TraceMySteps - Visualizing And Analyzing Personal Geolocation Data

Pedro João Cordeiro Francisco
Instituto Superior Técnico
Lisbon, Portugal
pedro.francisco@tecnico.ulisboa.pt

ABSTRACT

As the use of GPS tracking devices such as smartphones keeps growing, large amounts of data are produced. People are starting to use these volumes of data as means to gain insights about their mobility - places they have been to, distances travelled, routes taken, etc. It is hard to analyze this type of data and observe patterns with personal relevance. In this dissertation we present our crafted solution that allows people to understand their spatio-temporal information, with focus on the personal side of that information. In order for that to happen, our interface presents a number of visualizations that users can customize and filter, so that they can analyze their data (spanning from days to years) thoroughly. Finally, an evaluation with users showed that people could use and understand the system in its whole, thus proving the initial objectives were achieved.

Author Keywords

Personal semantics; spatiotemporal information; scalable visualization; movement data; data visualization.

INTRODUCTION

Nowadays, with the exponential growth in the usage of GPS tracking devices such as smartphones, smartwatches and tablets, massive amounts of data derived from that usage are generated. That data is usually associated with mobility, behaviors and patterns of animals and people. This situation leads to an increasing number of people carefully exploring the information they have available in order to plan their life and activities.

Lifelogging can be described as a practice in which people track their personal data. This data is generated by the person's physiological and social activities such as running, eating, traveling and working. As we will explore, this is an area that has started to gain interest from many people in the last couple of years, with an increase in blogs, websites and applications that offer various solutions and insights about this subject. Geographic lifelogging is a field of lifelogging in which a user tracks his mobility data - paths, trips, places, etc.

Most approaches to this mobility tracking do not cope with the personal aspect of data. This demonstrates the need for a universal solution that all types of users can use and further enhance their lives. The outline here is that collecting spatio-temporal data has become a frequent and simple process, but analyzing it and deriving information with personal significance is still a blurred area. So our aim is to design and create an application for users to thoroughly understand and analyze their spatio-temporal information, with focus on the personal aspects of the data regarding travels, stays and places the user visited.

To achieve this goal, we created a system capable of processing and storing a user's mobility data - it handles previously recorded GPS tracks - together with visualizations that include and present the personal semantic notes associated with those routes or locations.

First we needed to know what factors, to be observed in our solution, were important for potential users. Next, we had to understand which type of user mobility data we needed, and how to process it. Then we planned the architecture of our system, designed and implemented the back-end and front-end using technologies such as Flask, AngularJS and PostgreSQL. We analyze the scalability for both front-end and back-end.

Finally, we proceeded to execute user tests. Besides the validation of our objective, the focus was to get the global feeling of the solution, to understand user satisfaction and to evaluate the way users deal with the personal semantics.

RELATED WORK

In this section we explore existing works in the field of visual analytics, regarding the visualization of geospatial data. The section is divided in three subsections: Representation using tracks and timeline, representation using heatmaps and representation using space-time cube.

Representation using tracks and timeline

(Jeon 2014) uses an interface composed of a timeline and a calendar. A map contains circles that show the most common areas. (Attfield 2014), (Saraf 2014), (Chen 2014), (Hundt 2014) and (Wood 2014) use data sources such as GPS tracks. These works present an interface containing a map and a timeline. The maps show polygons to represent areas (Chen 2014) or colors for path densities (Attfield 2014). The remaining use GPS coordinates to show points of interest. (Saraf 2014) includes an interface for queries. The timelines for these works show the temporal distribution of time spent by people, with different colors for different places.

(Guo 2014) presents the data using dodecagons. It is mainly used to analyze traffic problems.

(Thudt 2013) uses a map-timeline: each map segment has its size determined by the duration on the timeline, while showing a certain section of the map.

(Krueger 2014) uses a map view with a lens tool, to zoom and avoid visual clutter. A timeline is used, showing with colors and icons the places for each hour of the day.

(Siddle 2014) shows tracks on map, with the direction of the track and frequency of it reflected in the thickness of the presented tracks. Aprilzero and Jehiah14 are blog entries that show a timeline with icons for the places and transportation methods, and GPS tracks on map categorized by colors, respectively.

(Andrienko 2011) and (Andrienko 2013) use a time graph (like a plot graph) and time bars, to show the movement of animals and GPS/GSM tracks, respectively.

(Larsen 2013) uses a spiral visualization to show GPS data as a time-series. It builds on top of a clock-dial metaphor with a circle corresponding to a time span) Data points can be shown using different color coding.

