Data Science Hackathon in Fitness data
(ENDOMONDO)

Chunxiang Li
University of Helsinki
Chunxiang.li@helsinki.fi

Han Xiao
University of Helsinki
han.xiao@helsinki.fi

Huibin Shen
Aalto University
huibin.shen@aalto.fi

Hongyu Su
Aalto University
hongyu.su@aalto.fi

Abstract

This report describes a personalized track recommendation system built by CH_3 team in Aalto Data Science Hackathon 2015. The developed recommendation system is able to analyze all available track information from ENDOMONDO and compute an ideal track which satisfies user profile or various input conditions. The system can also visualize the computed track both as a rout on the map and as a preview movie comprised by a collection of pictures along the track. In addition, we are able to extract information from the social network (e.g., TWITTER) base on time and GPS locations. Therefore, the computed track is annotated with enriched context. The annotation not only enables user to review the activities along the recommended tracks, it also provides opportunities to analyze and recommend tracks based on the contextual information (e.g., keyword searching, profile matching, sentiment analysis).

1 Introduction

Credits should be given to the members in Table 1.

Name	Affiliation	Student number
Chunxiang Li	U.H.	014027998
Han Xiao	U.H.	014343753
Huibin Shen	Aalto	381606

Table 1: Team members that need credits.

2 Data preprocessing

2.1 Preliminaries

ENDOMONDO dataset contains the information of a collection of 68479 sport tracks (e.g., walking, running, cycling). We use $\mathcal{T} = \{T_i\}_{i=1}^{n}$ to denote the collection of tracks where we have $n = 68479$. Each track $T_i \in \mathcal{T}$ is represented as a set of geographical points on the map (GPS locations) ordered by timestamps defined as

$$T_i = \{(x_{i,1}, y_{i,1}, t_{i,1}), \cdots, (x_{i,m_i}, y_{i,m_i}, t_{i,m_i})\},$$

where $x_{i,j}$ is the longitude, $y_{i,j}$ is the latitude, $t_{i,j}$ is the timestamp when $(x_{i,j}, y_{i,j})$ is recorded by the tracking devices, m_i is the total number of pointed recored for track T_i. In addition, we have $t_{i,j} < t_{i,k}$ when $j < k$.

Each track T_i is then represented as an undirected graph $G_i = (V_i, E_i)$ defined on a 2D surface. The vertex set $V_i = \{(x_{i,j}, y_{i,j})\}_{j=1}^{m_i}$ includes all GPS locations of track T_i. There exist an undirected edge $e_{i,k}$ between vertex $v_{i,k}$ and $v_{i,k+1}$ $\forall k \leq m_i - 1$. As a result, the edge set is defined as a collection of undirected edges $E_i = \{e_{i,k}\}_{k=1}^{m_i-1}$.

2.2 Global track graph

The goal is to analyze all available track information and recommend an ideal track for individual end user. We realize that it is feasible to maintain all individual track information T_i and V_i due to the following reasons

- It is computationally expensive to analyze 68479 tracks each time when we perform the recommendation or searching algorithm. In particular, the data files for all tracks take approximately 2 Gigabyte.
- It is difficult to measure global performances along the tracks (e.g., average/min/max speed on a location, number people running over a location) by analyzing track information separately.
- Individual track information is not an accurate representation of the track due to the nature of the GPS devices and the tracking algorithms. In other word, several tracks might be performed on the same street but have different trajectories in terms of GPS location data.

We summerize and represent all track information via a global track graph $G^{\mathcal{T}} = (V^{\mathcal{T}}, E^{\mathcal{T}})$.

