University of Eastern Finland

School of Computing

Master’s Thesis

Activity events recommendation

Abu Saed Mohd Sayem

ABSTRACT

A recommendation system is a subclass of an information filtering system where an item is recommended to a user. Items could be movies, restaurants and garments. It helps users quickly select information that is relevant to them. The purpose of using a recommendation system is to filter out information that is not relevant to users and to list all the information that is relevant to the user's profile. Search/retrieval are the normal solutions. Users can still be overwhelmed by this amount of information. To filter out the excess of information, recommendation systems need to be in place to filter out different items or information that is of interest to the users.

The aim of this study was to create an activity event recommendation system. In this study, an activity is an event involving user movement which could be walking, running, skiing, cycling, bus, train or car. The primary aim of this study was to add a recommendation system to MOPSI[1]. MOPSI is an application that helps users find who and what is around them. It allows users to find different services near them. It also allows users to share photos, trajectory tracking and chatting with other users. With all these features, MOPSI has unique data that can be utilized to create an activity event recommendation system.

A survey was also conducted for most active users in MOPSI. For this purpose, 150 activity events were created. The most active users then answered *Yes* or *No* to each of the 150 activity events. This indicates if the user would attend the event or not. With these answers, the ground truth for each event-user pair is obtained.

With each user's data in MOPSI, user profiles were created from trajectories, meetings, photos and services visited by the user. These user profiles combined with activity events and the ground truth obtained from the survey is used to created nine-dimensional feature vectors. The feature vectors have been split 50-50 for training and testing. Different classifiers were used to create models and predict if the users would participate in an event. The average accuracy that has been obtained is 80 %. This is then used to recommend events to users or predict events that are of interest to users.

Keywords: recommendation system, interest prediction, classifiers, location-based user profiles

[1] http://cs.uef.fi/mopsi

ACKNOWLEDGEMENTS

I would like to give my sincere thanks to University of Eastern Finland for accepting me to the IMPIT program. I would also like to thank all the professors who helped my gain knowledge in different areas of computer science. I have learned a lot during my studies at University of Eastern Finland.

I would also like to give my deepest gratitude my supervisor, Professor Pasi Fränti for his guidance during my study. His observation on my study and IT project has allowed me to move forward with my skills and knowledge in computer science. I would also like to express my deepest gratitude to Dr. Radu Mariescu-Istodor who has answered a lot of technical questions that I had during this study.

I would also like to express my deepest gratitiude to my sister for moral support and encouragement. I would like to thank my family who also supported me.

Table of Contents

1. Introduction

Mobile devices are being used everywhere. Most of these devices use *Global Positioning System* in them (GPS). With GPS, it is possible to know the geographical position of mobile devices. Since there is access to geographical information of mobile devices, it is possible to conduct useful research on such data.

There is a lot of data moving around the internet. People around the world use computers and mobile devices to communicate with each other, to buy or sell products online and to carry out numerous different tasks. Internet can now be thought of as a social network connecting people and organizations [1]. People are confronted with information overload where they are facing large amounts of data. This amount of data needs to be filtered in a personalized way for user to navigate through all of it. Numerous different solutions are suggested to tackle this problem with recommendation systems being one of them [2].

Since most mobiles use GPS, there is a lot information that can be utilized, and different services can be made of them. One of them being *location-based services* (LBS). These services include applications for travel, navigation and games. Famous travel applications include TripAdvisor and Foursquare. For navigation, applications such as Google Maps is used. Games such as PokemonGo and O-Mopsi [6] utilize GPS data.

Before discussing about the details of the recommendation system for events, it is best to describe how a recommendation system works and how others have used methods to implement different recommendation systems in existing literature. Recommendation system is a form of information filtering system used for recommending items to a user which is personalized [3]. Most recommendation systems fall under two different classes: collaborative filtering method and content-based methods.

*Collaborative filteri*ng method is used for recommending items or to predict a certain item for a particular user based on user's previous history and other similar or like-minded users [4]. *Content-based filtering* methods recommends an item to a particular user based on the item's description and the user's profile history.

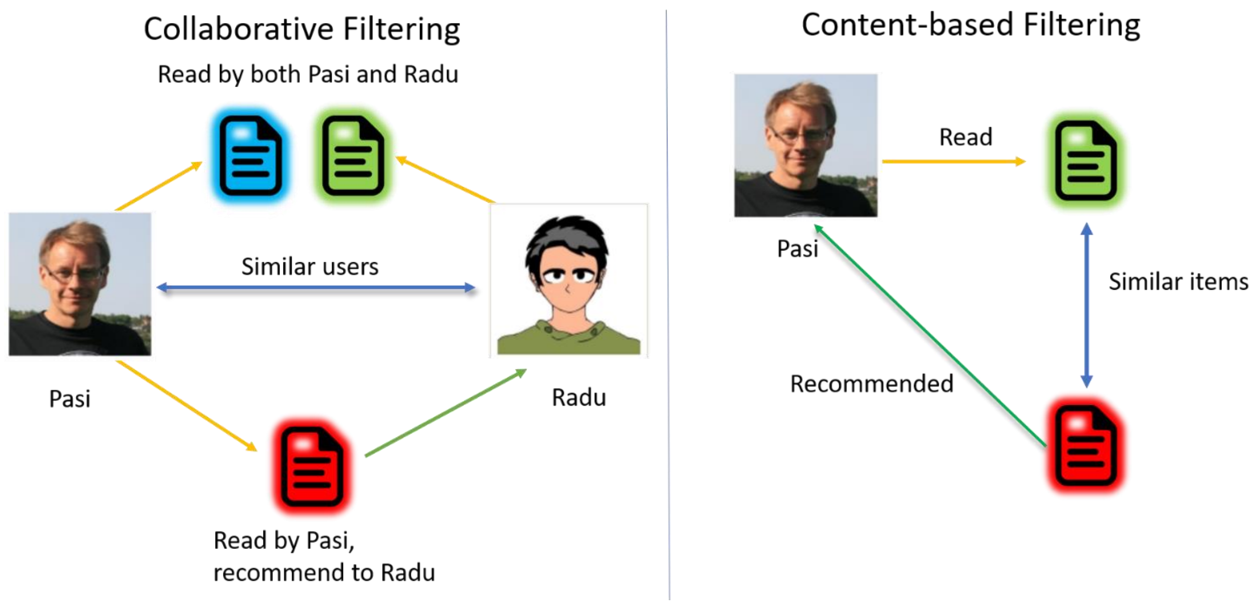


Figure 1: Collaborative filtering and content-based filtering recommendation methods

Figure 1 shows the abstract view of how collaborative and content-based filtering methods work. Pasi and Radu are active Mopsi users. Recommendation systems are widely used in online services. They originally appeared in books, movie and shopping. Users in e-commerce sites click on different products of interest, the recommendation system then can start recommending similar products to that user. Similarity of the products are measured on the content, but also other users with similar profiles. Recommendation systems for LBS also started to appear when GPS became available to consumer devices. There is also a lot of studies that have researched into building recommendation systems in LBS. These will be discussed later in this chapter.

One of the key problems in location-aware recommendation systems is to find out what is relevant to users. This was one of the key problems faced in this study. This will be reviewed next on existing literature how this problem is tackled by others on location-aware recommendation systems.

In one of the papers, relevance is stated by content, location, time and social network [7]. Content could be considered as the most relevant aspect in recommendation systems. Search engines can be recommendation systems. With a given user-keyword, the engine will match this to the content of a webpage. Location data can be leveraged in search engines as well. In [8], the authors show a way to extract address from the content of the webpage. Then the address can be tied to the location using a *gazeteer* that geo-codes the address. The location is now known and can be used in ranking the search. Recommendations system do not have keywords and must look for alternative ways for relevant content.

In [9], the authors implemented a recommendation system which uses a user’s past location history, preferences and recommends nearby shops. The system analyzes a user’s frequently visited shops and build recommendations by using the shops as items in collaborative-filtering algorithm. The authors implemented a place learning algorithm at the core of the system. The algorithm automatically finds the user's most frequented visited shops. The learning algorithm uses the unavailability of GPS signals to detect if the user has gone indoors. Following this, the system plots a list of frequently visited shops for each user. When the user requests for recommendations, the list of shops for the user is used. Similarity is calculated between the shops where the rating of the shop by the user and frequency of visits is used. The shops are then recommended based on the user's location. The system divides the map to fixed areas. Then a first order Markov model is used on the user's movement with areas being the nodes. Probabilities are then calculated from plotted user's location. Higher probability of some nodes indicate that user would move to those areas.

In [10], the authors implemented their method in a leisure-guide system. This predicts user's future leisure activity. It infers a user's activity from context and patterns. All this is done without the user issuing any query in the system. Their system uses eight different models. Collaborative-filtering is one of the models that is used. A content-preference models is also used which measures the similarity of an item to a user's profile such as previously viewed webpages and documents. Another model that is used is that the system adds more weight to places that are with a distance threshold. These are some of the models that are used in the leisure-guide system. The items with the highest scores scored by these eight are models are then recommended to users. This was then tested in the urban setting of Tokyo.

In [11], a restaurant mobile application has been developed by the authors to give location-based recommendations to it users. They have designed a multi-dimensional collaborative filtering algorithm for personalized recommendation. They have used information such as user's comments and tags to recommend items to the user. The authors have calculated a weighted average of votes from other users on an item which is a restaurant. In [12], the authors present their location-aware recommendation system that uses user ratings to recommend items. They have also used collaborative-filtering algorithm as their primary recommendation system using the ratings of users for recommendation items. The system produces recommendations based on spatial regions. If a user is in spatial region, the system will recommend items in that region. They also use a technique that they call a travel penalty algorithm. The further a user is away from recommendation items; the more weight or penalty is added to those items. This is done so that the system doesn't query all the items in the database. In [13], the authors propose a recommendation system to recommend friends to a user based on user's location history. The authors have proposed to use a user's location history as their implicit ratings on regions. They also propose a similarity measure between users based on their location history. They also estimate an individual's interest to an unvisited region by considering of that user's location history and other similar users. They propose measurement which produces a hierarchical graph-based method which models a user's location history. This results in a graph containing nodes. The similarity between users are calculated by checking the nodes that are shared between those users. Friends are then recommended to users who have high similarity scores with each other. Spatial regions that are unvisited by a user is recommended using the similarity scores. A user who is highly similar to a user and has visited the unvisited region is recommended to a user who hasn't.

In [14], the authors propose a recommendation system which builds user preferences by mining the user's social network profile. The system uses a content-based filtering algorithm approach. The items that are used is a list of restaurants. Restaurants are recommended to users which are accessible to them. Accessibility is defined by how fast the user travels or what mode of transportation is used which is automatically detected using decision trees followed by a hidden markov model

Some recommendation systems also use GPS trajectories as an input. In [15], the authors extracted attributes such as elapsed time, stay time and social aspect from the trajectories to recommend itineraries. This is done by extracting a graph based on user's trajectories. Stay points are first extracted which are then clustered which become locations that can be included in itineraries. Certain attributes are then calculated to determine location interest such as elapsed time and stay time. A graph is then built using these location clusters as vertexes and edges being the travelling time between vertexes. When a user requests for recommendations, the system uses this graph and the user's current location to recommend itineraries.

In [16], tourist locations are recommended to users. It is based on user's visiting history in geographically remote region. Regions which are similar to user's location history is recommended to them. Similarity between regions are computed using a set of geo-tags from some users. The most important locations for a user are computed using the user's geo-tags. Recommendations are made to a user by retrieving the most important locations of the user and then retrieving the similar regions.

Mopsi already has a recommendation system in place for recommending three types of items: services, photos and GPS trajectories [17]. GPS trajectories are also used in the recommendation system that is implemented in this study. Mopsi services are shops are stored in the local database.

However, recommending activity events is different from recommending books, movies and services. Traditional recommendation system relies on some form of feedback on items. Events happens in the future and events might not have any previous attendance data. Further literature will be investigated on this topic. In [18], the authors address the problem of no previous feedback on items. They recommend academic scientific talks or lectures to different users which are events that happen in the future. To reduce the need for explicit user feedback, the authors suggest an approach for implicit user feedback on previous events which are similar to future scientific talks. Their approach predicts how much a user can be interested in a scientific talk that will happen in the future. The system uses similarity of users based on their explicit feedback on past events.

This study also recommends future events, but they are moving events which have attributes like type of movement, speed and distance which is different because of these properties. In [19], Bayesian probabilistic model is used to recommend events that uses user's history of participation events. They use the social relations between users and implicit feedback to recommend events.

Cold-start problem is an issue when there is no past data about an item or a user. Then, it is difficult to recommend items to users. In [20], the system gives recommendations based on time and geographical preferences. They address the cold-start problem by exploiting group-memberships of users. They also propose that the relevance of a new event to a user can be modeled by getting the density estimation of the user in the region where the event is being held in.

In [21], the authors propose a personalized event recommendation system using ontology and spreading algorithm. The ontology results in a network with semantic concepts with relations between them. This results in a domain knowledge model. The spreading algorithm is used on the domain knowledge model to learn user behavior patterns by finding the user's interests. The model proposed by the authors uses the user's information, location, time and date. In [22], the authors system recommends events in social web by taking into three additional features into account. The first feature is to determine if an event is reachable or not. The second feature is to determine the reputation of an event. The third feature is to determine the list of attendees who are the user's friends. They use a scoring function which predicts the interest of a user in an event. The scoring function uses the event's location, ratings, confirmed participation and social interest. It also uses keywords from the theme of the event. They also construct different user profiles which contain keywords from the past events that indicates the general interest of each user. In [23], the authors have used a Bayesian network inference model to calculate the probability of the future behavior of the user to give push recommendations to the end user. In addition to using time, location and user behavior, they have also used environmental factors.

In [24], the authors have calculated the similarity of the users based on location history and social network. The similarity between users are measured mainly based on two parameters. The first parameter is the similarity between two users according to similar pages liked by both users in Facebook. The authors have used Jaccard coefficient with the number of pages liked by both users divided by total number of pages liked by both users. The second parameter used is the similarity in location history between users. In Mopsi, users have geo-tagged photos and trajectories. This is used to find how similar two users are in terms of their location history. This results in a frequency histogram. The similarity is then calculated using Bhattacharyya distance between the two histograms.

In [25], the authors have calculated the frequency of photo taking and other activities in pre-defined service locations to find similarity between users. User's histogram is first calculated. The activity points of a user are mapped to a nearby place. The activity points increase the histogram bins which denoted as places in the system. The next step, the authors use Bhattacharyya coefficient to calculate the similarity or distance between histograms which are statistical population.

The aim of this study is to create a recommendation system of activity events into Mopsi. Events defined here in this study is a form of moving activity. Mopsi has a set of data that can be used to create a unique recommendation system. The events that are recommended to users can also be used to predict events that are of interest to users. The data in Mopsi contains trajectories, photos, meetings records between users and much more that could be utilized. For this purpose, a web application was created to implement this recommendation system.

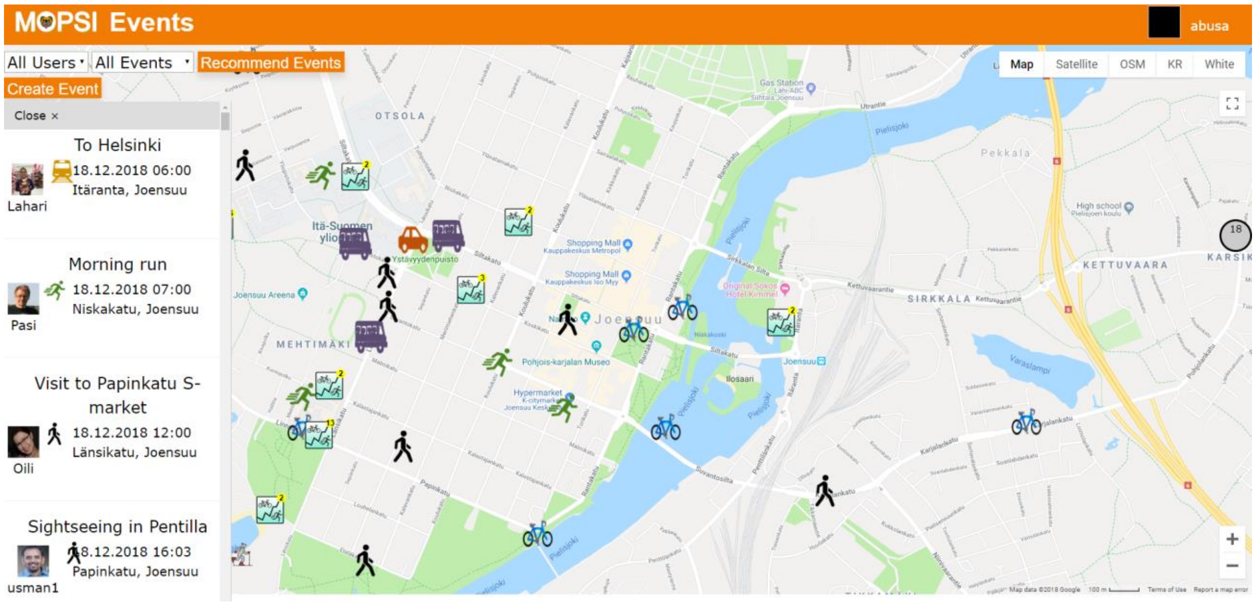
2. Mopsi event system

Mopsi is a social networking platform developed by the machine learning lab at the University of Eastern Finland (UEF). It helps users to discover who and what is around their location. Some of its features are route tracking, photo sharing, bus timetable and service recommendation.

The event system which is a sub-part of Mopsi, is the author's contribution as an IT project. Before describing the Event system, it is appropriate to give brief description of what is an event in the context of this study. *Event* is an activity of user movement which can be walking, running, cycling, driving, taking the bus or a train.

The event system can be used to schedule different activities. It can also be used as ride-sharing platform. It can be seen on Figure 2. An event has the following properties:

1. Title
2. Description
3. Distance
4. Time
5. Speed
6. Date
7. Time
8. Address
9. Route (Optional)



To Helsinki
18.12.2018 06:00
Itäranta, Joensuu
Lahari

Morning run
18.12.2018 07:00
Niskakatu, Joensuu
Pasi

Visit to Papinkatu S-market
18.12.2018 12:00
Länsikatu, Joensuu
Oili

Sightseeing in Pentilla
18.12.2018 16:03
Papinkatu, Joensuu
usman1

Figure 2: Mopsi event system

Figure 2 shows the event system once the page loads. It shows the events both on map which and a list on the left so that the user can quickly select an event.

2.1 Components of the event system

This platform can be considered as a single-page application as it does not redirect to any other pages. The system consists of multiple parts. Figure 3 shows the different components.

