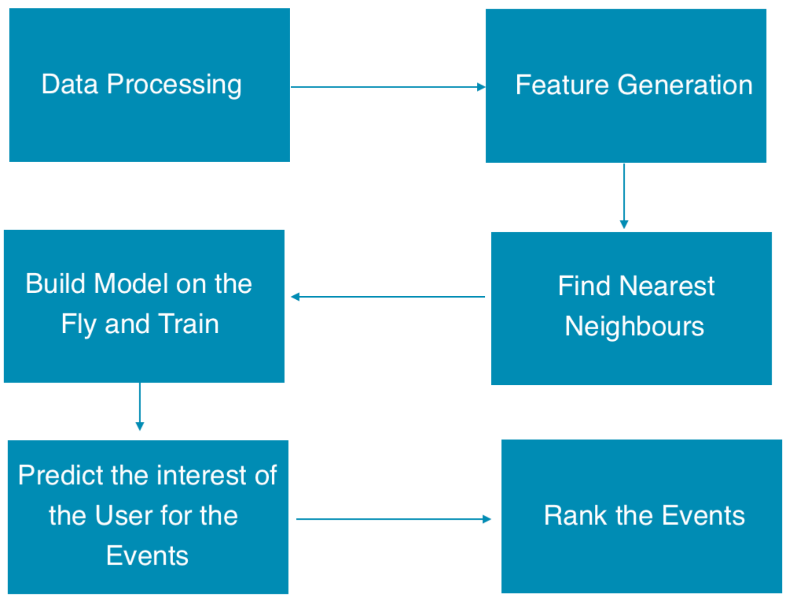
4.3 Proposed Solution and differences from existing work

Here we have proposed a solution that can address the above mentioned limitations which are specific to the Event Based Social Networks. We considered the social invitations and RSVPs as an explicit data in our proposed solution. RSVP - Répondez s'il vous plaît a phrase in french which can literally be translated to "Reply if you please". The explicit response of the users indicates his/her interest towards the invited event. Other factors taken into consideration are Events life time, location and geographic features and co-participation of users in events. In our solution our focus was to address the "New User" Problem and "New Event" Problem.

This solution follows the semi-Lazy Mining Paradigm (LAMP) to recommend the events to the users. The advantage of following the LAMP Paradigm is, we apply the lazy learning paradigm and find the k nearest neighbours and perform the eager learning on the set of nearest neighbours found from the preceding step.

We performed the k-Nearest Neighbours (kNN) search for the user as a first step and this solves the "New User" problem as the past preferences can be estimated by observing the neighbour's past preferences and attendance patterns.

From the k Nearest Neighbours, we retrieve the events for which we have the RSVP data of the users (Inclusive of events that these users were interested and not interested in). Then on these events, we built the classification models that can predict the interest of the user in an event. Since we retrieve the nearest neighbours of the queried user, we made the assumption that the model would be a good approximation for predicting the interest of the querying user.



Data Processing
Feature Generation
Find Nearest Neighbours
Build Model on the Fly and Train
Predict the interest of the User for the Events
Rank the Events

Overall Flow

We also addressed the “New Event” issue by finding the similarity between the new event and other existing events in the system. The similarity between the events were found based on the factors like location and occurrence of common root words found in the event description.

Other social factors such as past patterns of friends attending the events together and not attending the events were also taken into consideration while building the model.

Since the solution proposed follows the LAMP Paradigm it can handle data at very large scale with compromising the accuracy of prediction. By finding the nearest neighbours we are reducing the set of data over which the we perform the eager learning which is costly. We have implemented the above mentioned solution in Apache Spark, a fast and large scale data processing engine. We have mined all the social information with the help of NoSQL Graph Database - Neo4j.

5 Implementation

Tech Stack

Language	Python, Java, SQL, CQL
Libraries	numpy, scipy, scikit, pandas
Database	MySQL, Neo4j
APIs	Yahoo Placemaker API
Framework	Apache Spark

5.1. Data Preparation

5.1.1 Data Cleansing

Inconsistent data collected from the Events Recommendation Application are recognized and removed to reduce the irregularities in the available dataset.

Identified irregularities in data:
- NULL values in the required data columns i.e. User Id, Event Id, Event Timestamp.
- Invalid values in the data columns i.e. Invalid Event Timestamp, Invalid User Time Zone and missing User Location information

5.1.2 Data Enrichment

User data subset
- The missing Location data columns were populated from the Time Zone values.