- The world map is partitioned into small blocks of $10m \times 10m$. Each block is represented by the GPS location (x, y) of its centre, and is a potential vertex in the global track graph $v^{\mathcal{T}} = (x, y) \in V^{\mathcal{T}}$.
- Then we map the set of vertices $v_{i,j} \in V_i$ in the graph G_i of track T_i to $V^{\mathcal{T}}$ by assigning $v_{i,j}$ to the nearest vertex $v^{\mathcal{T}} \in V^{\mathcal{T}}$. Essentially, each vertex $v^{\mathcal{T}}$ corresponds to a collection of vertices from different tracks
$$v^{\mathcal{T}}{}_k = \{v_{i,j}\}_{i \in \{1,\cdots,n\}, j \in \{1,\cdots,m_i\}}, k \in \{1, \cdots, |V^{\mathcal{T}}|\}$$
- There exists an undirected edge $e^{\mathcal{T}} = (v^{\mathcal{T}}{}_p, v^{\mathcal{T}}{}_q) \in E^{\mathcal{T}}$ if there is an edge between $v_p \in v^{\mathcal{T}}{}_p$ and $v_q \in v^{\mathcal{T}}{}_q$. Mathematically, the edge set $E^{\mathcal{T}}$ can be defined as
$$E^{\mathcal{T}} = \{(v^{\mathcal{T}}{}_p, v^{\mathcal{T}}{}_q) | \exists (v_p, v_q) \in E_i, \forall v_p \in v^{\mathcal{T}}{}_p, v_q \in v^{\mathcal{T}}{}_q, i \in \{1, \cdots, n\}, p, q \in \{1, \cdots, |V^{\mathcal{T}}|\}\}$$

As a result, we have a global track graph $G^{\mathcal{T}} = (E^{\mathcal{T}}, V^{\mathcal{T}})$ which summerizes all track information and underlies all possible tracks that can be run by users. The global track graph is a high level abstraction of the data and serves as the input to the recommendation/searching algorithms described in the next section.

2.3 Annotations on the global track graph

We notice that the vertices in the global track graph are essentially GPS locations which not only can be annotated with the summary statistics computed from all track information but also can be annotated with enriched contextual information. Currently, we consider the following annotations on the vertices $V^{\mathcal{T}}$ of the global track graph $G^{\mathcal{T}}$

- We compute various summary statistics from all track information, which include e.g., maximum speed on the current location, minimum speed of the current location, average speed of the current location, popularity of the current location in terms of the number of tracks going through it.
- For each location, we extract the information from social network (TWITTER) and annotate the vertex with bag-of-words features.

City	Vertices	Edges
London	45942	46182
New York	4856	6543
Amsterdam	12357	15862

Table 2: Statistics of the global track graphs.

2.3.1 TWITTER information

Just like a community is characterized by its residents living in that area, a route is definitely can be defined by the persons running, living or walking along it. Along the development of digitalization of life, people would like to share their feelings or experiences through media social tools, such as facebook and twitter. According to the locations of some points in a route, we can get the information of twitters and the corresponding tweets sent from nearby. Therefore, when users would like to browser routes in a certain region or search a routes under some constrains, these social information can be displayed along these routes. In one hand, people can learn more about the routes instead of only from its geographic information but also maybe some feedback about that region. On other hand, furthermore, people would find some potential exercise company or even turn this kind of virtual friend into friend in reality. It may enrich their exercise events meanwhile broaden their social networking.

In the future, it seems can even add some tweets tag on the route allowing the user to follow the twitter user that he/she interets. Another possible way it to emble some other social elements into that map and even make it a potential business benefits for those company along the routes in city, especially for these new stores, they can easiy to advertise.

2.4 Technical details

Three cities are selected for demo (London, New York, and Amsterdam) by defining the city area as a block of $20km \times 20km$ and using the GPS location data of the city centre as the block centre. All tracks within the predefined city block are included in the analysis. We following the procedure described in Section 2.2 to build global track graphs. The statistics of the graphs are shown in Table 2.

3 Algorithm

Most application records the walking, running and cycling routes after the exercise have been done. However, when user move to a new place or travel to a new country, find a good running route will become a problem. In this section, we adpat algorithms in Graph for route recommendation.

We consider a simple case where a user specify the length of the route and we recommend a route that starts and ends at the same position with the length as required. Formally, given a undirected graph $G = (V, E)$ and a starting vertex s, the goal is to find a walk in the graph G of the length d ends at s. We have length information w_e for every edge $e \in E$. In practice, it is possible in G there is no walk of length d starting from s and ending at s, thus we allow the maximum length difference Δ. We may have a lot of tottering in the end, so we sort in the reverse order the walks by the number of different nodes the walk contains and take the first few ones. The simple BFS based algorithm is listed in Algorithm 1.

Algorithm 1 Naïve BFS based search.