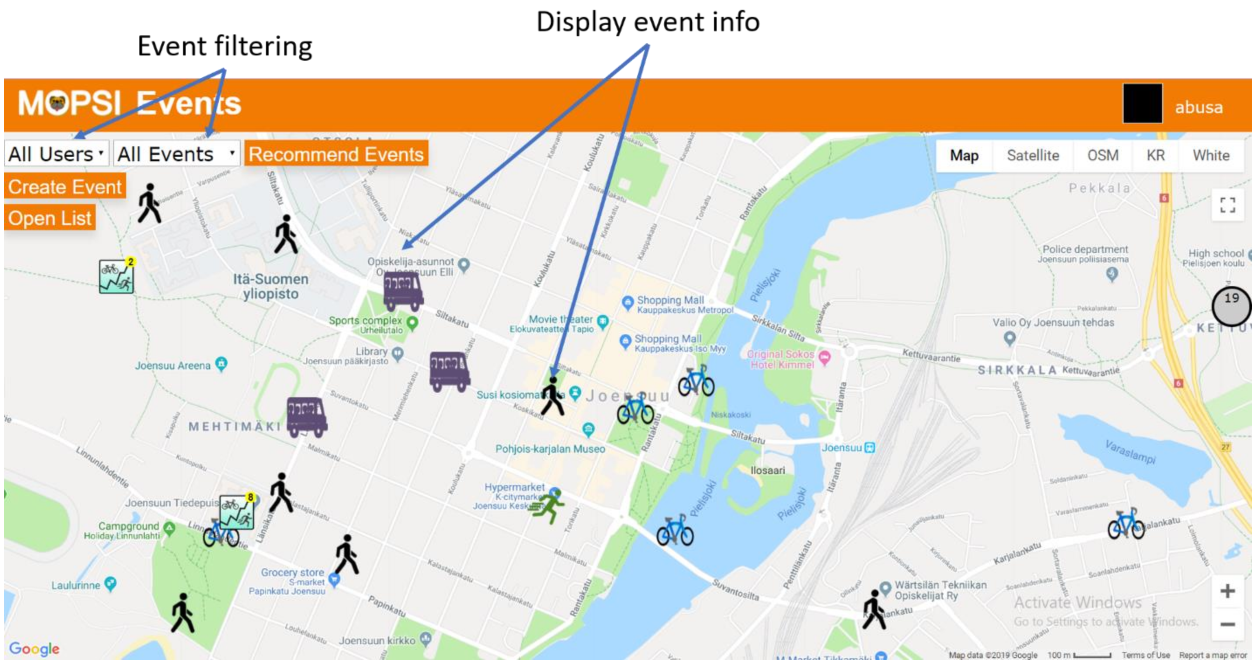


Figure 3: Parts of the event system

2.1.1 Event filtering

An example of event filtering can be seen on Figure 4. The user can filter events based on three categories. They are:

1. Events which are created by a single user.
2. Events according to time which are *today*, *weekly* and *monthly*.
3. The above two categories as a combination.

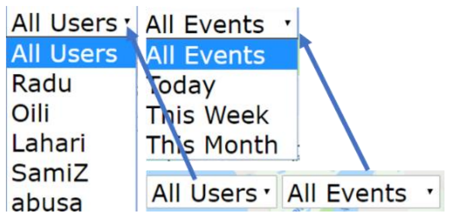


Figure 4: Filtering based on user and time

2.1.2 Display event

The user can click on any event and details are displayed: creator, date, address, participants, type of event and a route if it exists. Figure 5 shows what happens when a user clicks on the event. It shows the route associated to the event.

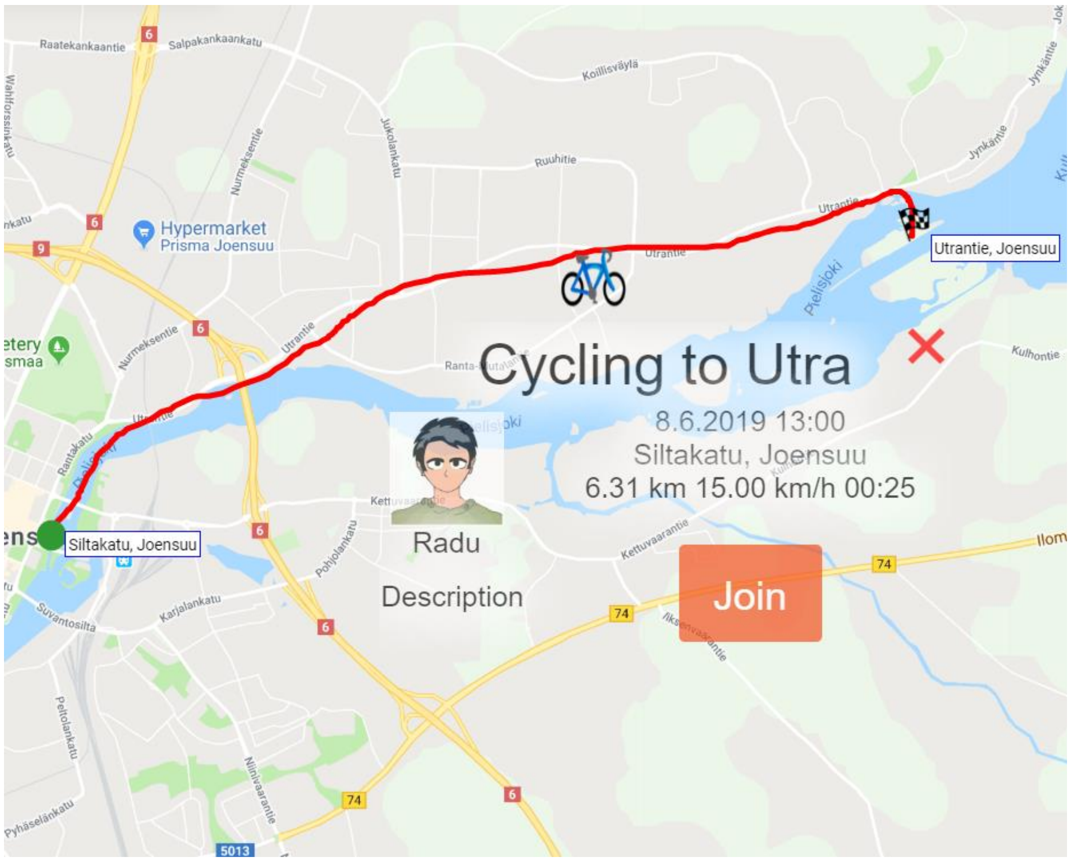


Figure 5: Display cycling event created by Radu, which starts from Siltakatu, Joensuu. Nobody has joined the event yet. The planned route is also shown.

The planned route can be drawn by dragging with mouse on the map. It can be further tuned by *trajectory snapping* which will be explained in detail in section 2.4. Figure 6 shows a demonstration of clicking the route to match it to the road network. *Trajectory snapping* off means that the system shows the route which is originally drawn on the map. *Trajectory snapping* on shows the route which is map matched to the road network.

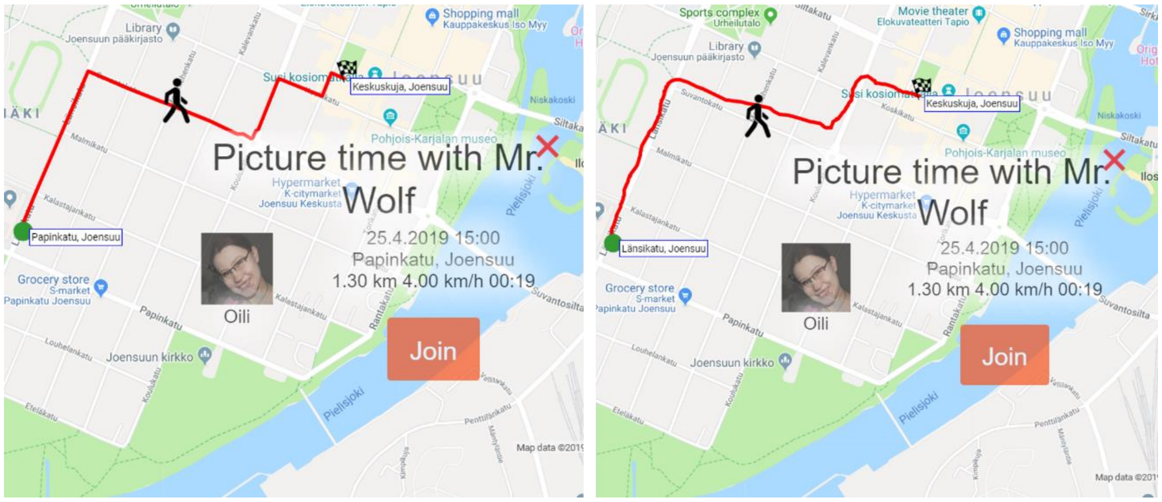


Figure 6: Trajectory snapping on/off.

Figure 7 shows how users can join an event. The user can also cancel his/her participation later if so decided. The user does not need a Mopsi account to use this application. The user can just enter the name that he/she wishes to use. This is demonstrated in Figure 8.

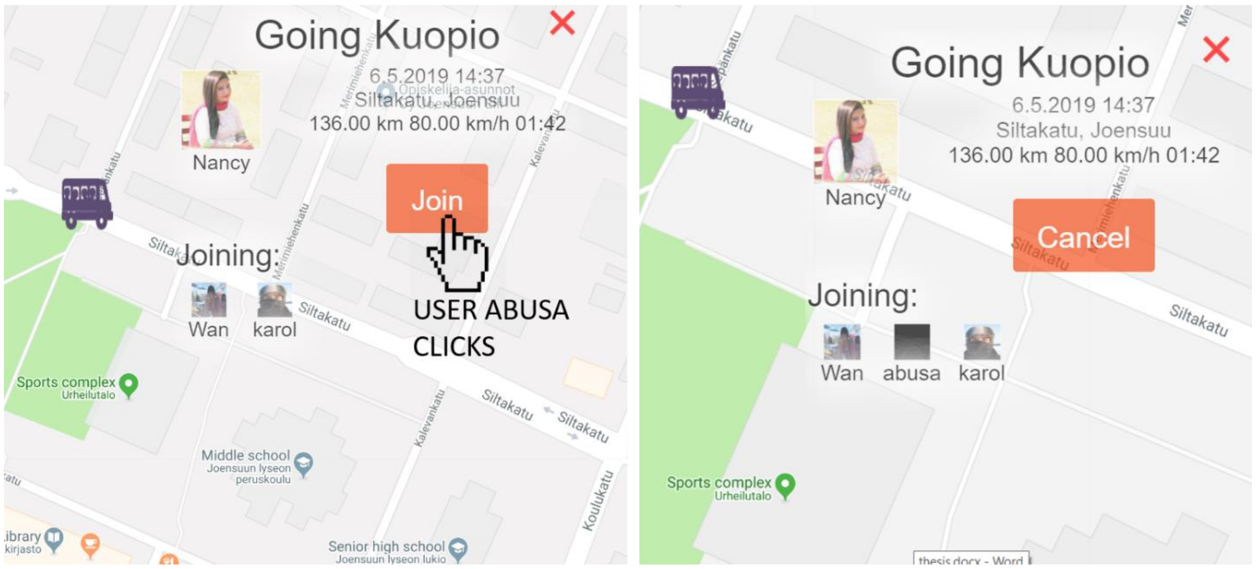


Figure 7: Join and cancel event.

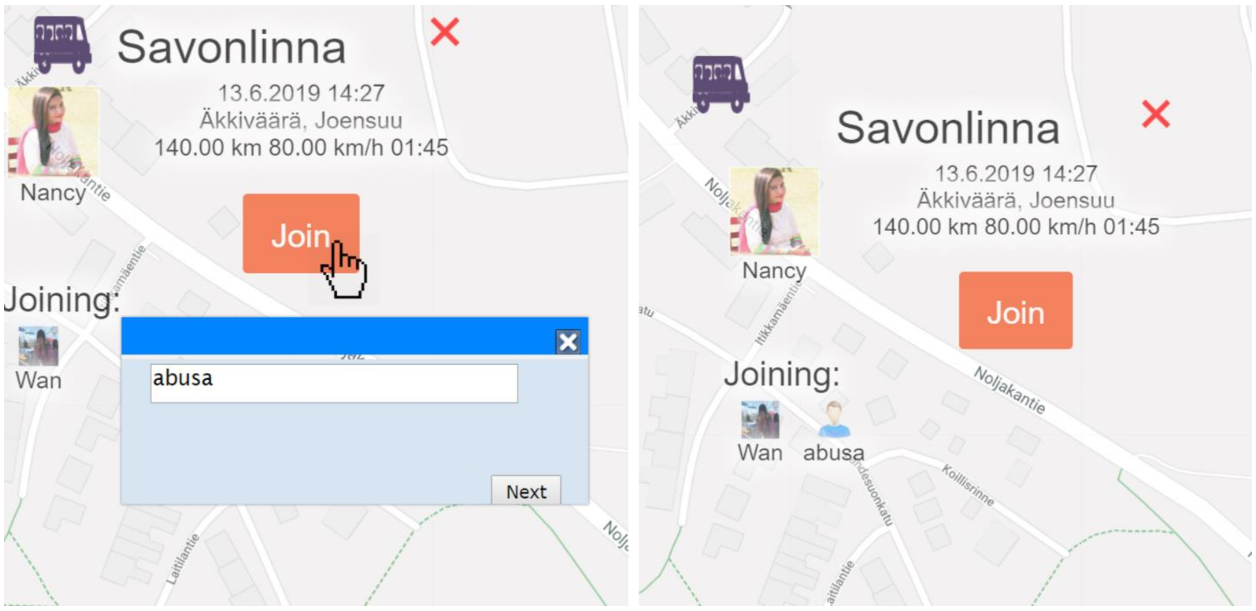
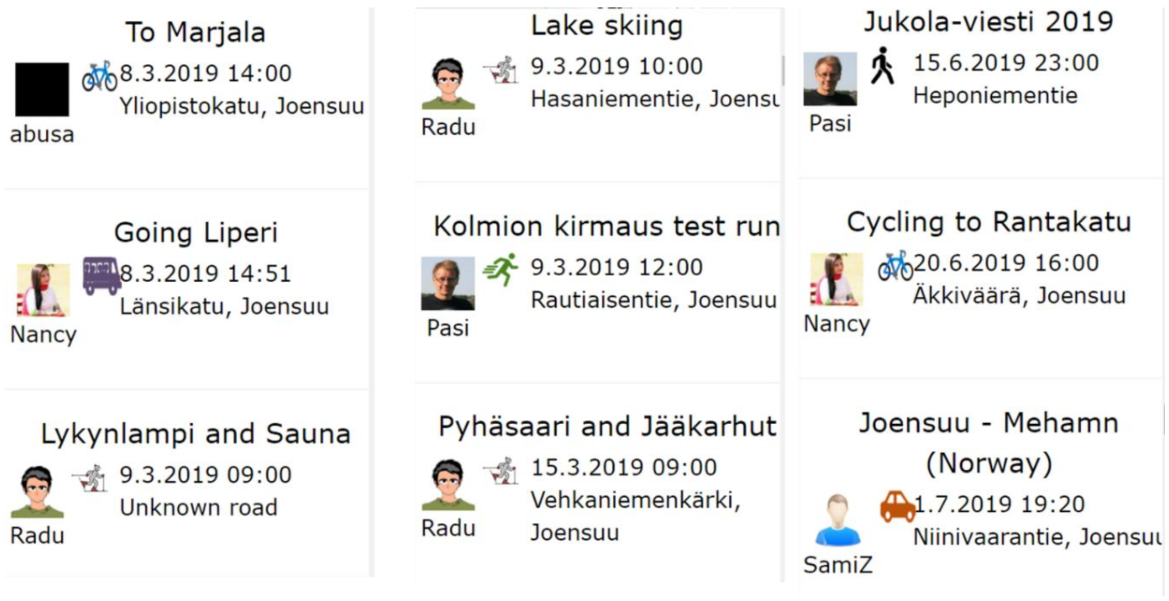


Figure 8: Joining event using guest account.

2.1.3 Event list

The user can also view events in the form of list by clicking on the button *Open List* and toggle between all of them. A demonstration of this is in Figure 9.



To Marjala
8.3.2019 14:00
Yliopistokatu, Joensuu
abusa

Going Liperi
8.3.2019 14:51
Länsikatu, Joensuu
Nancy

Lykynlampi and Sauna
9.3.2019 09:00
Unknown road
Radu

Lake skiing
9.3.2019 10:00
Hasaniementie, Joensu
Radu

Kolmion kirmaus test run
9.3.2019 12:00
Rautiaisentie, Joensuu
Pasi

Pyhäsaari and Jääkarhut
15.3.2019 09:00
Vehkaniemenkärki, Joensuu
Radu

Jukola-viesti 2019
15.6.2019 23:00
Heponiementie
Pasi

Cycling to Rantakatu
20.6.2019 16:00
Äkkiväärä, Joensuu
Nancy

Joensuu - Mehamn (Norway)
1.7.2019 19:20
Niinivaarantie, Joensuu
SamiZ

Figure 9: List of events. It contains nine events from five users. The events include one bus and one car trip, three skiing events, two bicycle events, one running and one walking event.

2.1.4 Event creation

The user can create an event in this application by clicking on the button *Create Event*. A pop-up window is then displayed. Figure 10 shows the pop-up.

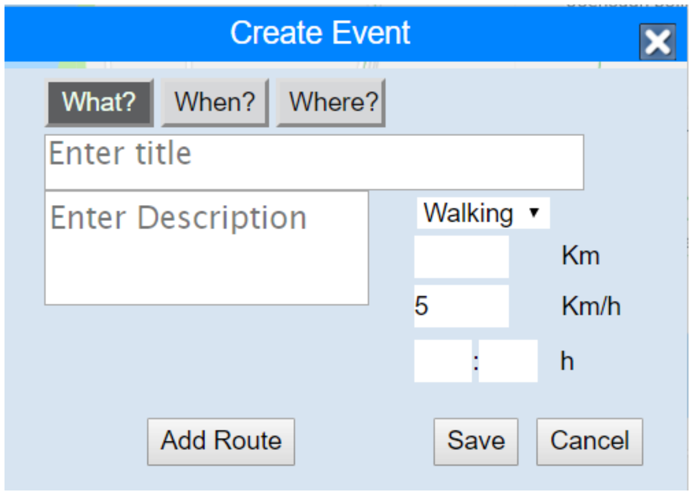
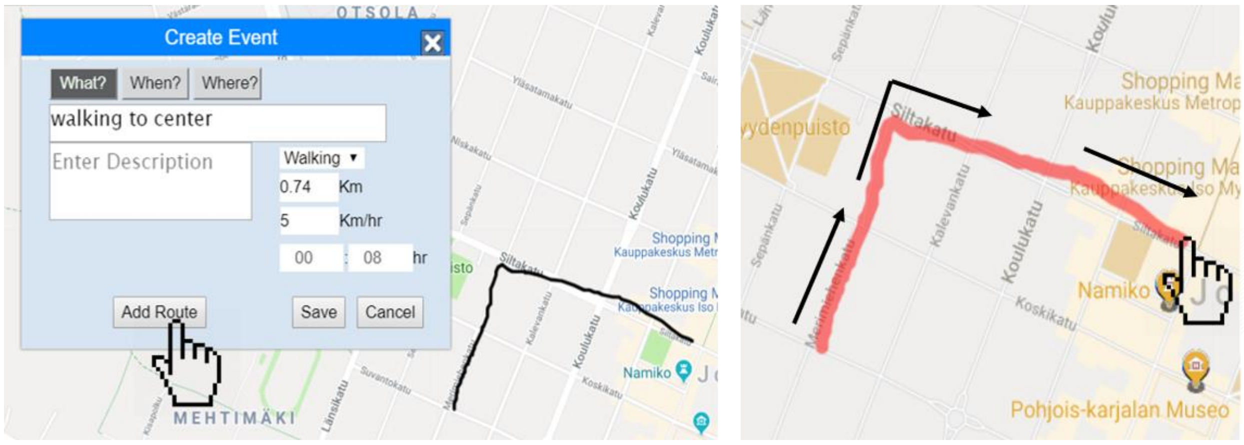


Figure 10: Create event pop-up.