Events data subset
- The missing State and Country data columns were populated from the City information.
- The missing City, State and Country data columns were populated from the latitude and longitude coordinates.
- The missing latitude and longitude coordinates were populated from the City, State and Country data columns.

Yahoo Placemaker API was used to enrich the geo data columns.

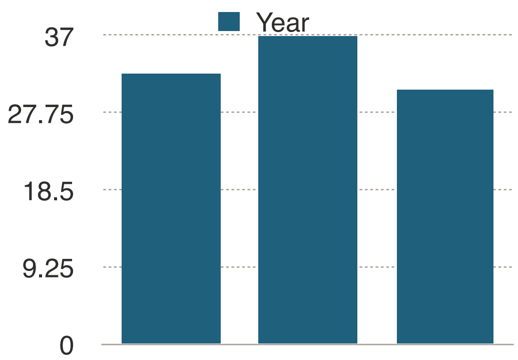
5.1.3 Data Transformation and Duplicates Removal

The format of the cleansed dataset was changed and normalized so that it could be loaded into MySQL and Neo4j. Then the duplicate data rows were eliminated.

5.1.4 Data - Temporal Split

The Events data subset were split temporally in the ratio 7:3:1 by choosing 3 proportional timestamps. 70% for Training, 20% for Cross Validation and 10% for Testing.

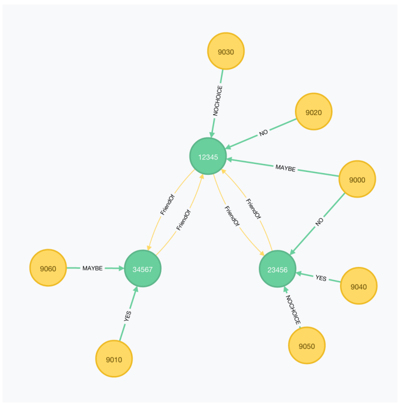
In more detail, when we built the classification model on the events to predict whether the user will be interested in an event or not, only 70% of the Events Related data i.e. the Event Information and Event Attendance were used for training. The next 20% data was used for Cross Validation i.e. To select the best value of k and the best Classifier model based on accuracy. The last 10% of data was used to make the final, unbiased conclusion about the accuracy that we hope to achieve.



Temporal Distribution

Since the data is always evolving, with new events coming into the system, temporal split on the events make accuracy on test data a good prediction on how the model will perform in the real environment.

5.2 Feature Engineering



5.2.1 Feature Generation

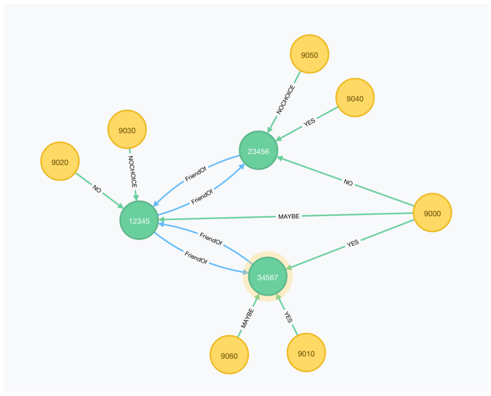
The below are the new features generated from the existing dataset and that were used in the model.

1) Number of Users -

- Attending
- Not Attending
- May be Attending
- Invited to the event

2) Number of User Connections (Until 3rd degree)

- Attending
- Not Attending
- May be Attending
- Invited to the event

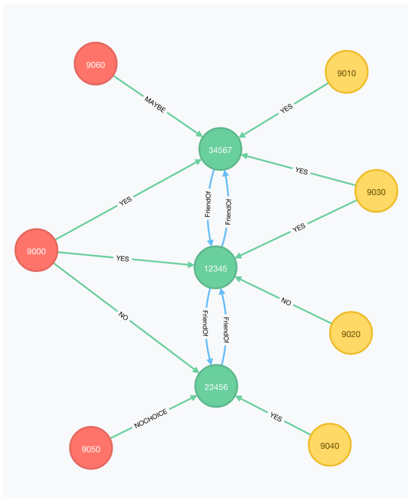


3) The similarity between the location of the user and the event.

4) For a particular user attending an event, number of users attending the current event who have also attended the past events along with the considered user.

5) Similarity between the considered event and the past events -

- the User attended
- the Friends i.e. connections (Until 3rd degree) of the User attended
- the User / Friends of the User did not attend



This similarity between the current event and the past events were found using the stem words occurring in the description of the events. K Means algorithm was used to determine the similarity.