Input: $G = (V, E), s, d$
Output: P, the set of walks of length d start and end at s
Init $P = \{\}, Q = \{\}$
$Q.add(s)$
while Q is not empty **do**
  $u = Q.pop()$
  for $v \in neighbors(u)$ **do**
    if $v == s$ and $depth(v) - d \leq \Delta$ **then**
      $P.add(v)$
    else if $depth(v) < d + \Delta$ **then**
      $Q.add(v)$
    end if
  end for
end while
Sort the walks in P by the number of different nodes in reverse order.

We can speed up the search by joining two walks into one. The idea is simple, when we have all the nodes in the next layer of the search tree, we test for all pairs of nodes if they are the same. For the nodes on different paths of the search tree, if they are the same, we can concatenate the two paths into one, the detail is listed in Algorithm 2.

Algorithm 2 Naïve BFS based search.

Input: $G = (V, E), s, d$
Output: P, the set of walks of length d start and end at s
Init $P = \{\}, Q = \{\}$
$Q.add(s)$
while Q is not empty **do**
  $C = \{\}$
  for $u \in Q$ **do**
    for $v \in neighbors(u)$ **do**
      if $depth(v) \leq d + \Delta$ **then**
        $C.add(v)$
        $Q.add(v)$
      end if
    end for
  end for
  for $u \in C$ **do**
    for $v \in C$ **do**
      if $u == v$ and $depth(u) + depth(v) - d \leq \Delta$ **then**
        $P.add$(Walk by joining u and v)
      end if
    end for
  end for
end while
Sort the walks in P by the number of different nodes in reverse order.

One of the problem for the above algorithms are the computational efficiency. The problem of finding a path of fixed length in a graph is NP-hard. Since each of the blocks in our data is about 10m to 20m, to find a route of 3km needs 150 to 300 blocks and the search tree is huge. In practice, we could only find route round 1km in reasonable time. As a result, to find longer route, we searched the historical routes around the area for every user and suggest the most similar one in terms of length.

Based on historical routes datasets, we could get the information about the number of visits of every places and take that as popularity. This open the possibility to recommend routes with most

popularity. We add this function as a post-processing of the routes in the set P which is just sorting based on the sum of number of visits of every places in a route. We show such an route at `http://ch3.strikingly.com/blog/recommend-route-with-popularity`.

We also retrieved twitter data related to places in the routes. By doing so, we open the possibility to add social information and even sentimental information in the route recommendation. We show such an route at `http://ch3.strikingly.com/blog/routes-with-social-info`.

4 Visualization

Three visualization tasks are considered in our work:

1. Routes with tweets as exampled in Figure 1
2. Popular routes in different cities as exampled in Figure 2
3. Global workout statistic as exampled in Figure 3
4. Recommended route preview movie as exampled in this link

For the first three tasks, we used Google Maps API [1]. For the last one, we use Google Street View API [2]. Some examples are shown below:

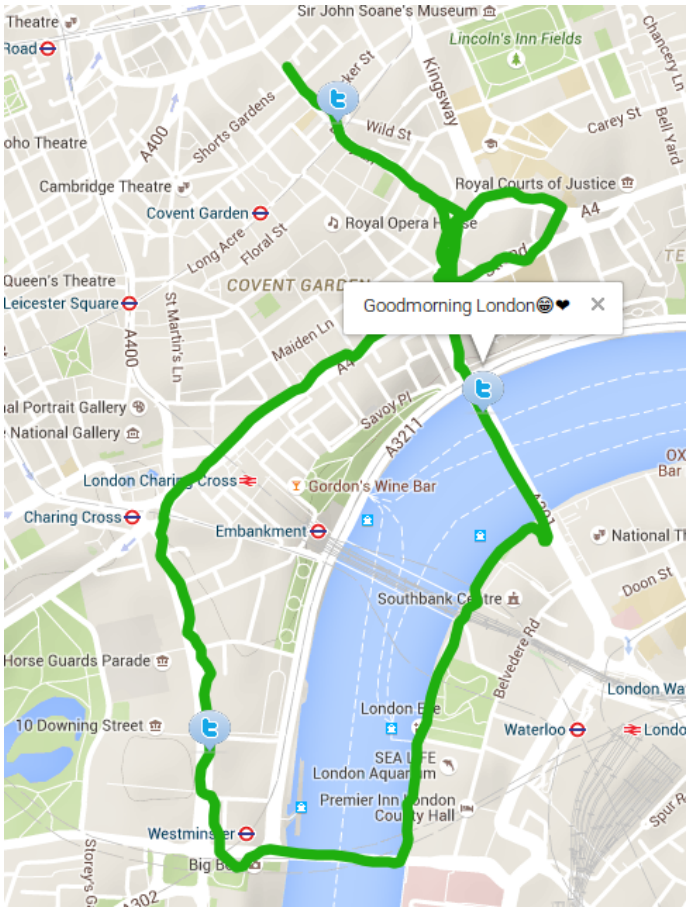

Figure 1: Some running route in London annotated with location-specific tweets

[1] https://developers.google.com/maps/
[2] https://developers.google.com/maps/documentation/streetview/

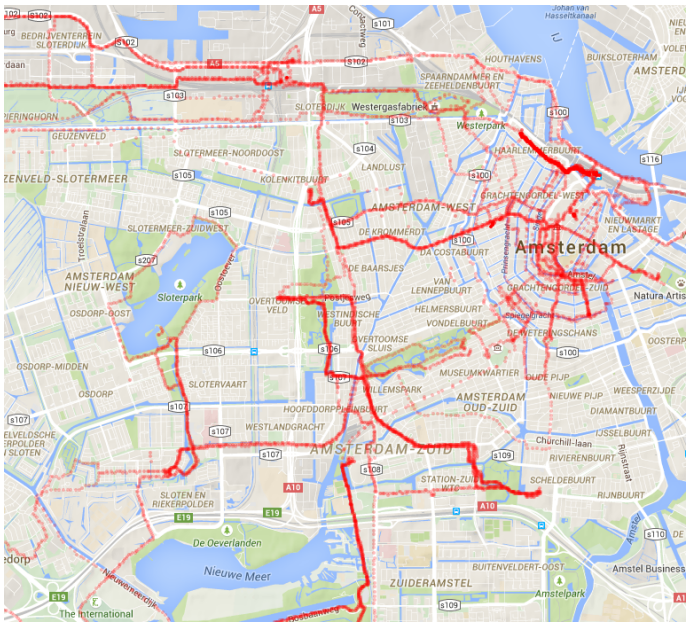

Figure 2: Popularity of running routes in Amsterdam: paths with bolder color are of higher popularity

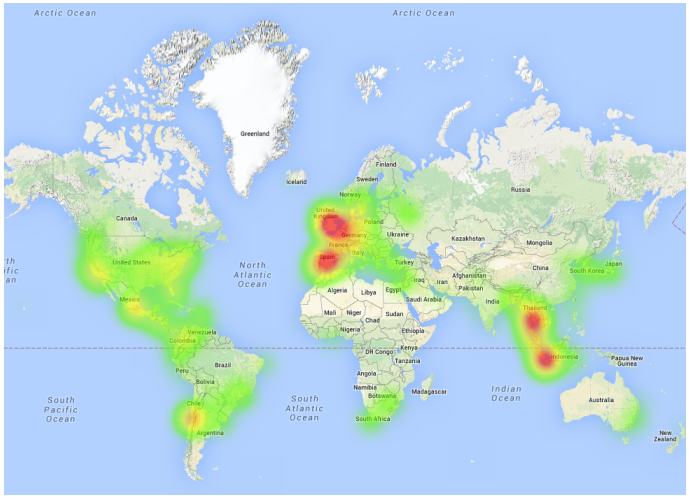

Figure 3: Global-wise workout distance heatmap

5 Conclusion and discussion

We have developed a personalized track recommendation engine. The developed system can analyze all available track information, compute and recommendation a ideal track for the end user which can either match the user profile or satisfy the constraints given by the end user. We realized the technical difficulties during the development of the system for which it is not feasible to maintain all individual track information. To solve the problem, we construct a global track graph by mapping and analyzing individual track information which serves as the unified input to out learning and searching algorithms. Additional, we mathematically formulate the recommendation problem as a graph searching problem and propose several running algorithms to solve the problem. We also visualize the result both on a map and as a short movie comprised with pictures extracted from the track. Finally, we combine the information from social network which not only adds enriched con-

textual information on the computed tracks but also shows the potential of building a sophisticated system in the future which is based on e.g., context searching/matching/learning.