In the *What?* tab, the user can enter title, description, type of event and add a route. Only title is necessary for an event to be created. The user can also edit these three entries later. An example of this is that if the user choses to modify the distance, then the system will automatically update the duration and pace.

The user can also add a route for an event. This can be done by clicking on the *Add Route* button. The user first must draw a route on the map by dragging the mouse. The resulting route is then displayed on the map. The user can choose to redraw if he/she is not satisfied with the drawing. This is demonstrated in Figure 11, which shows the route which has been drawn by the user. The statistics related to this route is then automatically calculated.

The *When?* part of this window allows the user to pick the date and time of the event which is demonstrated in Figure 12. The *Where?* part of the tab allows the user to pick the location of the event. The selected location is then shown by a marker which is the symbol describing the type of movement. The location is placed at the start point of the event. User can drag the marker and the address of the event is automatically updated.


Create Event

walking to center

Enter Description

Walking

0.74 Km

5 Km/hr

00 : 08 hr

Figure 11: Drawing route on map.

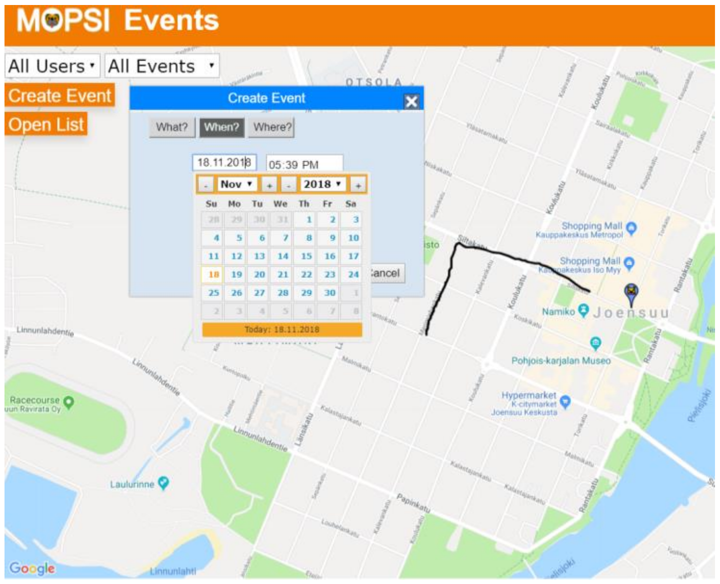


Figure 12: Date and time selection of an event.

2.2 Clustering map

The clustering map was developed in the machine learning lab in University of Eastern Finland. For the event system, the component described in [27] was used. It is a data visualization tool for data points on a map. It uses a grid-based clustering method to cluster data points on a map. There are three main steps involved. There is grid construction, initial clustering and merging the clusters. The first step involves creating rectangular grid cells. Then the initial clusters are made by assigning the data points to the cells. Each cell is an initial cluster. The final clusters are then formed by merging neighboring cells according to some criterion which is closeness of the cells. Selecting a cell size is not trivial here because if the cell size is small, it leads to more computations. For the clustering of the event system, size was determined as to what size of the icons should be shown on the map.

The clustering of the data points helps the events system with the clutter problem. The *clutter problem* is a situation when there are too many data points on the screen making it over-crowded. An example of the clutter problem can be seen in Figure 13 on the left. If the clustering API is not used, the system would overwhelm the user with different icons on the map. The clustering can be observed in Figure 13 on the right. Using clustering, it becomes easier for the user to navigate through different activity events.

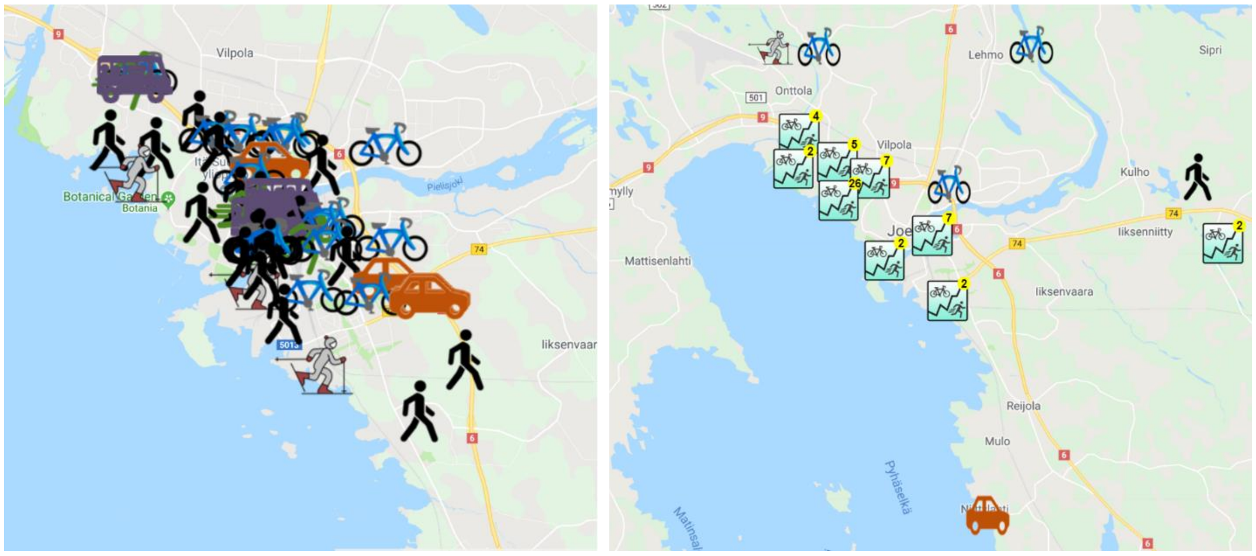


Figure 13: Event system without (left) and with clustering (right).

As it can be seen in Figure 13 on the left, if no clustering was used, it would overwhelm the user with a lot of icons. I contributed a bit to this tool. The tool was using static images for icons. It was using static images also for displaying the number of data points in a cluster. If others want to use the clustering API, they do not have to manually change the source of the images to display. Since this was implemented, there can be unique icons for each data point. I have also made the API a bit more modular. The old clustering API[2] and modified clustering API demo[3] is available. Figure 14 shows the icons that are used in the system to display the events for each movement type.

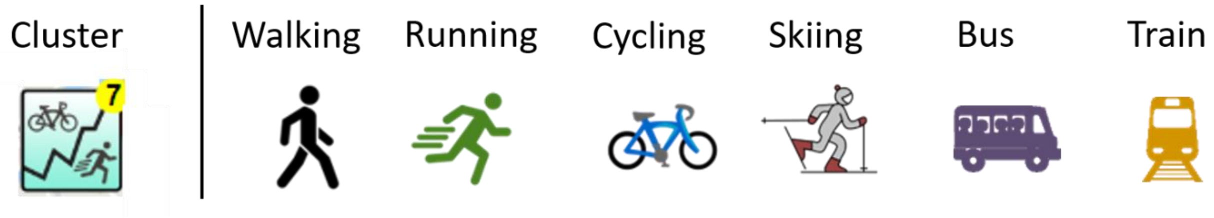


Figure 14: The cluster icon (left) indicates that there are seven events within the cluster. The individual icons that are used for different movement types are shown on right

2.3 Trajectory snapping

A trajectory snapping algorithm has also been implemented in the system. *Trajectory snapping* is matching a trajectory to the road network on the map [28]. Route points are a set of spatio-temporal points of *x*, *y* and *t*. A point (*x*,*y*) indicates a moving object with geographical location at time *t*. This can be corrected by matching the route point on to the road network to correct this error. However, this is not the reason why trajectory snapping is used in the events system as the route is not recorded during the event. It is used for the usability purpose so that users can see what route to follow during the event. A user drawn route and trajectory snapped route in Figure 15 and Figure 16

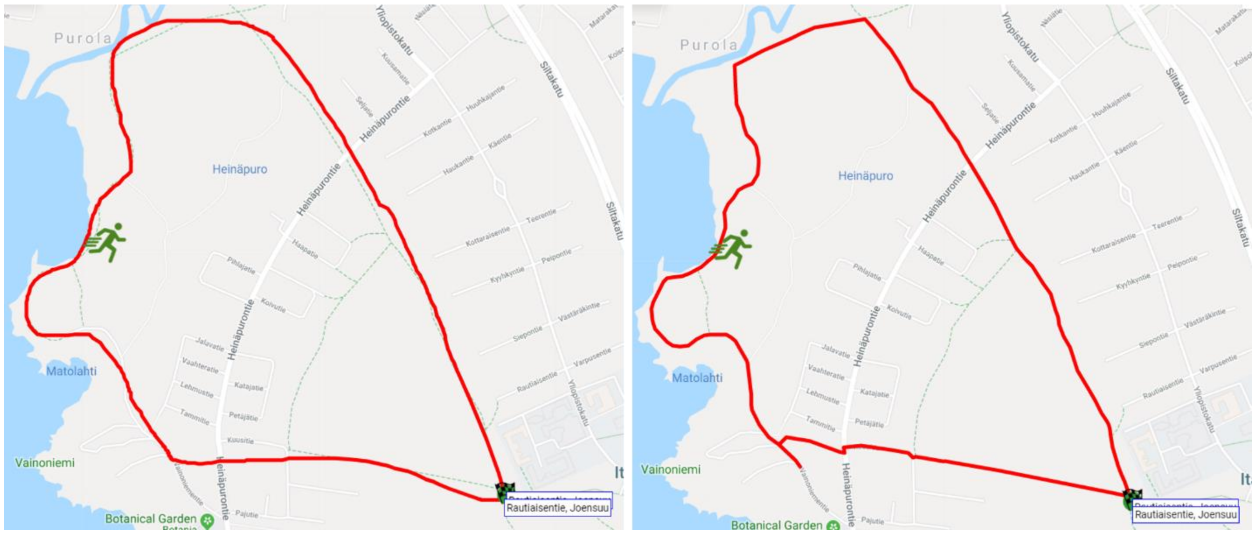


Figure 15: User drawn route on the left and the trajectory snapped route on the right

2 http://cs.uef.fi/mopsi/markerClustering/
3 http://cs.uef.fi/mopsi_dev/events/ClusteringDemo/

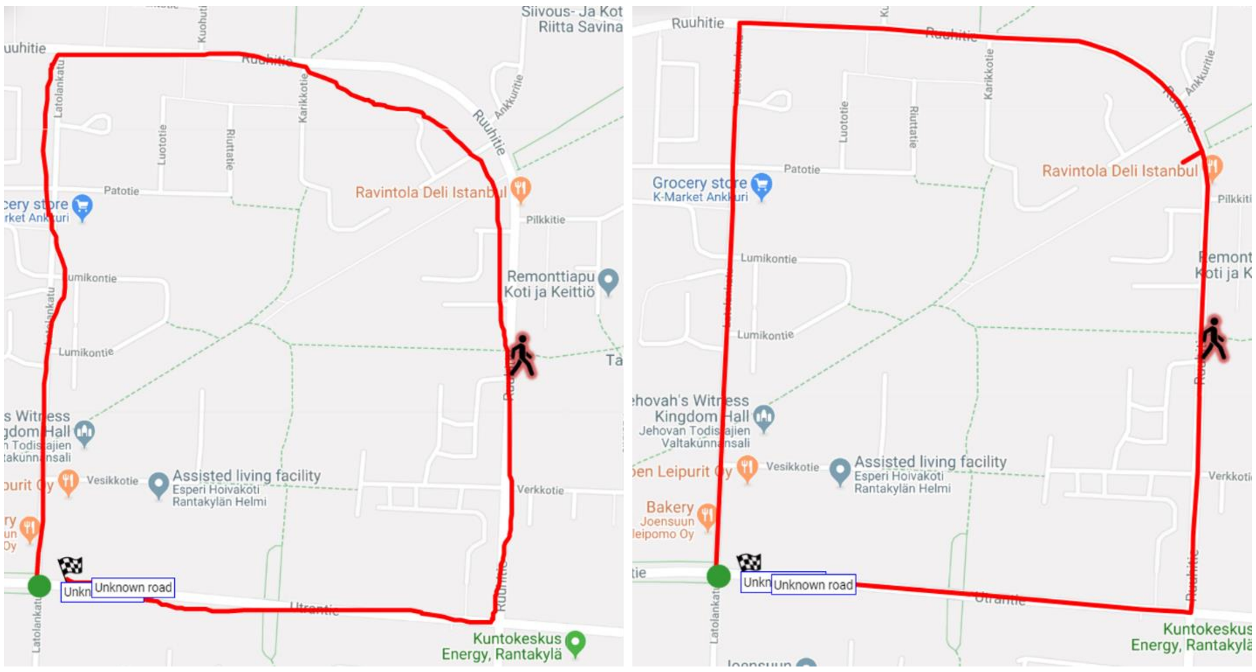


Figure 16: User drawn route on the left and the trajectory snapped route on the right.

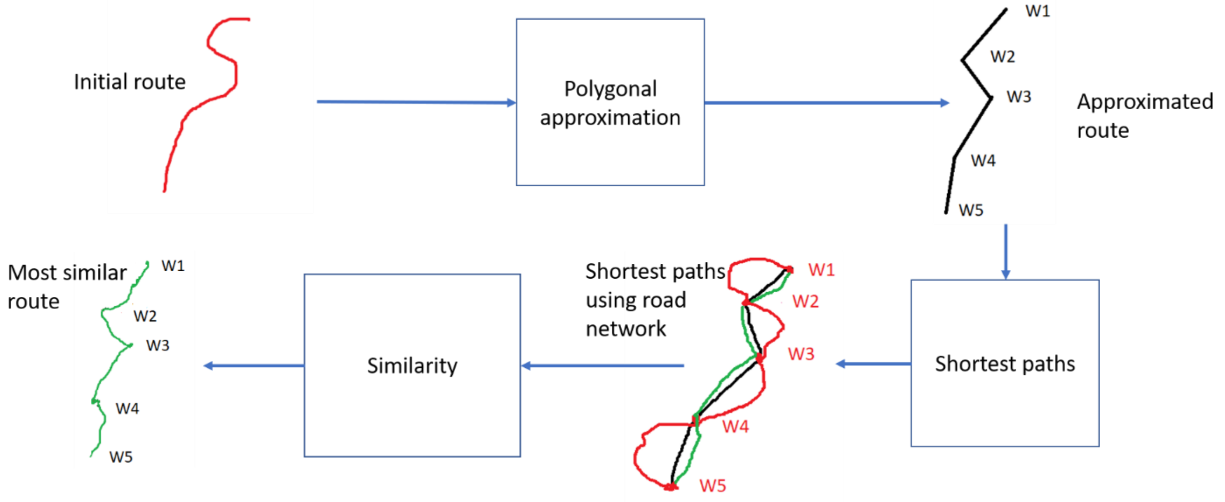


Figure 17: Trajectory snapping process

The idea for the map matching algorithm is to compute the shortest paths between the points and get the most similar path compared to the original drawing. Below are the steps that is performed by the algorithm.

Figure 17 shows how the trajectory snapping process takes place. The steps in Figure 17 are as follows:

1. Obtain user drawn route
2. Get representative points by polygonal approximation

3. Find alternative paths between each consecutive representative point.
4. Select the most similar alternative between each representative point.

In step 1, the user draws a route on the map. The route contains an ordered list of points. Google Maps API (JavaScript) is used for achieving this. It gets this list of points by tracking the mouse movements. The list needs to be reduced to proceed further.

In step 2, a route polygonal approximation method is used from [32]. This method returns reduced points by preserving the basic shape. This step returns a reduced list of points (W_1, W_2, W_3.... W_n). The third step is to calculate alternatives routing between each point in a sequence. They are calculated using Google's routes generation. One way of how to calculate the alternative routes is using Dijkstra's algorithm which finds the optimal (least cost) route or shortest route. The alternative routes are then generated by artificially inflating the cost of the edge or waypoint by adding weights to it and then use Dijkstra's algorithm again to calculate alternative routes [33]. The previous step then returns a list of alternatives between each point. Then each alternative is compared to the reduced points using a similarity method called C-SIM [26]. The alternative route with the highest similarity between each reduced point is then added to the final list of trajectory-snapped route.

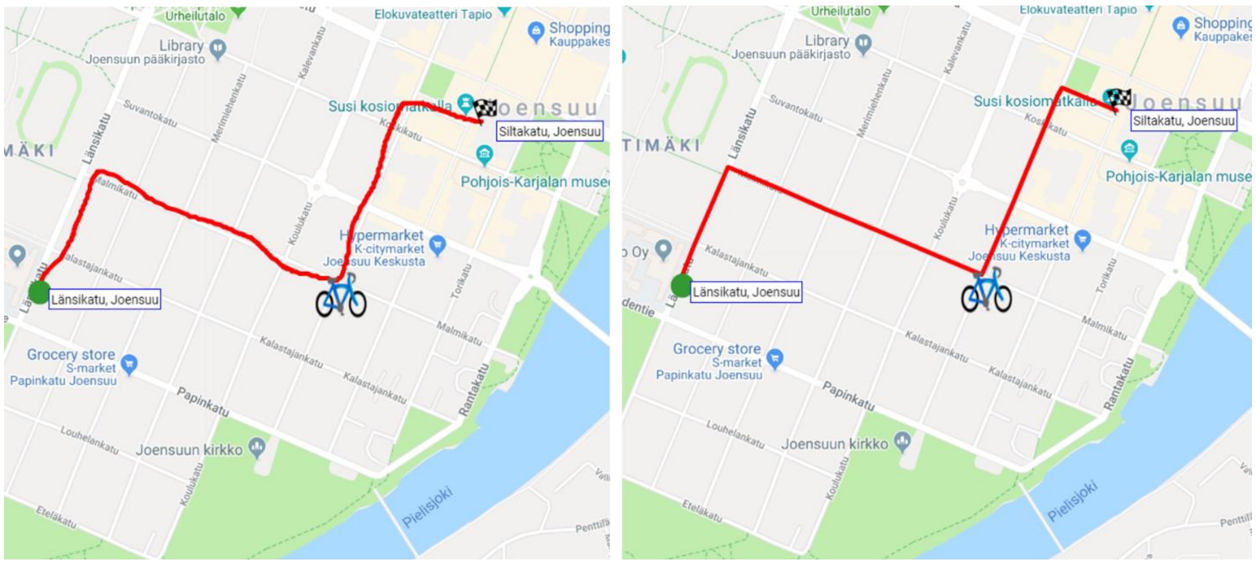


Figure 18: An example of the trajectory snapping method matching the road network as the user intended. User drawn route is on the left while trajectory snapped method is on the right.