6) Time to Event. The most important feature that denotes the time difference between the start of the Event and the time at which the User was notified about the Event.

7) Age Profile

8) Gender Profile

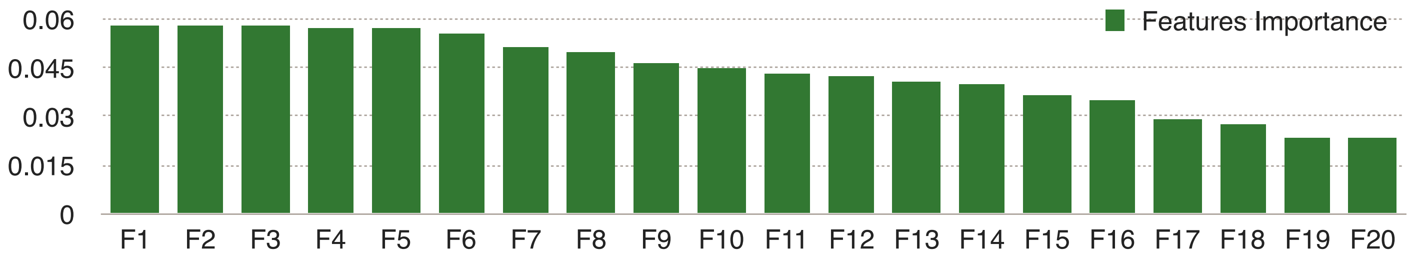
9) Was the User invited or not ?

5.2.2 Feature Selection

In the current problem, the output to the model is formalized as binary i.e. we have to predict whether the user will be interested or not interested in the futuristic event under consideration. So selecting one of the classifier algorithms to determine the necessary features for the model would be the next logical step.

A temporal subset from the existing dataset was considered. The new features were generated for the selected data subset. A Tree based Estimator from the scikit-learn package was used to calculate the importances of the features. The determined importance score is known as the ‘gini importance‘ or ‘mean decrease impurity’ which denotes impurity in the nodes averaged by the trees in the forest.

Then a Random Forest model was built and tested for the error rate. The features with the lowest importance scores below a particular threshold were discarded, the model was rebuilt and again tested for the error rate. This process was repeated until there was an increase in the error rate. The features that were used to build the previous model i.e. the model built before there was an increase in the error rate were used in the final model.



5.3. Querying Users - K Nearest Neighbours

As mentioned above, our solution follows the LAMP Paradigm and as a part of lazy learning, we performed k Nearest Neighbour (kNN) search. Finding the nearest neighbour is the first step involved in the solution. The kNN was applied on the user level and the nearest neighbours of the user under consideration were found. The kNN search was performed based on the below mentioned parameters.

- Gender
- Location
- Locale
- Age
- Time Zone in which user attended the maximum events

The dataset that we used to build the model had a relatively lesser number of user profiles i.e. 38 thousand. In the kNN Search, we chose an arbitrary k value and selected all the k nearest users. The connections of the k users until the 3rd degree were also considered. Then we selected the events attended by those users. The number of events attended by the users increased proportionally. The events obtained from the above process forms the candidate for the subsequent processes involving eager learning.

The kNN Search is implemented using the k dimensional-tree. k-d tree is a basically a binary tree. Every node in a k-d tree is a k-dimensional point. The average case running time complexity of all operations such as Search, Insert, Delete happens in linear time - O(logn). We used Python scipy library for building the k-d tree and performed the kNN Search.
The implementation of k Nearest Neighbors can be broken down into the following parts.

- Gathering User Data.
- Conversion of the user data into the suitable data structures in order to use the library

- Build the tree.
- For each user, query and find his/her nearest neighbour.

5.4 Querying Events

After getting the k nearest neighbors, we retrieve the events that these users and connections of these users (Until 3rd degree) have responded.

5.5 Classification Model

As discussed earlier in the Feature Selection section, we considered a Classifier approach as the output of the model is binary predicting the interest of the user in the futuristic event under consideration. Apache Spark's machine learning library, MLlib comes with some of the best Classifier algorithms. We tried the below algorithms to build the model with the already selected features.