TrajRank and OD-Wheel, both by (Lu 2014) use a map visualization with tracks and polygons, respectively. OD-Wheel uses a cluster wheel to allow trip frequency between time intervals. TrajRank uses a horizon graph view that shows the distribution of trajectories over a day, classified from low traffic (green) to bad traffic (red).

(Wang 2014) uses a 2D heatmap to visualize trajectory curvature, a timeline as a 3D terrain and a graph to show temporal distribution of trajectories.

(Lundblad 2013) introduce choropleth maps and other visualizations such as scatter plots and bar charts as means of using them for storytelling.

(Liu 2011) proposes a view with three layers: heatmap, location, custom. Then, a road view shows all trajectories passing through a road segment. All views are linked.

(Otten 2015) proposes a visualization based on a map network. The places are shown as circular map extracts. A slider allows the selection of a time period and a semantic zoom lens allows for detail exploration on each view.

Most of these works have scalability issues, cluttering the views when the datasets are too big. The data collection methods are based on GPS. Most works present their interface as a mix of tracks in map and timeline. Regarding preprocessing, many works use cluster algorithms. Finally, a minority of these works approach the personal semantics issue. This is the problem we are addressing.

Representation using heatmaps

Works like (Spek 2010) or (Liu 2011) use heatmaps to show the highest density of people or vehicle movement in streets and roads. This can be used to identify points of interest such as stores or parking lots.(Stotz 2014) creates a simple heatmap of his GPS tracks, to show the most used transportation method and the roads with higher density of tracks - the track color is brighter.

Their biggest advantages are the simplicity in observing the mobility patterns of one person and their scalability. The drawback is the lack of detail in the maps, besides the patterns. They need some descriptive view of the data.

Representation using space-time cube

Following the guidelines of (Kraak 2003), a basic space-time cube consists of a cube with a representation of geography on its base), while the cube's height represents time. Typically it contains space-time paths for individuals or groups. (Andrienko 2011) and (Andrienko 2013) use space-time cubes to show GPS trajectories and animal movement, respectively. Space-time cubes seem outdated and have severe scalability problems: it is hard to identify and explore big amounts of data as the cube gets cluttered.

DATA COLLECTION

Based on our already stated objective to design a visualization that allows users to understand and analyze their spatio-temporal information, with focus on personal semantics, we have to collect personal mobility data to investigate if we achieved this objective. This data comes from two sources: the first one is our personal mobility data, collected through the use of GPS-enabled devices (smartphone) and stored in files; the second one are notes made by users about their tracks. These notes allow users to give a personal semantic context to their tracks.

GPS tracks

Users must first collect that said information by collecting GPS data, ideally on a daily basis. The used device for the data collection is not important, although nowadays is far easier to download a mobile application (available for various smartphones and operating systems) for that purpose. What is imperative here is that the GPS data collection has to produce some file output that we can use as input for our application, namely its back-end.

Various smartphone apps for GPS allow the exporting of the recorded information to a file or a set of files, in a common format among them: GPX. GPX stands for GPS Exchange Format, an XML schema that stores GPS data and allows its use in other software applications.

Each file that is generated by these recording applications is called a track. It contains location data - latitude, longitude, elevation, a timestamp for each point.

Personal semantic notes

As described in the previous subsection, we now know where the spatio-temporal data that will be used to achieve our main objective comes from. Now we need the personal meaning of that information, to help us fully achieve our main objective. To create these personal semantic notes, we will adopt the LIFE format created by Daniel Gonçalves, available on GitHub (https://github.com/domiriel/LIFE) and exemplified on figure 1. This library contains helper functions that will later allow us to process the semantic notes on our back-end.

```
--2016_01_15
@UTC
0000-0915: home
0924-0930: seixalinho station
1000-1008: cais do sodre station
1009-1024: cais do sodre station->saldanha station
1025-1027: saldanha station
1030-1743: INESC
1746-1750: saldanha station
1751-1815: saldanha station->cais do sodre station
1815-1834: cais do sodre station {Waiting for the boat trip back to Montijo. The boat was late}
1920-2359: home [dinner]
```

Figure 1. Semantic notes for a single day.

This format allows the association of labels (meaningful names to the user, such as "grandmother's home", for

instance) to specific coordinates of the data collected by the GPS devices. This means the user can later search and visualize his personal semantic information on the application front-end, using the associated labels that represent the real world coordinates.

The base goal of this format is to assign a semantic meaning to a certain time interval. In the context of our work, it is mainly used to assign a name to the time a user was indoors. As we will see next, it can also be used to assign a name to a travel (underground or bus, for instance).

We can consider the time intervals present in these semantic notes to be the complement of the time intervals in which we have GPS tracks recorded.