Figure 18 shows an example of trajectory snapping method showing accurate matching. This trajectory is matching exactly as the user intended it. Figure 19 shows an example of when the trajectory snapping method is less accurate. The highlighted part oin Figure 19 shows that there is only one road on the road network. It cannot match exactly to the user drawn route on the left because there is only one road in that location of the road network. Instead, it finds the closest road to the user drawn route which is different what the user intended to use.

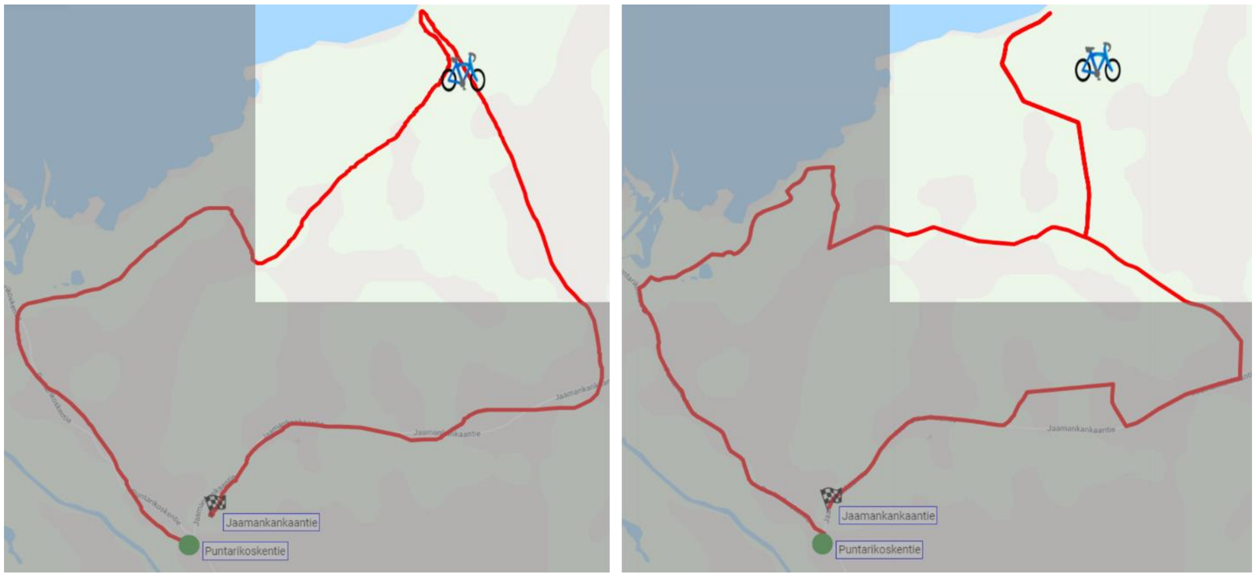


Figure 19: An example of the trajectory snapping method not working.

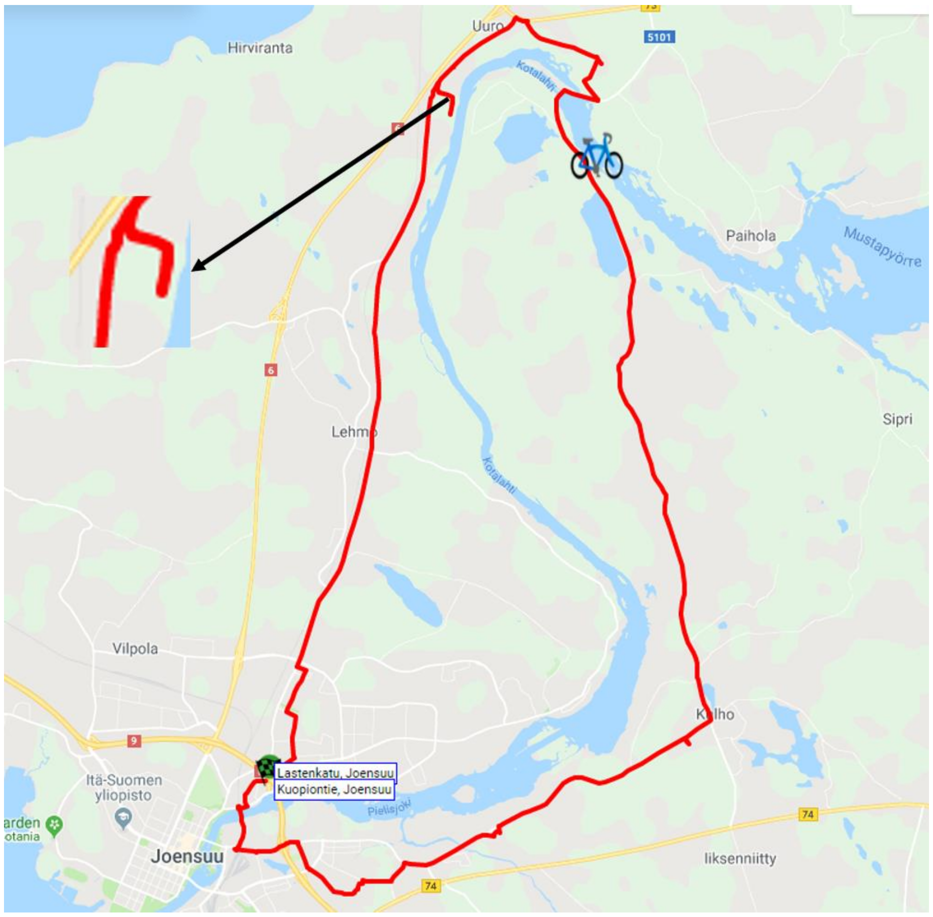


Figure 20: Artefact of the trajectory snapping method.

The trajectory snapping method works accurately and precisely with less artefacts when the users draws the route exactly on the road network. Figure 15 and Figure 18 shows examples of this. However, if the user draws the route a bit further away from road networks, more artefacts is produced. The method is not always efficient for a few reasons:

1. It also outputs some artefacts as shown on Figure 20
2. It does not work for areas where there is no road network as demonstrated in Figure 19.

3. User profiling

Before describing the recommendation system, it is appropriate to give a brief description about the Mopsi application. Using the data in Mopsi, user profiles are created. These profiles are used for the recommendation system and predicting attendance.

3.1 Trajectory statistics

In Mopsi, users can record their trajectories live when they are moving. The movement type of the trajectory is automatically detected and is detailed in [34]. The movement type is classified to four different types. They are walking, running, cycling and driving. The trajectory is first divided into segments with similar speed. A cost function is then used which minimizes the sum of the inner speed variance in all segments. A soft classification is then performed on each segment using priori probabilities which are defined in [34]. An inference algorithm is then applied using a second order *hidden markov model* (HMM). It is used for finding correlations between different trajectory segments. Unfortunately, this method does not detect skiing type although they have been recorded. Skiing trajectories are recognized as cycling, walking and running trajectories. We therefore had to extract these trajectories to classify them as skiing as follows. There are two conditions in which these trajectories could be classified as skiing. They are:

1. Skiing in Finland is usually done in December, January, February and March. Trajectories should be recorded in these months
2. The trajectories are then compared to an online skiing track using similarity method between trajectories described in [26]. A trajectory is compared to known skiing routes in *OpenStreetMap*. A threshold of more than 75 % is used to compare the skiing track to the user's trajectory.

In Mopsi, a trajectory is classified as skiing if both these conditions apply. This data may not be 100% accurate since sometimes skiing in Joensuu are can also be done during April. But since skiing can be mostly done in those four months mostly, we will limit to those months only.

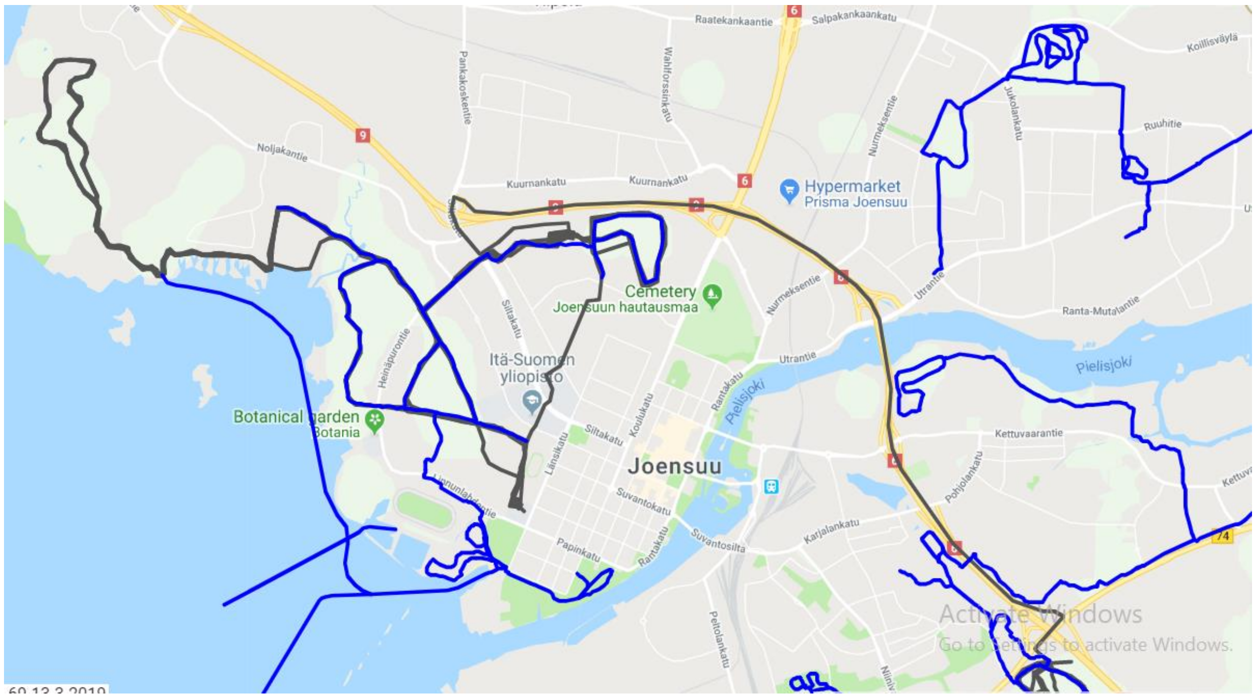
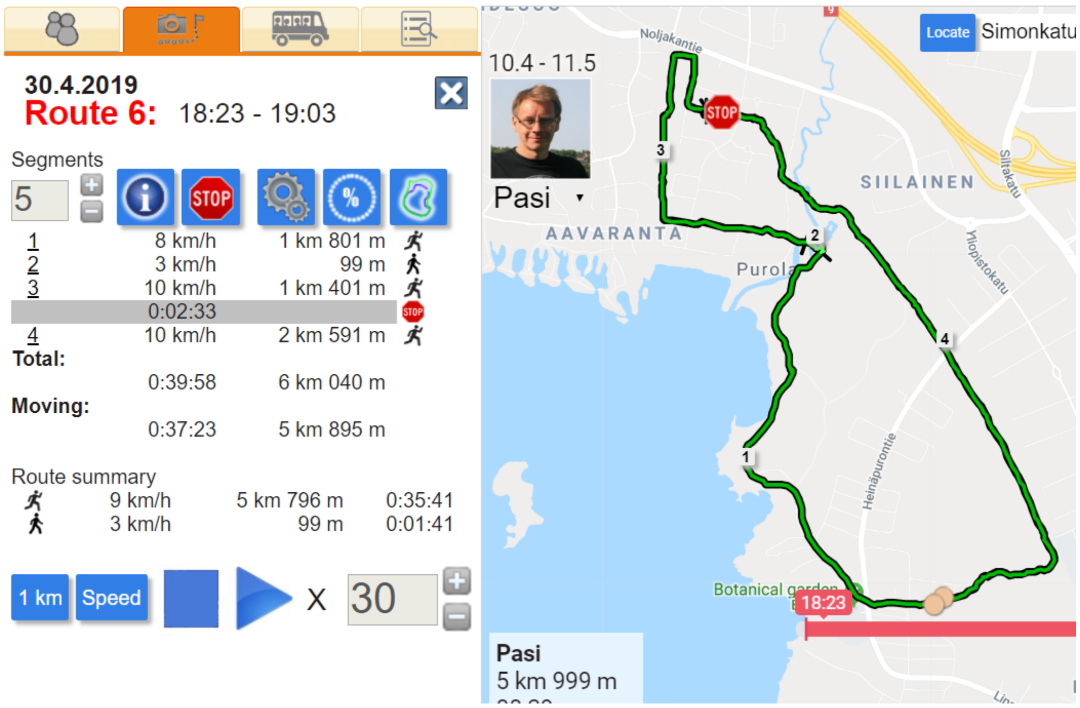


Figure 21: The routes that are highlighted in blue are skiing routes in Joensuu. If a user's trajectory is more 75 % similar to a skiing route and if the trajectory is recorded during the winter season, it is classified as a skiing trajectory.



30.4.2019
Route 6: 18:23 - 19:03

Segments
5

1	8 km/h	1 km 801 m
2	3 km/h	99 m
3	10 km/h	1 km 401 m
	0:02:33	
4	10 km/h	2 km 591 m
Total:	0:39:58	6 km 040 m
Moving:	0:37:23	5 km 895 m

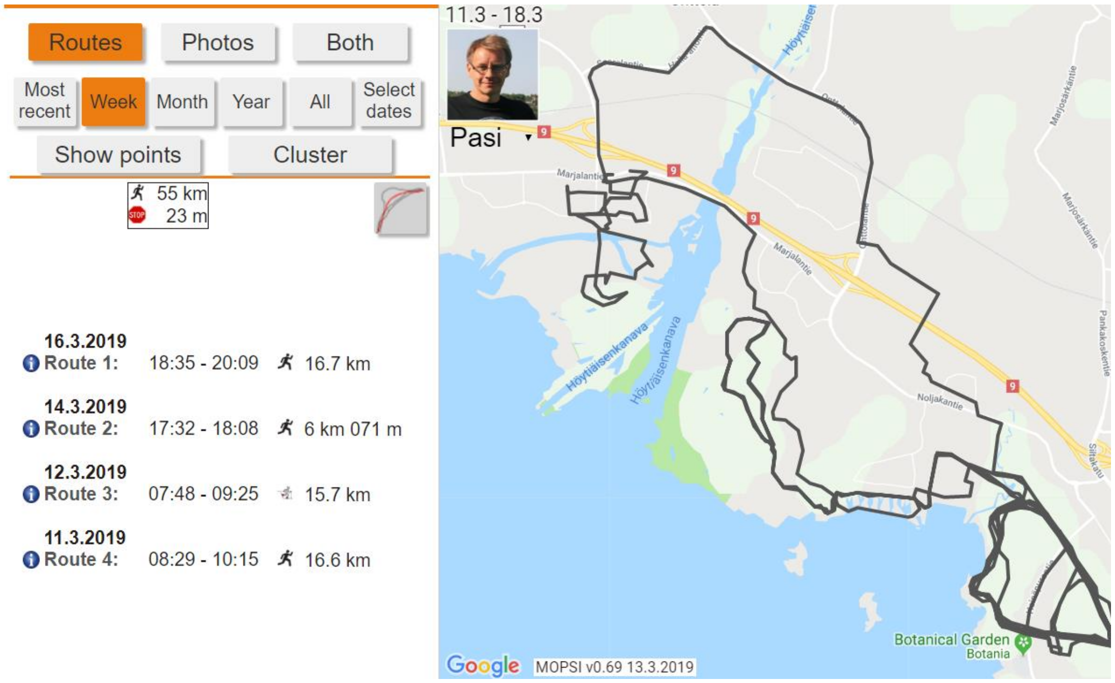
Route summary

9 km/h	5 km 796 m	0:35:41
3 km/h	99 m	0:01:41

Figure 22: Automatic detection of user movement type. The trajectory consists of two running segments and one short walking segment.

Figure 22 shows an example of how the GPS trajectory is segmented to detect the type of the trajectory. All these trajectories are recorded in Mopsi. The detection of the movement type method described earlier in this section and [34] can be seen in Figure 22 on the left side. The trajectory has five main segments with running being the first followed by walking and then running. The fourth segment is the stop segment and then the last one is running. This trajectory is then finally characterized as running by the method in [34].

These GPS trajectories can be used to create a part of the user profile. An example of user's trajectories can be seen in Figure 23. The left side shows a list of the trajectories while the right side shows the trajectories on the map.