- Decision Tree
- Random Forest
- Gradient Boosted Tree

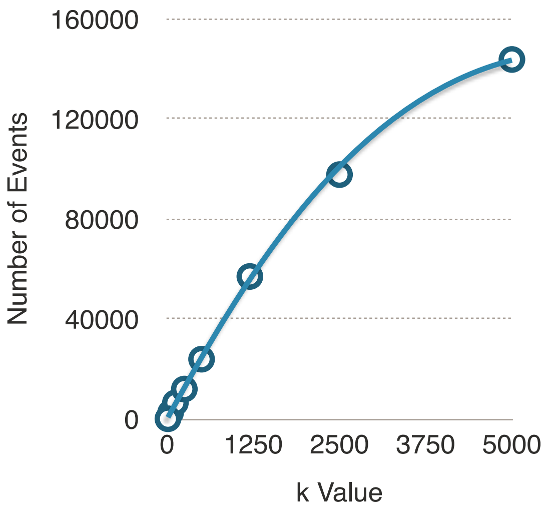
These algorithms are implemented using the MapReduce paradigm by the Apache Spark to allow scalability [1].

6. Experiment and Findings:

6.1.1. Extracting Similar users and Related events:

In the kNN search, we experimented with different values of K. For different values of K, the number of events retrieved are shown in graph.

From the documented observations, we can see that the values of k lie in the range of 0.1% - 10% of the total available User profiles. The corresponding events that were selected, were in the range of 0.1% -5% of the available events in the application.



Performance. For the k value of 5000 - the largest value that we experimented with i.e. for finding 5000 nearest neighbours, the Recommender Engine took less than 0.1 seconds per user.

6.1.2. Building the model:

Performance. The entire process flow from finding the k Nearest Neighbours, building the model, cross validating and testing the model took around 3 seconds per user when run on a single machine.

It took more than a day to run for the entire dataset. i.e. for 38,000 Users. However, since the Recommender Engine was built over the Apache Spark Framework, the most time consuming sections of the engine i.e. Querying Related Events, Building the model and Decision Making and Classification; can be distributed across multiple machines, thus achieving an acceptable runtime.

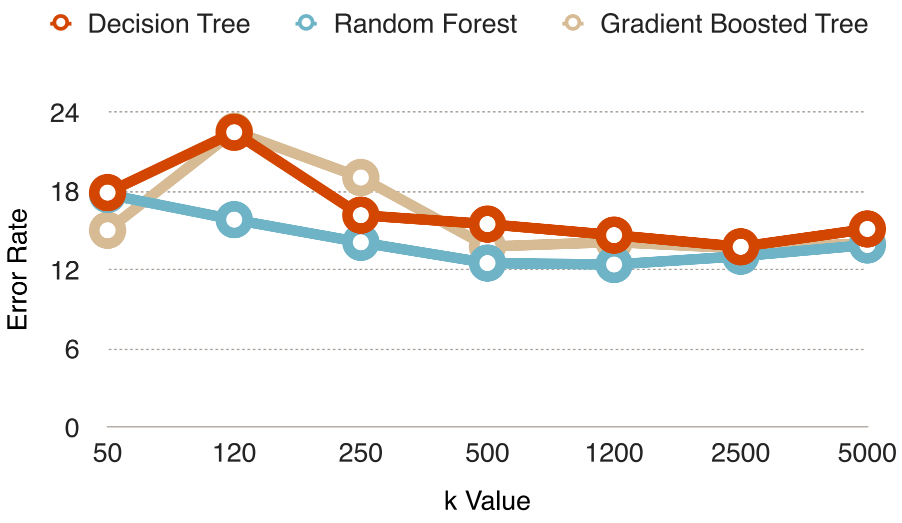
6.2. Accuracy

The following table sums up the cross-validation accuracy for different values of K from different classification models.

From the various runs, we found that Random Forest Classifier model with K = 1200 resulted in best accuracy. Thus, in our final model, we used Random Forest with K = 1200. On the 10% Events test data, this resulted in an accuracy of 16.8%.

Ranking of Events

After fetching the events, the queried user would be interested in attending, the events were ranked. The below criteria were taken into consideration to rank the events.



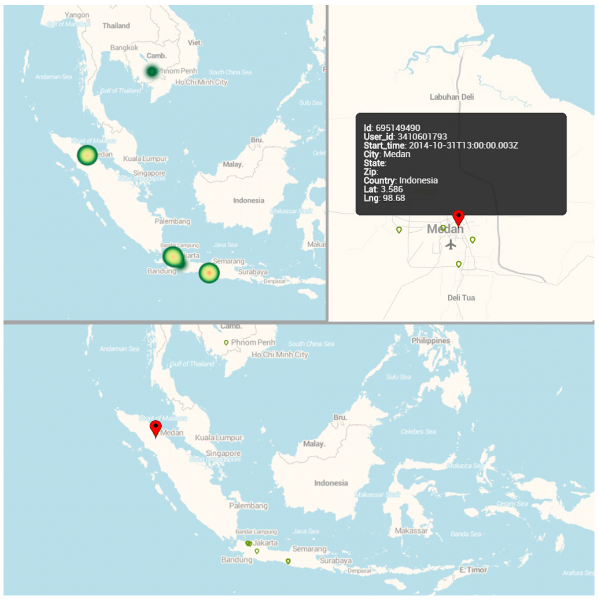
Location Aware. The distance between the User's location and place of occurrence of the Event.