Real world data problems

The following problems are related to erroneous behavior of the GPS devices or created by the user during operation of them: dataset size, GPS accuracy, cold/start end, loss of signal and low battery capacity.

Other problems arise during the creation of a user's personal semantic notes: multiple meanings, user forgetfulness, and adapting to real world changes (places can change names and a name can refer other places).

Related to both data types: privacy and user burden.

Solving real world data problems

There is the need to lower the dataset as much as possible, while maintaining the necessary files and the overall spatial information intact.

To reduce the number of points we chose to use the Ramer-Douglas-Peucker algorithm (RDP), since it is a well-known line simplification algorithm and one of fastest. It also keeps a sufficient quantity of points so that little to no data is lost. To apply the RDP algorithm to the tracks and create the new, simplified versions of them, we will use the implementation provided by Tkrajina (https://github.com/tkrajina/gpxpy).

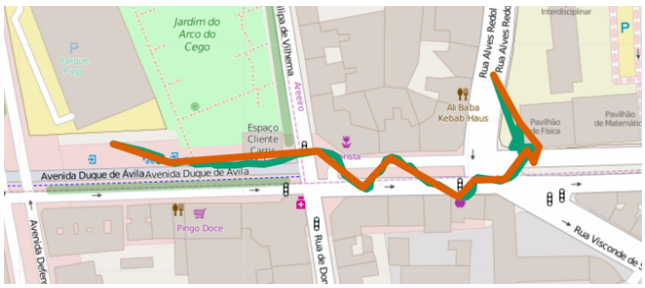

Figure 2. Comparing an original track (green) with a RDP-simplified track (orange)

To check the performance of the algorithm, we compared two versions of a single track. The original track contained 153 points, while the simplified version contained 13 points. It is a 12 times reduction in the number of points. Figure 2 also compares the information shown by both tracks. We can see that despite the number of points is reduced, the path is still the same.

Regarding battery capacity, one solution can be the use of a spare battery or a power bank.

About adaptation to real world changes, every time a location changes name, a mechanism is needed to inform all the stored locations referring to that place that they should also change names.

In regard to privacy, our application is not made with the intent of sending data to the internet. However, as future work, it may need some privacy towards the existence of multiple users (login).

User survey - which factors are relevant

Even with these issues, we still have to know which factors are relevant for the users. So we decided to create a survey. The survey was made using the Google Forms platform and was published and distributed via friends and colleagues through social media. After a short introduction to the work that we will do, users considered the following case: during an interval of several months or years, they collected GPS data of the places they visited, paths they took to get there and temporal data associated to it (date, hour, etc.). Now, they could explore that data in order to better understand their mobility patterns. Then, they answered 16 questions in a scale from 1 to 5.

After the questions were made, we decided to group them. We used the topics of time, places, routes, distances and patterns. We made a statistical analysis, starting with the removal of invalid answers and outliers.

We observed that time is the topic that users find more critical. The next one is the routes they take to and from their destinations. The third most critical topic is the places they visit. Distances and patterns (with focus on, for instance, distances covered) are the less critical topics.

ARCHITECTURE

The scenario in which our system will be used is straightforward: a user will utilize his smartphone (with GPS capabilities) on a daily basis, recording his paths when moving from one place to another, stopping the recording each time he enters a place. Then, the collected tracks will be complemented with personal semantic notes that give meaning to the places and trips the user made. By processing these two types of data on our application, the user will be able to analyze his personal mobility data. The designed architecture for our solution, as depicted in figure 3, we adopted a client-server model. This was attained through the use of a REST between the front-end and the back-end. The RESTful state was achieved by using a Python based web server on the back-end, that handles the requests from the client side. We used technologies such as Flask-RESTful, a Flask extension that allows the creation of REST API's, Psycopg, a PostgreSQL adapter for Python, and PPyGIS, an extension for Psycopg that adds support for geometry objects. A database will be fed with the GPX files and personal semantic notes.

The front-end consists of a manager, the visualizations and a bus. We decided to use the AngularJS framework to build it. The data manager acts as a broker: sends the requests made by the user to the server and then receives the information, sending it to the connected visualizations to be displayed. We use D3, a Javascript library which helps