16.3.2019
Route 1: 18:35 - 20:09 16.7 km

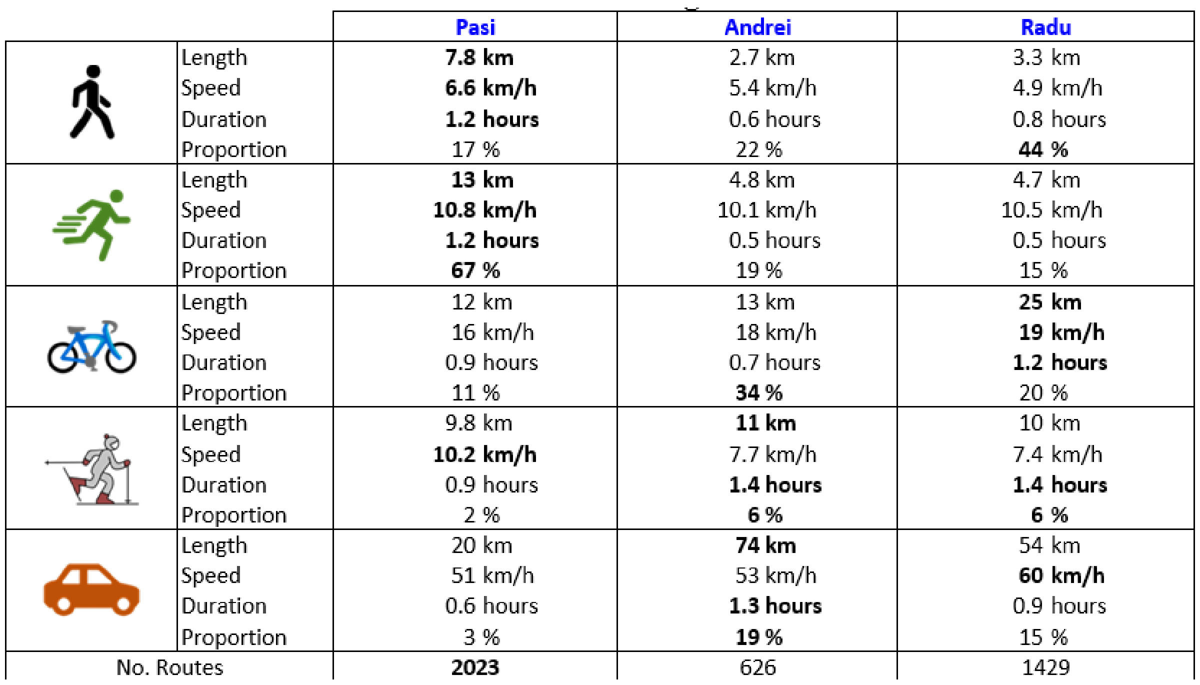
14.3.2019
Route 2: 17:32 - 18:08 6 km 071 m

12.3.2019
Route 3: 07:48 - 09:25 15.7 km

11.3.2019
Route 4: 08:29 - 10:15 16.6 km

Figure 23: User trajectories consisting of three running and one skiing.

Using all the GPS trajectories of a user, part of the user profile was created. Each trajectory has segments with speed, distance, duration, start time, end time and movement type associated with it. For each movement type, we calculated the minimum, average and maximum of speed, distance and duration. Segment statistics are used here to calculate this. The users who have the most data recorded are Pasi, Andrei and Radu. The statistics of their trajectories can be seen in Figure 24.



		Pasi	Andrei	Radu
🚶	Length	**7.8 km**	2.7 km	3.3 km
	Speed	**6.6 km/h**	5.4 km/h	4.9 km/h
	Duration	**1.2 hours**	0.6 hours	0.8 hours
	Proportion	17 %	22 %	**44 %**
🏃	Length	**13 km**	4.8 km	4.7 km
	Speed	**10.8 km/h**	10.1 km/h	10.5 km/h
	Duration	**1.2 hours**	0.5 hours	0.5 hours
	Proportion	**67 %**	19 %	15 %
🚲	Length	12 km	13 km	**25 km**
	Speed	16 km/h	18 km/h	**19 km/h**
	Duration	0.9 hours	0.7 hours	**1.2 hours**
	Proportion	11 %	**34 %**	20 %
⛷	Length	9.8 km	**11 km**	10 km
	Speed	**10.2 km/h**	7.7 km/h	7.4 km/h
	Duration	0.9 hours	**1.4 hours**	**1.4 hours**
	Proportion	2 %	**6 %**	**6 %**
🚗	Length	20 km	**74 km**	54 km
	Speed	51 km/h	53 km/h	**60 km/h**
	Duration	0.6 hours	**1.3 hours**	0.9 hours
	Proportion	3 %	**19 %**	15 %
No. Routes		**2023**	626	1429

Figure 24: Trajectory statistics of Pasi, Andrei and Radu.

Figure 24 shows the average length, speed and duration of each movement type of three users. It also shows the proportion of a movement type compared to all trajectories. This shows an idea of what type of movement a user prefers. Pasi prefers running over other movement types as it covers 67% of his total trajectories. Radu prefers walking and cycling being his second favorite activity to do. Andrei's profile is more balanced with cycling being his preferred activity.

The GPS trajectories recorded in Mopsi have also timestamps. Using these timestamps, we can check what time of the day is a user usually active. We can also check what day of the week and the season in which a user is more active. A demonstration of this is in Figure 25. Pasi's activity on Monday and Friday is less compared to other days. According to Pasi, he plays football and floorball on those days and is therefore less likely to join an event. This could variate between different users. The classifiers could capture this aspect of the user profile.

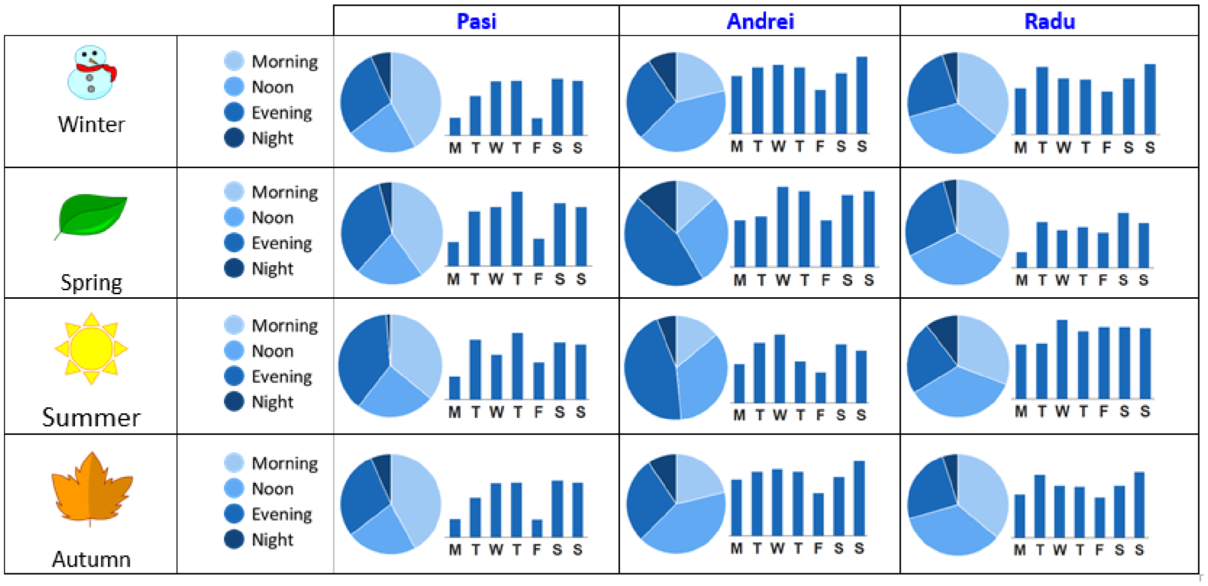


Figure 25: User activity in season, week and time of day.

The seasons are been pre-determined by months. They are:

1. Winter – December, January, February, March
2. Spring – April, May
3. Summer – June, July, August
4. Autumn – September, October, November

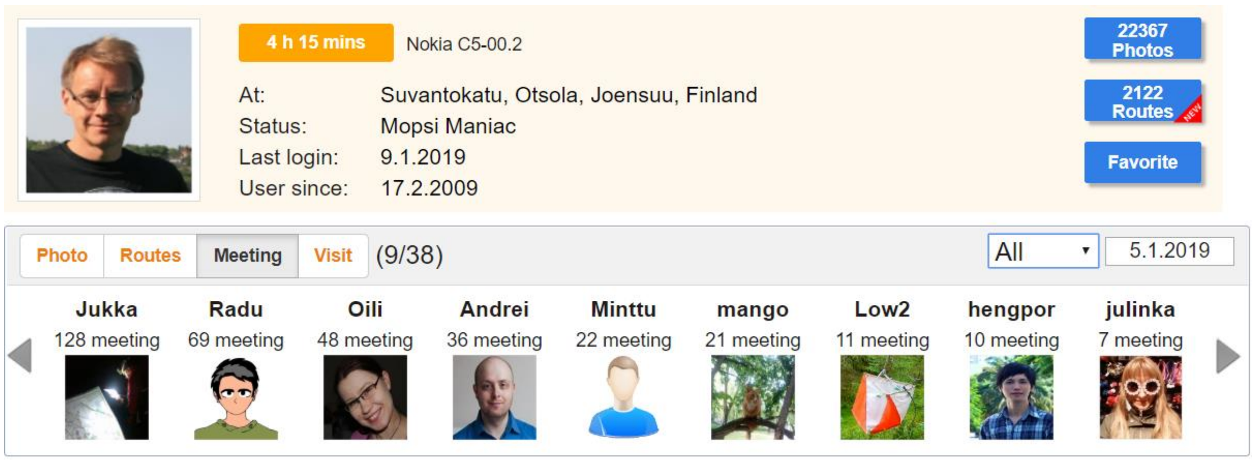
The time of the day has also been pre-determined by the hour of the day. They are:

1. Morning – 4 am – 12 pm
2. Afternoon – 12 pm – 5 pm
3. Evening – 5 pm – 9 pm
4. Night – 9 pm – 4 am

3.2 Meetings

In Mopsi, there are also meetings of users recorded. These can also be used to create a part of the user profile to see the direct relationship between different users. This has already been implemented in Mopsi and it is re-used in this study. The only modification that was done to the code was to make it reusable.

A *Meeting* in Mopsi is defined when a situation arises where multiple users are in close-proximity to each other at the same time. User locations are clustered using a method called single-link agglomerative clustering with a distance threshold applied. This clustering method was chosen to preserve the relationship in case of elongated clusters. There are some inaccuracies with the meetings data. It sometimes classifies users being in a meeting even though it is not a meeting. The method does not work when users are moving either by car, train or walking. When users are moving using a constant speed, the GPS sends location updates every 30 seconds, but it may not send it at the same time, so it appears as though that the users are not together [34].



4 h 15 mins Nokia C5-00.2

At:	Suvantokatu, Otsola, Joensuu, Finland
Status:	Mopsi Maniac
Last login:	9.1.2019
User since:	17.2.2009

Jukka	Radu	Oili	Andrei	Minttu	mango	Low2	hengpor	julinka
128 meeting	69 meeting	48 meeting	36 meeting	22 meeting	21 meeting	11 meeting	10 meeting	7 meeting

Figure 26: Meetings of user Pasi.

Figure 26 shows the list of users that had meeting with Pasi. With this list of meetings, the following statistics can be constructed that are shown in Figure 27. Figure 27 shows the top 5 users or friends for Pasi, Andrei and Radu. For example, in the list of Pasi's meetings, Jukka has the highest number of meetings with Pasi. It can be inferred that Jukka is the top friend for Pasi. This information can be used to find the *closeness* between the host of the event and a participant.

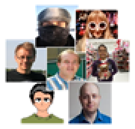
	Pasi		Andrei		Radu	
Meetings	415		544		847	
 Users	Jukka	31 %	Radu	28 %	Andrei	18 %
	Radu	17 %	Jukka	8.0 %	Jukka	9.2 %
	Oili	12 %	Pasi	6.6 %	Pasi	8.0 %
	Andrei	8.7 %	Julinka	5.5 %	Oili	6.0 %
	Minttu	5.3 %	Rezaei	5.0 %	Rezaei	3.6 %
	...		...		...	

Figure 27: Top friends of Pasi, Andrei and Radu

3.3 Photo and service keywords

In Mopsi, users can upload and share photos. This data can also be used partly for the user profile. Figure 28 shows an example of Pasi's uploaded photos. With each of the photos, the users can enter a description of the photo. An example of this can be seen in Figure 29

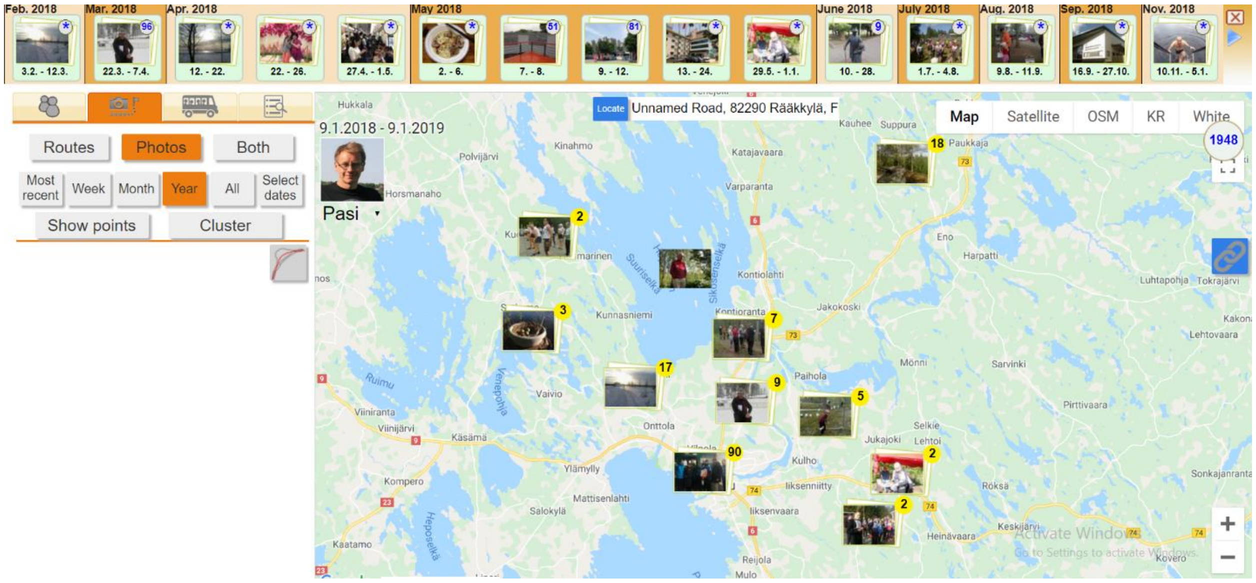


Figure 28: Uploaded photos of user Pasi visualized on map.

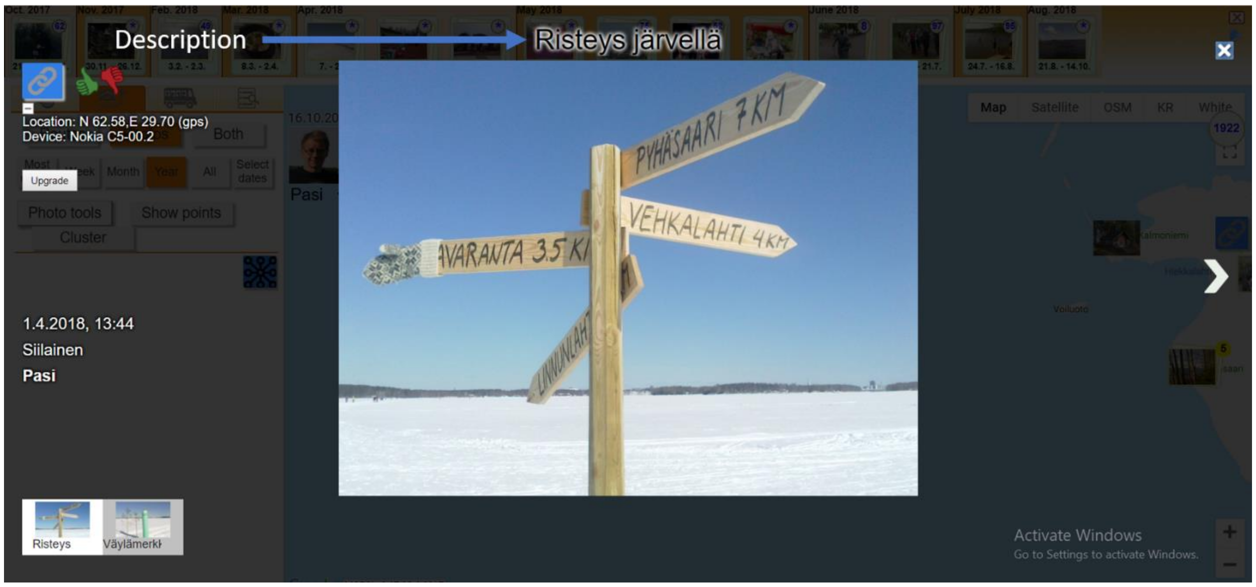


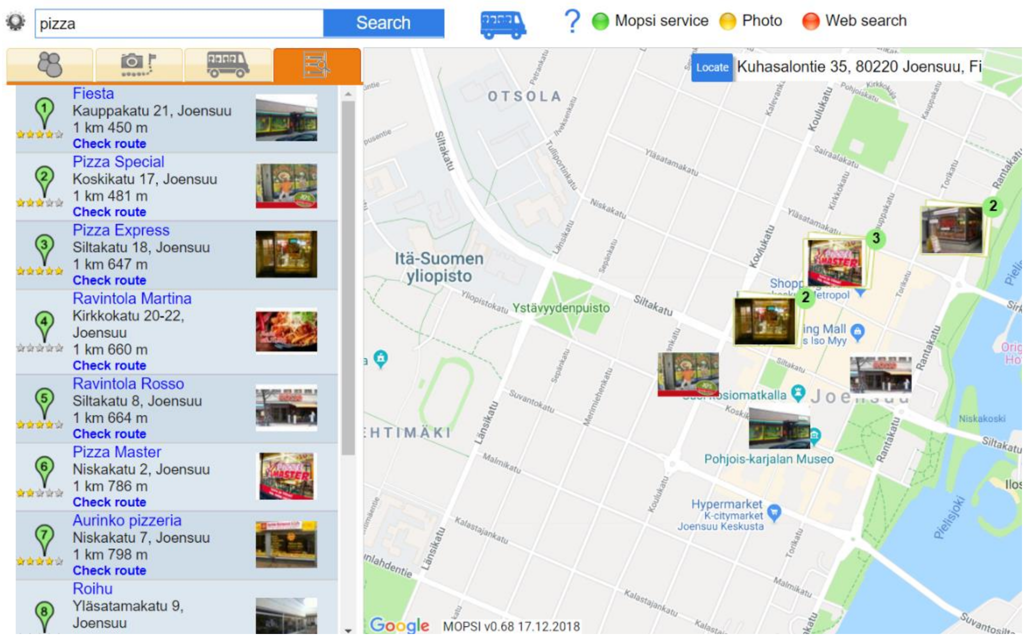
Figure 29: Description of the photo uploaded by Pasi

Since there are descriptions with photos like in Figure 29, this data can also be utilized for the user profile. This information can be used to see if an event's title matches the description of the photo. Similarity of the description and title indicate that a user is more likely to participate the event. Figure 30 shows the top keywords used by Pasi, Andrei and Radu. The percentage in Figure 28 is calculated as the number of times a keyword is used from the total amount of keywords. For example, Pasi's top keyword which is *Park* is used 1.2 % from his total number of keywords which is 44081.