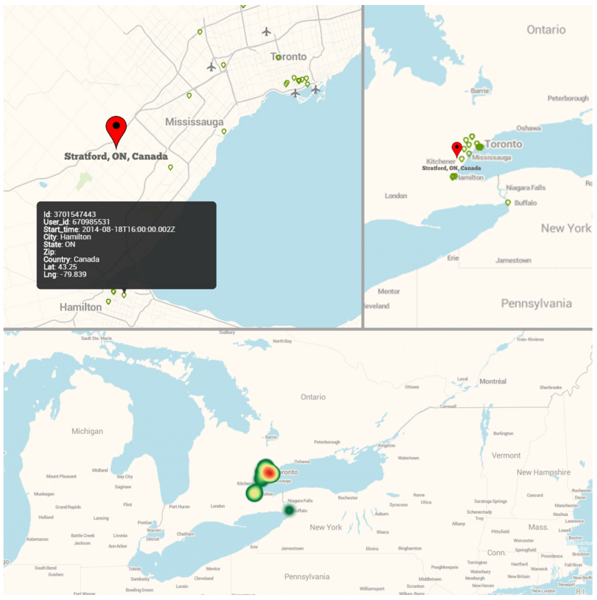
Most Popular Events. The events with maximum positive RSVPs were given higher priority.

Time to Event. The time difference between the start of the Event and the time at which the User was notified about the Event. The events with maximum Time to Event values were given precedence.

7 Recommender Engine: Examples

The above two maps indicate the top 30 recommendations made to the users from two different geographical locations. We can clearly see that the recommendations made to the users lie in the


Id: 695149490
User_id: 3410601793
Start_time: 2014-10-31T13:00:00.003Z
City: Medan
State:
Zip:
Country: Indonesia
Lat: 3.586
Lng: 98.68


Id: 3701547443
User_id: 670985531
Start_time: 2014-08-18T16:00:00.002Z
City: Hamilton
State: ON
Zip:
Country: Canada
Lat: 43.25
Lng: -79.839

same or nearer to the user's geographical location proving the authenticity of the built personalized events recommender engine.

8 Conclusion and Future Work

By solving the Events Recommendation problem, we demonstrated that the explored problem is quite different from existing traditional recommender systems. Some of the options to ease this issue were the utilization of LAMP paradigm and RSVP information. We explored quite a number of classifier models by leveraging the essential features of the Events Based Networks that can influence the design of personalized recommender engines.

In future, we plan to research the impact of group memberships on the User's interest to attend events. We would like to consider the Availability of Event Tickets to build the model. Trending events can viewed as one of the most important factors to make recommendations. Considering Time to travel factor is also one of good directions to steer the focus in the solving the problem.

9 References

1. E. Minkov, B. Charrow, J. Ledlie, S. Teller, and T. Jaakkola. Collaborative future event recommendation. In Proceedings of the 19th ACM international conference on Information and knowledge management, CIKM '10, pages 819–828, New York,

2. X. Liu, Q. He, Y. Tian, W.-C. Lee, J. McPherson, and J. Han. Event-based social networks: linking the online and offline social worlds. In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '12, pages 1032–1040, New York, NY, USA, 2012. ACM

3. T. D. Pessemier, S. Coppens, E. Mannens, S. Dooms, L. Martens, and K. Geebelen. An event distribution platform for recommending cultural activities. In WEBIST, pages 231–236, 2011

4. T. De Pessemier, J. Minnaert, K. Vanhecke, S. Dooms, and L. Martens. Social recommendations for events. In CEUR workshop proceedings, volume 1066, page 4, 2013.

5. H. Khrouf and R. Troncy. Hybrid event recommendation using linked data and user diversity. In Proceedings of the 7th ACM Conference on Recommender Systems, RecSys '13, pages 185–192, New York, NY, USA, 2013.

6. https://spark-summit.org/2014/wp-content/uploads/2014/07/Scalable-Distributed-Decision-Trees-in-Spark-Made-Das-Sparks-Talwalkar.pdf