	Pasi		Andrei		Radu	
Keywords	44081		3379		3016	
Photos	Park	1.2 %	Road	2.1 %	Snow	1.1 %
	View	1.1 %	Joensuu	1.6 %	Beach	1.1 %
	Street	1.0 %	SciFest	1.2 %	View	1.0 %
	Restaurant	0.9 %	Kaislakatu	1.1 %	Lake	1.0 %
	Talo	0.9 %	Mopsi	1.0 %	Koli	0.8 %
	...		...		...	

Figure 30: Top keywords from photo descriptions

There is another feature called the service recommendation in Mopsi. A user simply types a word into the search box and a list of nearby services are returned. Service in this context means a service provider, for example a pizza restaurant. Users in Mopsi can visit these service points. There is data in Mopsi about the history of the users visiting these service points. These visits are kind of check-ins, except that they were generated automatically. Figure 31 shows a list of services with the keyword *Pizza*.



1 Fiesta
Kauppakatu 21, Joensuu
1 km 450 m
Check route

2 Pizza Special
Koskikatu 17, Joensuu
1 km 481 m
Check route

3 Pizza Express
Siltakatu 18, Joensuu
1 km 647 m
Check route

4 Ravintola Martina
Kirkkokatu 20-22, Joensuu
1 km 660 m
Check route

5 Ravintola Rosso
Siltakatu 8, Joensuu
1 km 664 m
Check route

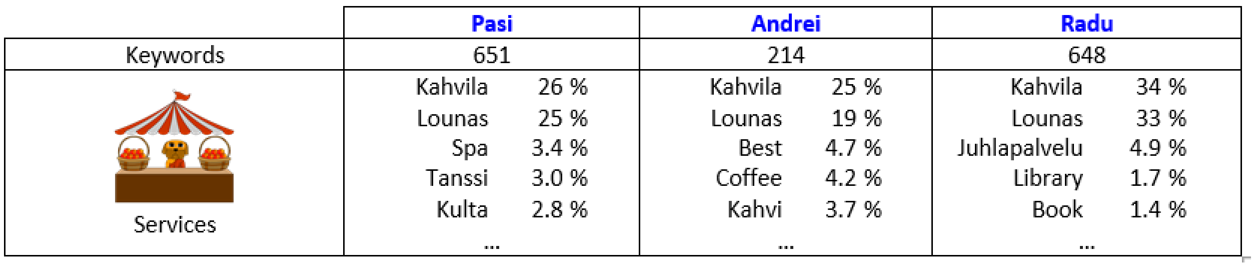
6 Pizza Master
Niskakatu 2, Joensuu
1 km 786 m
Check route

7 Aurinko pizzeria
Niskakatu 7, Joensuu
1 km 798 m
Check route

8 Roihu
Yläsatamakatu 9, Joensuu

Figure 31: Service recommendation with the keyword *pizza*

Each of these services has keywords associated with it. For example, a *pizza restaurant* would have the keyword *pizza* and *restaurant* with it. Figure 32 shows the top keywords of services that Pasi, Andrei and Radu have visited.



	Pasi	Andrei	Radu
Keywords	651	214	648
Services	Kahvila 26 % Lounas 25 % Spa 3.4 % Tanssi 3.0 % Kulta 2.8 % ...	Kahvila 25 % Lounas 19 % Best 4.7 % Coffee 4.2 % Kahvi 3.7 % ...	Kahvila 34 % Lounas 33 % Juhlapalvelu 4.9 % Library 1.7 % Book 1.4 % ...

Figure 32: Top keywords of services visited by user Pasi, Andrei and Radu

The photo and service keywords data had to be cleaned because of some noise in the data such as symbols or numbers in a keyword. Symbols, numbers and stop words were removed from a keyword to be valid for user profiling.

Stop words are the most common words used in a sentence. These words are usually removed before or after natural language processing is performed because it could take valuable processing time. Table 1 shows a sample list of stop words.

Table 1: Sample list of stop words

I	Me
Doing	Against
Our	But
Will	Between
Than	Too
Very	Through
During	Here
Same	Having

The sections discussed above shapes the user profiles for each user. Using this data, we can leverage the information if a user would attend an event or recommend events to them. The data used for user profiling is summarized in Table 2.

Table 2: The frequencies of that data used from Mopsi to create user profiles

Data	**Pasi**	**Andrei**	**Radu**	**All**
Trajectories	2023	626	1429	4078
Meetings	415	544	847	1806
Photo descriptions	44081	3379	3016	50476
Services	651	214	648	1513

4. Compatibility between user and event

In this chapter, the features used for classification will be discussed. The compatibility of a user and an event will also be discussed. Compatibility in this context means if a user is likely to attend for a given event. These compatibility measurements can also be used to recommend events for a user.

4.1 Classifiers

Classification is the process of identifying which class (or category) a new data observation belongs to, based on training data set where the past data observations are already categorized. Algorithms that perform classification are called *classifiers*. Binary classifiers are used here to match a user with an event to predict if the user will attend an event, and which event can be recommended to a user.

Features are used as data observations for classification. Features are some measurable properties that contains information to describe a category's characteristics. A single feature vector in this system will contain information about a user-event pair which will be described in detail later in this chapter.

Four different classifiers are used to perform binary classification. The first classifier used is *k-nearest-neighbor (kNN)*. It is a classification method where the training data is stored and when a new unseen data point *x* is classified simply by comparing it to its neighbors in the dataset. Out of all the training vectors in the dataset, it identifies the *k* nearest neighbors by some distance measure between *x* and its training vectors regardless of class labels. Euclidean or hamming distance functions are commonly used as distance measures. Each of the *k* samples has a class that it belongs to. The *x* is then assigned to the class that has the maximum number among the *k* neighboring samples.

The second classifier is *support vector machine* (*SVM*). SVM builds a model where data points are assigned to different categories. It builds a clear line with maximum margin between categories so that unseen data points can be assigned to these categories [30]. An example of how SVM could build a line in the feature space in the training phase, is that it could build a model that defines the gap between *true* and *false* observations. *True* would indicate that a user is attending an event while *false* would mean the user is not attending an event. A new data point would be plotted on either side of the line which is then classified to one of the categories.

Naive bayes is the third classifier that is used. It is one of the probabilistic classifiers. Bayes classifier builds a model and assigns labels in the training set. A naïve bayes classifier considers each feature to contribute independently that this belongs to one of the class variables without considering the correlations between them. Bayes assigns the most probable class to an unseen data point which is described by its feature vector [31].

The fourth classifier that is used in this study is *decision trees*. Decision tree uses observations from the data set to conclude the item's target class variable. The leaves in the tree represents the class labels and the branches represent the conjunctions to why it leads to a class label. For a new input, the decision tree classifies it as one of the target categories by starting at the root node of the tree. Then, it follows down a specific path from the root to the leaf node which is then categorized as one of the target classes. Each parent node in the tree has a decision criterion which makes the input fall to either side of the tree down to its leaf nodes which is then categorized as one of the class variables. Each parent node has an input variable which is one of the features from the feature vector which is chosen by the model to determine which region does the input fall under for classification [30].

4.2 Features

Before any classification task was done, we extracted feature vectors from the Mopsi dataset. *Feature modelling* must be done where each feature must be extracted to form a feature vector. For feature modelling, a binary class function has been defined for every user-event pair. It is a binary classification problem where the value is true if the prediction of the event for a user is matching.

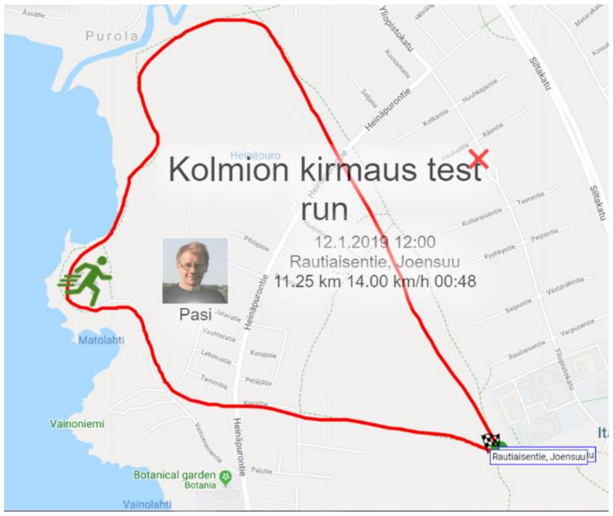
$$Match(u,e) = \begin{cases} TRUE & if\ Classifier(u \circ e) = Positive\ Class \\ FALSE & if\ otherwise \end{cases} \tag{1}$$

Equation 1 shows user u and event e. This binary function constructs feature vectors using information from the user profile and the information from the specific event. The classifiers mentioned previously is then used to classify if an event is suitable for the user or not. The recommended events for a specific user are constructed with equation 2. Similarly, it can be predicted if a user will join an event with equation 3.

$$Recommended(u) = \{e \in E \mid Match(u,e) = TRUE\} \tag{2}$$

$$Predicted(e) = \{u \in U \mid Match(u,e) = TRUE\} \tag{3}$$

We created a 9-dimensional feature vectors for each user-event pair. Each feature vector belongs to one of the following classes which will be described in the following sections. Consider two events in Figure 33. When the features will be described, the compatibility of a user and an event will also be discussed with real statistics from Mopsi.



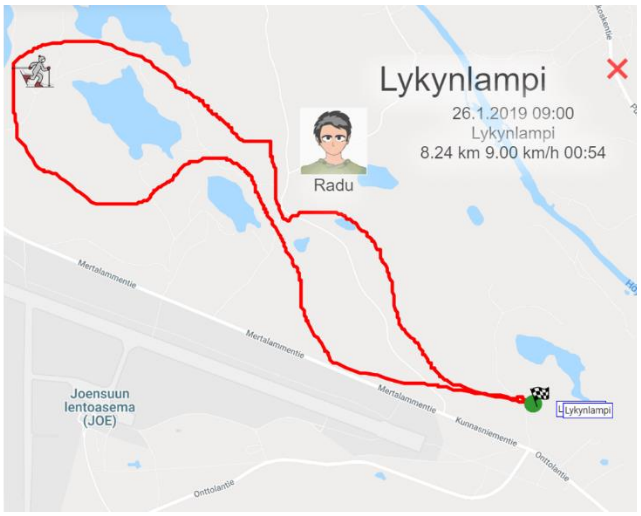


Figure 33: Examples of Pasi's (left) and Radu's events (right)

4.2.1 Community

One of the feature vectors belongs to this class. The meetings data from the user profile has been leveraged to create this feature vector. This feature has been considered to capture and quantify the relationship between the host of the event and a user. Equation 4 shows how this feature is calculated.

$$Community(u,h) = \frac{MeetingCount(u,h)}{\sum_{i=1}^{|U|} MeetingCount(u,u_i)} \quad (4)$$

This feature measures how likely would a user *u* meet the host of the event *h* based on past meetings to the total number of meetings of *u*. In this feature, the more the user *u* has meetings with the host of the total meetings of *u*, the higher is the probability that the user would join this event. The higher the percentage that the host has meetings with user *u*, the higher the probability. Take for example Pasi's event in Figure 33. Let us figure out if user Radu would be a good match for Pasi's event in terms of this feature. Figure 27 shows the percentage of meetings for Pasi, Andrei and Radu. Pasi covers 8% of Radu's total meetings, the third highest. This means that Radu has a high chance of joining Pasi's event by using this feature alone. An example of how this feature is used in the classifier is 69/847. The numerator is total meetings between Pasi and Radu while the denominator is the total meetings that Radu had.

Other features related to this class were also considered. One of them is that a user would join the event if he/she sees a common friend who is already joining the event. In practice, this feature did not have any significant effect of the probability of the user joining the event.

4.2.2 Suitability

There are two feature vectors related to this class. The first is how often a user performs the movement activity which is specified by the event. For this feature vector, the data from the user profile has been analyzed. Namely, the route statistics of the user has been analyzed to see how often she/he performs the movement type which is specific to the event in the past. The season also has been considered. The reason for this is that as the season changes, the user's interest in a movement type might change. For example, in the summer, a user would prefer cycling while in the winter, skiing would be preferred by the user.

$$Suitability(e,r)_{TYPE} = \frac{1}{|R|}\sum_{i=1}^{|R|}(SameType(e,r_i) \text{ AND } SameSeason(e,r_i)) \qquad (5)$$

Equation 5 shows the value of the feature vector. Here, *R* is the total number trajectories recorded in Mopsi by the user while *e* is the event and *r* are a trajectory from the set *R*.

Let us consider whether user Radu is a good fit for suitability (type) of Pasi's event. Pasi's event is of running type. Radu has a total of 76 routes which are in the same season and the same type as Pasi's event. Radu has a total of 1429 routes recorded in Mopsi. So, what does this say about if Radu will join this event or not? In total, 5.3% of Radu's trajectories are in the same season and same type as Pasi's event. It is quite a small number, so Radu may not join Pasi's event on Figure 33. Let us consider if Pasi would join Radu's event in Figure 33. In Figure 24, Pasi has 2 % of his trajectories to be skiing. It is less likely that Pasi would join Radu's event by this feature alone.

The second feature vector which belongs to this class is related to the skill of the user *u* to participate in an event *e*. The skill is defined by the speed and distance. Before the equation for this feature vector is discussed, it is appropriate to discuss the movement activities of a user. User Radu's activities will be discussed. Radu's activities are mostly walking, cycling and skiing.

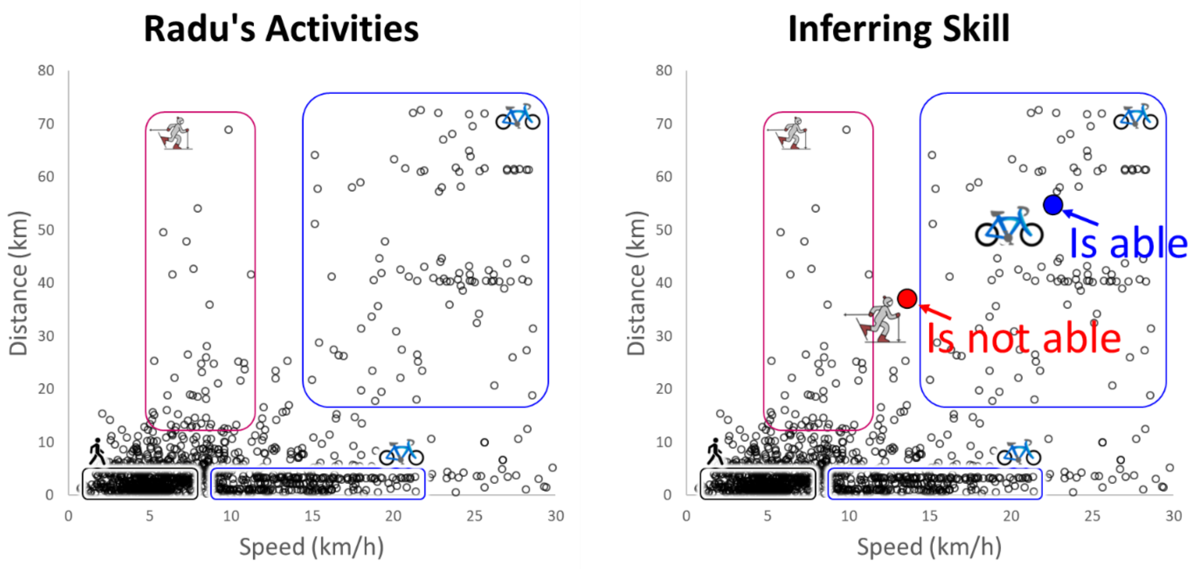


Figure 34: User Radu's trajectories visualized with speed and distance.

Figure 34 shows the activities of Radu. There are several patterns that can be observed. Radu is a long-distance cycling enthusiast. At the top-right on Figure 34, this is reflected on the data as well. There is a cluster of long-distance cycling for Radu. The 60 – 70 km distance corresponds to running which are only a few observations. The 40 km cycling corresponds to training. The 15 – 20 km distance observations are a cluster where Radu has done cross-country skiing. The bottom cluster which has the most observations corresponds to commuting to work by walking or cycling.

To give an idea of how and why this feature vector is relevant, two different events will be considered. The cycling event which has a 55 km distance at a pace of 23 km/h. To evaluate if Radu can participate in this event, the top right corner on Figure 34 shows that Radu has participated to more tougher events at higher distance and speed. Radu is therefore a good fit for this event. Another example of an event is the skiing event which covers 35 km distance at a pace of 14 km/h. Radu has observations which is greater than 35 km distance, but his pace is slower. This event would be too fast-paced for Radu so it is not be a good fit for him. Equation 6 shows the formula for this feature vector.

$$Suitability(e,r)_{SKILL} = \frac{1}{|R|}\sum_{i=1}^{|R|}(Speed(e) < Speed(r_i) \text{ AND } Distance(e) < Distance(r_i)) \quad (6)$$

4.2.3 Familiarity

One of the features belongs to this class. This class refers to the geographical region where an event will take place. Some users prefer to explore new regions while some users would attend events because the region is familiar to them. Familiarity is calculated using similarity of the event route and past trajectories that has been recorded in Mopsi by the user. Familiarity is calculated as the percentage of the previous trajectories that have a similarity of more than 75 % with the trajectory of the event. Equation 7 shows how this is calculated.

$$\boldsymbol{Familiarity(e,r)} = \frac{\mathbf{1}}{|\boldsymbol{R}|}\sum_{\boldsymbol{i=1}}^{|\boldsymbol{R}|}(\boldsymbol{CSIM}(\boldsymbol{Route}(\boldsymbol{e}),\boldsymbol{r_i}) > \mathbf{75}\%) \tag{7}$$

The cell size used for the similarity measure is automatically determined by the zoom-level used by the event host to draw the trajectory. The reason behind this is to consider the different drawing precision at various zoom levels of the map. A threshold of 75 % was set experimentally because for the differences caused by users living at different locations than common meeting points. The more trajectories a user has similar to the one in the event, the more likely the user is predicted to join the event with this feature alone.

4.2.4 Availability

There are three feature vectors related to the availability class. The first feature measures the activity of a user in a season. The trajectories of the users have been used to determine the activity of a user in a season. The motivation of using this as a feature is to determine if a user is usually active in a season or does it gradually change in other seasons. A user could be very active in summer while the activity goes down during winter. If a user has high activity in a season and the event is also in the same season, the more likely the user would join the event. Equation 8 shows how the feature vector is calculated.

$$Availability(e,r)_{SEASON} = \frac{1}{|R|}\sum_{i=1}^{|R|}(Season(e) = Season(r_i)) \tag{8}$$

The trajectory statistics from the user profile have been used here. Here, *r* is a trajectory from the trajectory set *R*. Let us consider if Radu is compatible with Pasi’s event. Pasi’s event in Figure 30 will be held in the winter season. Radu has 567 trajectories out of 1429 in the winter season. With this feature vector alone, Radu is predicted join Pasi’s event. An example of how this feature is used in the classifier is 567/1429.

The second feature vector is related to if the user is active on the day of the week when the event is taking place. Equation 9 shows how this feature vector is extracted.

$$Availability(e,r)_{DAY} = \frac{1}{|R|}\sum_{i=1}^{|R|}(Day(e) = Day(r_i)) \qquad (9)$$

The route statistics from the user profile is used here. Here, *r* is a trajectory from the trajectory set *R*. The routes that has been recorded by the user in Mopsi are used to find which routes took place on what day of the week. The motivation of using this feature is to determine if the user is active on certain days of the week compared to other days. A user may be less active physically on weekends compared to weekdays. If an event is on a weekday and a user is usually active on that day, the user might join the event.

Let us consider if user Radu is compatible with Pasi's event according to this feature. Pasi's event in Figure 33 is going to be held on 9.2.2019. The event is on Saturday. Radu has 257 trajectories out of 1429 on Saturdays. 18% of Radu's trajectories are on Saturdays. Based on this feature alone, Radu may join Pasi's event.

The third feature which belongs to this class is related to the time of the day which is morning, afternoon, evening or night. The trajectory set has been used to calculate what route took place on what time of the day. Equation 10 shows how this has been calculated.

$$Availability(e,r)_{TIME} = \frac{1}{|R|}\sum_{i=1}^{|R|}(Time(e) = Time(r_i)) \qquad (10)$$

Let us consider if Radu is compatible to Pasi's event. Pasi's event is in the afternoon. Radu has a total of 521 trajectories in the afternoon. In total, 36% of Radu's trajectories is in the afternoon. Considering this feature, Radu would join Pasi's event.

4.2.5 Interests

Interests has also been measured. The keywords from the photos and services that user has visited is used from the user profile. The photo keywords are from the description that user has entered in the system. Equation 11 and 12 shows how these two feature vectors have been calculated.

$$Interests(e,k)_{PHOTOS} = \frac{1}{|P_K|}\sum_{i=1}^{|P_K|}(Levenshtein(Title(e),k_i) > 90\%) \qquad (11)$$

$$Interests(e,k)_{SERVICES} = \frac{1}{|S_K|}\sum_{i=1}^{|Sk|}(Levenshtein(Title(e),k_i) > 90\%) \qquad (12)$$

Here, P_k and S_k are the set of keywords obtained from the photo description and the services visited by the user, k_i is a single keyword, and *Levenshtein* is a method used here for calculating the distance between the two strings. The distance is then converted in to a similarity measure by normalizing it with respect to the longest of the two strings.

All the words in the title of an event is used to check if these words exist in the keywords used in photos or services by the user.

Let us consider if user Pasi would join Radu's event in Figure 33. The title of Radu's event is *Lykynlampi*. Pasi does not have any keywords in photos or services that has this keyword so Pasi is predicted not to join the event with these two features.

I have discussed how each of the features can be used to measure compatibility with a user and an event. Considering all of them at once would be quite complex to discuss. We will discuss how the classifiers can be used to simplify and show how all of these features can be combined and can be used for prediction or recommendation.

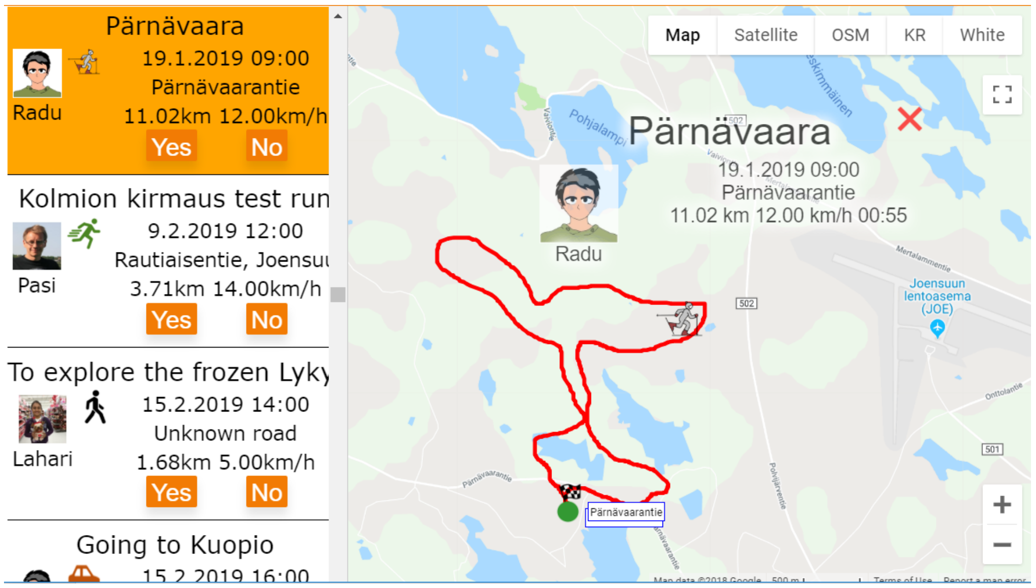
5. Evaluation

Experiments were performed using real events created by 8 volunteers in the city of Joensuu for the next 12 months. A total of 150 events were created by the volunteers. This time frame was chosen so that all seasons are covered. Table 3 shows the number of events created by the volunteers.

Table 3: 150 events created by 8 volunteers. The events which are highlighted in blue are the dominant moving type of each user.

	🚶	🏃	🚲	⛷	🚗	🚌	🚆	**Total**
Pasi	1	**14**			4		1	**20**
Radu	3	4	7	**10**	1			**25**
Oili	**13**		4		2	1	1	**21**
Abu	4	1	**5**		3	4	1	**18**
Lahari	6		**9**		3	1	1	**20**
Sami	**7**	1	4	1	4			**17**
Nancy	**6**	4	4		1	4		**19**
Usman	**10**							**10**
Total	**50**	**24**	**33**	**11**	**18**	**10**	**4**	**150**

A special tool was then made for ground truth collection. Twelve most active Mopsi users were asked to mark down whether they would participate to the events listed in Table 3. Several of the users are now currently living abroad so we asked them to mark down their participation as if they were still living in Joensuu. Figure 35 shows the ground truth collection tool. The tool is essentially the same as the real Mopsi Event system but with two buttons, *Yes* and *No* for each event. The *Yes* and *No* answers allow us to have ground truth for each event. When user presses either of the two buttons, the answer is registered, and the event is removed from his/her list.



Pärnävaara
Radu
19.1.2019 09:00
Pärnävaarantie
11.02km 12.00km/h
Yes No

Kolmion kirmaus test run
Pasi
9.2.2019 12:00
Rautiaisentie, Joensuu
3.71km 14.00km/h
Yes No

To explore the frozen Lyky
Lahari
15.2.2019 14:00
Unknown road
1.68km 5.00km/h
Yes No

Going to Kuopio
15.2.2019 16:00

Pärnävaara
Radu
19.1.2019 09:00
Pärnävaarantie
11.02 km 12.00 km/h 00:55

Figure 35: The tool for ground truth collection with events shown on the list. Clicking on each event will show the properties of the event on the map.

The answers for each of the twelve volunteers are summarized in Table 4. For the classifiers, the collected data was divided 50 % for training and 50 % for testing instead of the normal 80 / 20 % dividing. The reason is to cover every season for training and testing. The events were sorted by date and we took every odd event of every user for training and even event for testing. Another reason is that some users were attending more events in a season and just adding those data in either training or testing would not end up giving good results. If those data were only in testing, the classifiers would see a form of unseen data that was not covered in training.

Table 4: The twelve volunteers and their choices to attend the events. The number of attendances is shown per training and testing. The numbers which are highlighted in blue are users who are attending most of the events while the numbers which are highlighted in red are users who attend the least.

	TRAINING		**TESTING**		**TOTAL**	
Name	**Attending**	**Percent**	**Attending**	**Percent**	**Attending**	**Percent**
Andrei	33 / 75	44 %	37 / 75	49 %	70 / 150	47 %
Pasi	9 / 65	14 %	10 / 65	15 %	19 / 130	15 %
Jukka	21 / 75	28 %	25 / 75	33 %	46 / 150	31 %
Karol	29 / 75	39 %	35 / 75	47 %	64 / 150	43 %
Mikko	**1 / 75**	**1 %**	**0 / 75**	**0 %**	**1 / 150**	**1 %**
Zhentian	**71 / 75**	**95 %**	**72 / 75**	**96 %**	**143 / 150**	**95 %**
Radu	20 / 62	32 %	23 / 63	37 %	43 / 125	34 %
Oili	**3 / 64**	**5 %**	**1 / 65**	**2 %**	**4 / 129**	**3 %**
Matti	24 / 75	32 %	30 / 75	40 %	54 / 150	36 %
Julinka	48 / 75	64 %	49 / 75	65 %	97 / 150	65 %
Sami	3 / 65	5 %	6 / 65	9 %	9 / 130	7 %
Lahari	13 / 65	20 %	10 / 65	15 %	23 / 130	18 %
TOTAL	275 / 846	33 %	298 / 848	35 %	573 / 1694	34 %

Table 5 shows how the features described in chapter 4 correlate with the user's decisions to participate an event. Not every feature is important to every user. The host of the event seems to highly correlate to users like Pasi, Jukka, Karol and Oili. The correlation shows that the host of the event is what is most important to them. For user Radu, the suitability of the event is what mattered most. He does not participate events which are unsuitable to him. For the class familiarity, the location of the route seems to be important to Karol, Andrei, Zhentain, Matti and Sami. Julinka seems to have the highest correlation in the *Interests* class compared to other classes.

Table 5: Pearson correlation of the feature and user's decision to participate. Correlations above 0.3 are highlighted in blue and the exceptionally high ones are highlighted in **bold**.

Name	Community	Suitability		Familiarity	Availability			Interests	
		Type	Skill		Season	Time	Day	Photos	Services
Andrei	0.06	0.17	0.18	0.04	0.00	0.33	0.05	0.10	0.10
Pasi	0.28	0.12	0.13	0.14	0.12	0.19	0.08	0.22	0.07
Jukka	**0.60**	0.15	0.21	0.22	0.29	0.21	0.02	0.05	0.18
Karol	0.35	0.06	0.38	0.12	0.21	0.14	0.03	0.02	0.00
Mikko	0.17	0.09	0.15	0.06	0.02	0.08	0.13	0.05	0.00
Zhentian	0.19	0.04	0.19	0.06	0.33	0.16	0.23	0.06	0.00
Radu	0.11	**0.42**	0.17	0.24	0.31	0.18	0.07	0.20	0.13
Oili	0.30	0.12	0.03	0.16	0.06	0.11	0.16	0.08	0.03
Matti	0.12	0.03	0.19	0.06	0.11	0.35	0.09	0.04	0.08
Julinka	0.08	0.08	0.03	0.03	0.09	0.04	0.09	0.13	0.22
Sami	0.00	0.03	0.04	0.04	0.03	0.08	0.02	0.00	0.00
Lahari	0.00	0.22	0.11	0.04	0.09	0.07	0.14	0.06	0.00
TOTAL	0.19	0.13	0.15	0.10	0.14	0.16	0.09	0.08	0.07

All correlations are relatively low for users Sami, Julinka, Mikko and Lahari. It does not mean that the features are not meaningful, but a combination of the features can still lead to accurate prediction for these users. However, for some users like Jukka, only one feature, which is the community feature, is enough for successful prediction because he mostly attends Pasi's events. It can be noticed in Table 5 that community, suitability and availability features are the most important in general.

5.1 Classification

For kNN, we used a grid search for finding a good parameter for the number of neighbors (k). We found out that k = 4 is the best parameter from the range 1 to 10. For SVM, we found out that a logarithmic grid with base 10 for parameters C and γ. This was done by optimizing for the training set.

A personalized model has been computed for every user with respective to their training data. A general model was also computed using the training data of all the users. A general model is also being used to find out if it is more accuracy prediction than the personalized model. Accuracy and precision were used to evaluate the models. Equation 13 shows how to calculate the accuracy and Equation 14 shows how to calculate the precision.

$$Accuracy = \frac{number\ of\ correctly\ classified\ events}{total\ number\ of\ events} \tag{13}$$

$$Precision = \frac{number\ of\ correctly\ predicted\ events}{number\ of\ events\ user\ is\ attending} \tag{14}$$

Prediction by random chance is 50 %. A good prediction system should therefore be better than 50 %. The models computed here; all give accuracy higher than 50 % no matter which model is used. Table 6 and Table 7 shows the accuracy and precision of general and personalized model for each classifier that was used. On average, the best accuracy is given by decision tree which is 73 – 76 %. KNN gives 68 – 75 % accuracy, while SVM gives 66 – 75 % which is still good. Bayes classifier gives the worst results which is 55 – 63% but it is still better than random chance. KNN gives the best prediction performance for half of the users (6/12). Decision tree gives the best results for 4 users while SVM and Bayes give best prediction performance for one user each.

Table 6: Accuracy shown for each user using general and personalized model. The best models are highlighted in blue

	KNN		Decision Tree		SVM		Naïve Bayes	
	General	**Personal**	**General**	**Personal**	**General**	**Personal**	**General**	**Personal**
Andrei	56 %	55 %	60 %	59 %	51 %	60 %	51 %	49 %
Pasi	86 %	83 %	72 %	75 %	78 %	75 %	58 %	51 %
Jukka	65 %	65 %	79 %	77 %	71 %	79 %	44 %	80 %
Karol	60 %	55 %	63 %	51 %	55 %	53 %	47 %	61 %
Mikko	91 %	100 %	95 %	99 %	92 %	93 %	64 %	100 %
Zhentian	45 %	95 %	80 %	92 %	19 %	89 %	80 %	67 %
Radu	59 %	60 %	59 %	70 %	60 %	59 %	37 %	43 %
Oili	95 %	98 %	88 %	94 %	95 %	97 %	25 %	82 %
Matti	59 %	64 %	57 %	61 %	59 %	71 %	52 %	53 %
Julinka	39 %	56 %	55 %	63 %	35 %	56 %	57 %	63 %
Sami	88 %	91 %	89 %	92 %	88 %	88 %	68 %	89 %
Lahari	78 %	83 %	77 %	83 %	83 %	80 %	75 %	17 %
Avg	68 %	75 %	73 %	76 %	66 %	75 %	55 %	63 %

Table 7: Precision shown for each user using general and personalized model.

	KNN		Decision Tree		SVM		Naïve Bayes	
	General	**Personal**	**General**	**Personal**	**General**	**Personal**	**General**	**Personal**
Andrei	68 %	64 %	63 %	59 %	50 %	62 %	50 %	49 %
Pasi	67 %	0 %	10 %	20 %	0 %	25 %	23 %	19 %
Jukka	46 %	45 %	67 %	62 %	100 %	70 %	35 %	71 %
Karol	69 %	55 %	68 %	47 %	56 %	53 %	47 %	63 %
Mikko	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %
Zhentian	100 %	97 %	97 %	97 %	100 %	97 %	97 %	98 %
Radu	20 %	33 %	41 %	61 %	25 %	40 %	33 %	38 %
Oili	0 %	0 %	0 %	0 %	0 %	0 %	2 %	0 %
Matti	46 %	59 %	45 %	52 %	0 %	75 %	43 %	46 %
Julinka	60 %	68 %	68 %	73 %	50 %	66 %	63 %	64 %
Sami	0 %	0 %	33 %	100 %	0 %	0 %	14 %	0 %
Lahari	30 %	0 %	27 %	33 %	0 %	0 %	20 %	15 %
Avg	42 %	35 %	43 %	50 %	32 %	40 %	36 %	39 %

Pasi has high accuracy of 83 % with the general model. This means that the *kNN* classifier predicts accurately that Pasi is not going to attend a lot of events. The precision however is poor for Pasi's personalized model with 0 %. In other words, the personalized model for Pasi was not able to predict even a single event that Pasi is participating. The general model was able to predict with 67 % precision.

For user Zhentian, the reverse is true. The personalized model gives a better accuracy because some of the events that he is participating in is not matching his user profile. Zhentian likes to explore new things and likes to attend a lot of events that does not match his user profile, the personalized model captures this aspect. The precision is poor for two users, Oili and Mikko, because they are attending only a very few events. This shows the aspect that it can be challenging to predict users who are very selective of their events.

The data is well-structured, so the decision tree classifier is performing well. The decision tree also performs well with users who have less data such as Lahari and Sami. Therefore, the decision tree can be used for cold-start users who have less data.

Table 8: The best accuracy and precision (in parenthesis) using the best classifier for each user.

	Best of all	**Classifier**	**Model**
Andrei	56 % (68 %)	KNN	general
Pasi	86 % (67 %)	KNN	general
Jukka	80 % (71 %)	Naive Bayes	personal
Karol	60 % (69 %)	KNN	general
Mikko	100 % (0 %)	KNN	personal
Zhentian	95 % (97 %)	KNN	personal
Radu	70 % (61 %)	Decision Tree	personal
Oili	98 % (0 %)	KNN	personal
Matti	71 % (75 %)	SVM	personal
Julinka	92 % (73 %)	Decision Tree	personal
Sami	92 % (100 %)	Decision Tree	personal
Lahari	83 % (33 %)	Decision Tree	personal
Avg.	80 % (60 %)		

Table 8 shows the best classifiers for each user using accuracy and precision. This can be utilized in the events system by selecting the best classifier for each user individually to recommend events or predicting attendance. The average accuracy for the users reached up to 80 %. To compare this to other systems found in literature, two different system will be referred. The mean average precision in [18] is reported 55 % for recommending scientific talks to users. An accuracy of 76 % was reported in [23] for recommending services to moving users. Therefore, the accuracy and precision reported with the method presented in this study is no worse than the ones reported in [18] and [23].

5.2 Decision tree

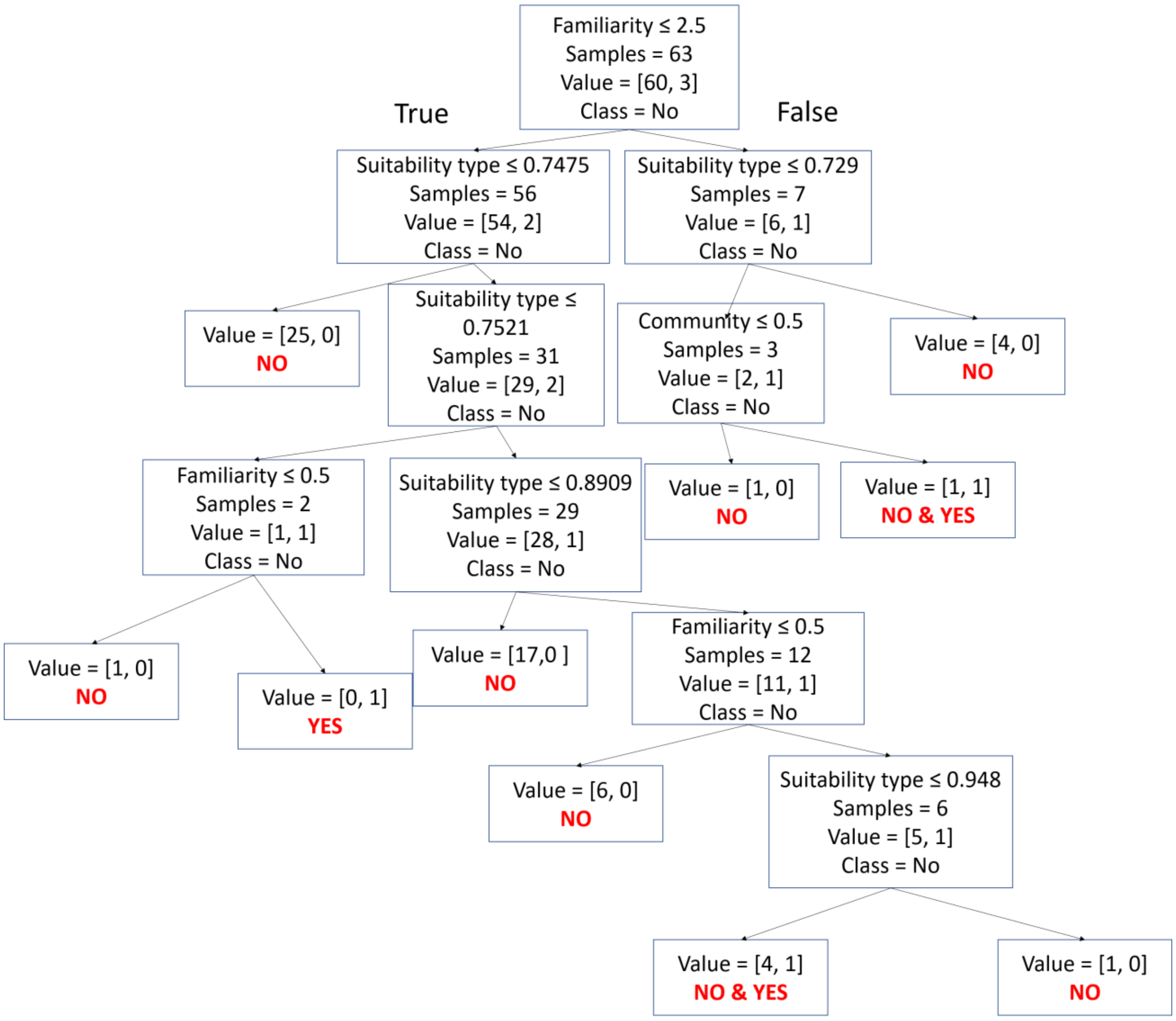


Figure 36: User Oili's decision tree.

Figure 36 shows Oili’s decision tree which achieved 94 % accuracy in Table 4. At each parent node, there is a threshold for each feature. For example, at the root node, *Familiarity* is used to divide the tree. If the *Familiarity* score is less than or equal to 2.5, it would navigate to the left side of the tree and if the *Familiarity* is more than 2.5, it would navigate to the right side of the tree.

The *Value* variable indicates that the number of the left side are samples which output *No*, while the number on the right side indicates the samples which are *Yes*. The *Yes* samples indicate that user is attending an event while *No* indicates that the user is not attending the events.

The *class* variable at each parent node indicates the class, either *Yes* or *No* which is the major proportion of the samples at each node. As it can be observed from Figure 36 at the root node, Oili has chosen not to attend to a lot of events.

Some of the leaf nodes has both *Yes* and *No* answers because the decision tree was not able to break down the nodes to more nodes for each class as the ratio of *No* answers is significantly larger than *Yes* answers.

Oili has been interviewed after she completed the survey and she reported that most of the events are not suitable to her as she normally does not participate in such activities. The decision tree seems to have deduced that Oili’s most important decision relates to the suitability type of the event. There are 5 nodes in the decision tree that show that this particular feature is an important factor to Oili.

5.3 Qualitative evaluation

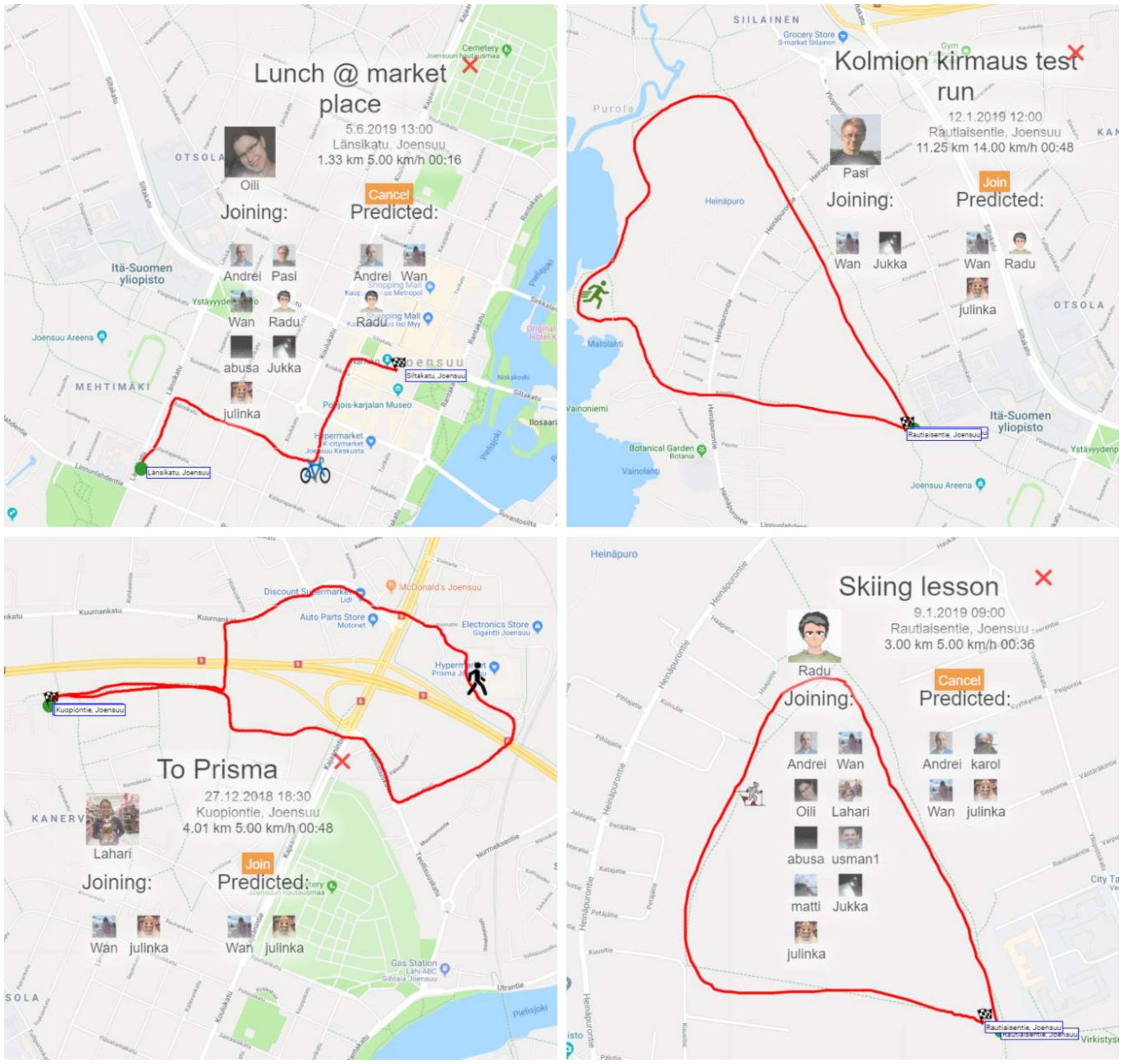


Figure 37: Some events with who is predicted to join them.

An example of Oili’s event in Figure 37 *Lunch @ market place*. Three users are predicted to join the event, but more users have joined. The market place itself is popular during the summer and the social nature of the event is not captured by any of the features.

Another event which is Pasi’s *Kolmion kirmaus test run* for which three users are predicted to join. Two users, Radu and Julinka are wrongly predicted. They have participated in similar events in the past, but they have decided not to maybe because the event is more about speed which might have made it less attractive for the two users. The model does include this feature but was not enough to predict this. The model that was used to predict each user's attendance is the personalized decision tree since gave the best results compared to other models in the experiments.

The third event named *To Prisma* is hosted by Lahari. The users predicted to join is 100% correct. Zhentian (Wan) and Julinka have Prisma in their interests and were thus predicted to join.

The fourth event is Radu's event *Skiing lesson*. Andrei, Wan and Julinka are predicted to join which are correct. User Karol however is wrongly predicted. This might be because Karol is a regular skier but decided not to come as this event is for beginners. Karol has plenty skiing data and this maybe a reason why he is predicted to join. Other unpredicted users are also joining the event. One reason could be that these users are new in Joensuu and may not have sufficient data to conclude that they will be joining the event for prediction. These users are the cold-start users who do not have enough data.

6. Conclusions

In this study, the Mopsi event system has been described. It is sub-part of an application called Mopsi. During this thesis project, the event system[4] has been implemented and added to Mopsi.

Mopsi has data that can be utilized to build a recommendation system for activity events. For building a recommendation system, user profiles have been created from different user's data. Data such as meetings, trajectories, photos and services have been used to create user profiles. The user profiles give us different information such as top friends of a user, activity of the users in different time of day, week and season. It also gives information such as the movement preferred by the user. It also gives information such as services used by the user.

Volunteers have created real events during 1.9.2018 – 1.9.2019. A survey has been conducted for experiments. The survey tool is a simple variation of the Mopsi Events system with the difference that volunteers had to answer **Yes** or **No** to different events. This allows us to have ground truth whether users would participate to an event or not.

With the ground truth obtained from the survey, a 9-dimensional feature vector were created from the user profiles. The feature vectors have then been split 50-50 for training and testing. It has been done in a way that both training and testing set contains events from all the seasons. Different classifiers have been used to train the data. We then created a general and a personalized model for each user. General model contains the training set from all the users while the personalized model contains training set for each user. General model works best with some users while a personalized model works better with other users. Using all the classifiers which provide the best accuracy for each user, we report an average accuracy of 80%. This accuracy is better than 50 % which is a decision by random chance. To compare the accuracy achieved in this study, we refer to two other event recommendation systems. In [18], the authors report a mean average precision of 55 % for recommending scientific talks as events. In [23], an accuracy of 76 % was reported for recommending services to users.

With the unique data that is in Mopsi which includes user trajectories, social meetings, photos taken, and services visited by the user, we learned how to prepare the data to extract feature vectors from them. We learned that each user should have their own model which reports the best accuracy and precision for that user. But for 25 % of the users, personalized decision tree model provides the best accuracy and precision. Personalized decision tree also provides the best accuracy compared to other models with a reported 76 % accuracy with 50 % precision which is also the highest. This is because the data that was prepped and the feature vectors that was extracted from the data is well structured. We learned that decision tree provides good accuracy and precision when the data has a good structure. However, some classifiers provide better accuracy and precision for some users. We learned that there is no single classifier that can provide good prediction results for all users. In this study, we provide a skeleton on how to extract features from location-based data and use classifiers to predict or recommend items to users. In this study, items are in the form of activity events.

For future work, the following improvements can be considered. First, the trajectory snapping algorithm could be improved. There are artefacts that are produced by the algorithm, which does not work as intended when the trajectory is not close to the road network. More meaningful feature vectors could be added to make the recommendation system more accurate. An example of an additional feature could be the location of where the photos taken by the user could be used. This feature could indicate the location that is liked by the user.

4 http://cs.uef.fi/mopsi/events/

REFERENCES

1. Wellman, B., 2001. Computer networks as social networks. Science, 293(5537), pp.2031-2034.
2. Walter, F.E., Battiston, S. and Schweitzer, F., 2008. A model of a trust-based recommendation system on a social network. Autonomous Agents and Multi-Agent Systems, 16(1), pp.57-74.
3. Park, M.H., Hong, J.H. and Cho, S.B., 2007, July. Location-based recommendation system using bayesian user's preference model in mobile devices. In International conference on ubiquitous intelligence and computing (pp. 1130-1139). Springer, Berlin, Heidelberg.
4. Sarwar, B.M., Karypis, G., Konstan, J.A. and Riedl, J., 2001. Item-based collaborative filtering recommendation algorithms. Www, 1, pp.285-295.
5. Pazzani, M.J. and Billsus, D., 2007. Content-based recommendation systems. In The adaptive web (pp. 325-341). Springer, Berlin, Heidelberg.
6. Fränti, P., Mariescu-Istodor, R. and Sengupta, L., 2017. O-Mopsi: Mobile orienteering game for sightseeing, exercising, and education. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), 13(4), p.56.
7. Fränti, P., Chen, J. and Tabarcea, A., 2011, May. Four Aspects of Relevance in Sharing Location-based Media: Content, Time, Location and Network. In WEBIST (pp. 413-417).
8. Tabarcea, A., Gali, N. and Fränti, P., 2017. Framework for location-aware search engine. Journal of location Based services, 11(1), pp.50-74.
9. Takeuchi, Y. and Sugimoto, M., 2006, September. CityVoyager: an outdoor recommendation system based on user location history. In International Conference on Ubiquitous Intelligence and Computing (pp. 625-636). Springer, Berlin, Heidelberg.
10. Bellotti, V., Begole, B., Chi, E.H., Ducheneaut, N., Fang, J., Isaacs, E., King, T., Newman, M.W., Partridge, K., Price, B. and Rasmussen, P., 2008, April. Activity-based serendipitous recommendations with the Magitti mobile leisure guide. In Proceedings of the sigchi conference on human factors in computing systems (pp. 1157-1166). ACM.
11. Yang, F. and Wang, Z., 2009. A mobile location-based information recommendation system based on GPS and WEB2. 0 services. database, 7, p.8.
12. Levandoski, J.J., Sarwat, M., Eldawy, A. and Mokbel, M.F., 2012, April. Lars: A location-aware recommender system. In 2012 IEEE 28th international conference on data engineering(pp. 450-461). IEEE.
13. Zheng, Y., Zhang, L., Ma, Z., Xie, X. and Ma, W.Y., 2011. Recommending friends and locations based on individual location history. ACM Transactions on the Web (TWEB), 5(1), p.5.

14. Savage, N.S., Baranski, M., Chavez, N.E. and Höllerer, T., 2012. I'm feeling loco: A location based context aware recommendation system. In Advances in Location-Based Services (pp. 37-54). Springer, Berlin, Heidelberg.
15. Yoon, H., Zheng, Y., Xie, X. and Woo, W., 2012. Social itinerary recommendation from user-generated digital trails. Personal and Ubiquitous Computing, 16(5), pp.469-484.
16. Clements, M., Serdyukov, P., de Vries, A.P. and Reinders, M.J., 2011. Personalised travel recommendation based on location co-occurrence. arXiv preprint arXiv:1106.5213.
17. Waga, K., Tabarcea, A. and Fränti, P., 2012, October. Recommendation of points of interest from user generated data collection. In 8th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom) (pp. 550-555). IEEE.
18. Minkov, E., Charrow, B., Ledlie, J., Teller, S. and Jaakkola, T., 2010, October. Collaborative future event recommendation. In Proceedings of the 19th ACM international conference on Information and knowledge management (pp. 819-828). ACM.
19. Qiao, Z., Zhang, P., Zhou, C., Cao, Y., Guo, L. and Zhang, Y., 2014, June. Event recommendation in event-based social networks. In Twenty-Eighth AAAI Conference on Artificial Intelligence.
20. Macedo, A.Q., Marinho, L.B. and Santos, R.L., 2015, September. Context-aware event recommendation in event-based social networks. In Proceedings of the 9th ACM Conference on Recommender Systems (pp. 123-130). ACM.
21. Neves, A.R.D.M., Carvalho, Á.M.G. and Ralha, C.G., 2014. Agent-based architecture for context-aware and personalized event recommendation. Expert Systems with Applications, 41(2), pp.563-573.
22. Cena, F., Likavec, S., Lombardi, I. and Picardi, C., 2014. Should i stay or should i go? Improving event recommendation in the social web. Interacting with Computers, 28(1), pp.55-72.
23. Huijuan, L., Kejie, C. and Bai, L., 2007, May. Bayesian network based behavior prediction model for intelligent location based services. In 2007 2nd IEEE Conference on Industrial Electronics and Applications (pp. 703-708). IEEE.
24. Fränti, P., Waga, K. and Khurana, C., 2015. Can Social Network Be Used for Location-aware Recommendation?. In WEBIST (pp. 558-565).
25. Fränti, P., Mariescu-Istodor, R. and Waga, K., 2018, June. Similarity of Mobile Users Based on Sparse Location History. In International Conference on Artificial Intelligence and Soft Computing (pp. 593-603). Springer, Cham.
26. Mariescu-Istodor, R., & Fränti, P., 2017. Grid-based method for GPS route analysis for retrieval. ACM Transactions on Spatial Algorithms and Systems (TSAS), 3(3), 8.
27. Rezaei, M. and Franti, P., 2018. Real-Time Clustering of Large Geo-Referenced Data for Visualizing on Map. Advance in computer and electrical engineering, 18(4), pp.63-74.
28. Yin, H. and Wolfson, O., 2004, June. A weight-based map matching method in moving objects databases. In Proceedings. 16th International Conference on Scientific and Statistical Database Management, 2004. (pp. 437-438). IEEE.

29. Larose, D.T., 2005. An introduction to data mining. Discovering Knowledge in Data, John Wiley & Sons Publication, Hoboken, New Jersey, USA, pp.1-25.
30. Bishop, C.M., 2006. Pattern recognition and machine learning. springer.
31. Rish, I., 2001, August. An empirical study of the naive Bayes classifier. In IJCAI 2001 workshop on empirical methods in artificial intelligence (Vol. 3, No. 22, pp. 41-46).
32. Chen, M., Xu, M. and Franti, P., 2012. A fast $ o (n) $ multiresolution polygonal approximation algorithm for GPS trajectory simplification. IEEE Transactions on Image Processing, 21(5), pp.2770-2785.
33. Blewitt, R.L., Maptuit Corp, 2004. Alternate routes generation. U.S. Patent 6,732,048.
34. Waga, K., Tabarcea, A., Chen, M. and Fränti, P., 2012, October. Detecting movement type by route segmentation and classification. In 8th International Conference on Collaborative computing: networking, applications and worksharing (CollaborateCom) (pp. 508-513). IEEE.
35. Mariescu-Istodor, R and Fränti, P., Detecting user actions in location-based systems, Int. Conf. on Location Based Services (LBS), Adjunt proceedings, Zürich, Switzerland, 1-6, January 2018.