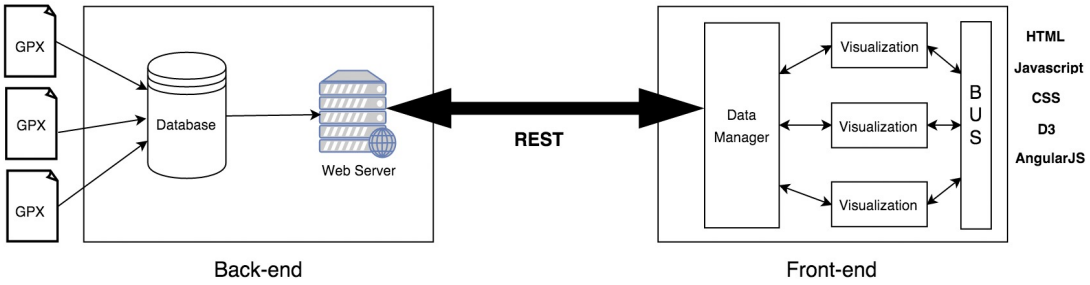

Figure 3. Solution schema

creating dynamic and interactive data visualizations in web browsers. For some of the visualizations we will also use Leaflet, and some of its plugins. Finally, there is a bus that links all the active visualizations. This bus was planned to be implemented using the Observer Design Pattern in Javascript. But as we will see, we will make use of a tool that AngularJS provides us in order to achieve the same result. This bus will allow all visualizations to be linked.

BACK-END

After understanding the architecture of our solution, here we will present our approach regarding the back-end.

Data-model

The data model is a SQL schema that contains the instructions for the creation of the tables we will be using to store our data. Note that its base version was not created by us and we just adapted some tables by adding new columns, in order to store specific data related to our solution. Also, in our case, one of tables is not being used.

We have four tables: *locations*, that stores the name and geographic location of a place; *stays*, that stores the label, dates and times of stays; *trips*, that stores the start and end locations of a trip, its date, timestamps and geographic points that bound it. The fourth table, *trips_transportation_modes* is not used.

Track processing

So as to start populating the database and its tables, we first need to apply the Ramer-Douglas-Peucker (RDP) algorithm to the collected tracks. We use a script that for each track on a folder, divides it in various segments and then applies the RDP algorithm on every segment. The RDP algorithm takes as argument the expected maximum space between track points after the simplification, in kilometers. We found the value of 5 for this argument as optimal in our context.

After each segment is simplified, it is written again on a new file that will contain the same path as the initial track but with those segments simplified. The new file will be named with the date of the track and a number to differentiate tracks within the same day.

Populating the database

We use another script that accesses the directory of simplified tracks, opens and parses it using the aforementioned Tkrajina library, and then loads each track. Figure 4 describes the process of loading tracks. We start by decomposing each track in segments, and each segment in points. Then, we get the timestamp of each point and access the LIFE file in order to know where the user has been at that timestamp. If he had indeed been at a place at that time and a label for start locations is empty, we consider that point a starting location. A similar process is applied for end locations. This means that in the end, after processing the segments and finding the proper points and labels, we can finally insert a trip in the database.

Also, in our loading process for each track, we observe that if a location is not a starting or end point of a trip, we consider it as a single location and in order to have more information on the database, we also add it to the *locations* table. The other instance in which we add a location is in the trip insertion. When a location already exists we update the point cluster of the location. Then we determine and update the centroid by calculating the mean of the latitudes and longitudes of the point cluster. If the location does not yet exist, we insert the new location, where the cluster and the centroid are the first collected point. The frequency in which the user visits the location is also calculated and inserted, using the LIFE file.

Then we open a new connection to the database and call a function that parses the LIFE file and populates the database with all the stays and labels of locations that have no tracks available. This procedure allows us to process the semantic notes, independently of the geographic information. This way we make sure that all the stays and locations are correct and also that the database population is consistent with our choice of shifting the importance towards the semantic notes (LIFE file).

We also performed load testing in our solution, in order to observe if it could handle with a lifetime of semantic annotations and personal geolocation data. To achieve that, we replicated our data to match the quantity collected by the author's supervisor (circa 5 years). We used 2812 GPX files. We also assume the collection of about two/three tracks a day. These tracks totaled 608.3 MB of data and contained 4392048 points. We also had to replicate our LIFE file, leading to 1073 different locations.

We started by simplifying the tracks using the RDP algorithm, which resulted in the same number of GPX files. The total file size was reduced to 59.3 MB, with the processing completed in 13 minutes and 38 seconds. This is a size reduction of 90.25%. Albeit it took almost 15 minutes, we think it is an adequate processing time, since in a real world situation a user will not process five years of

data at once. The insertion in the database plus some web server preprocessing took 10 minutes and 24 seconds.

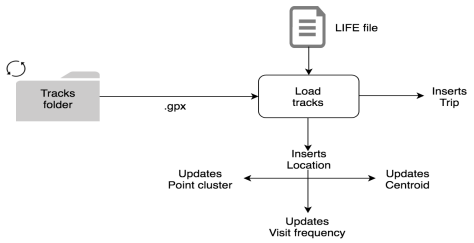

Figure 4. Processing of a folder containing simplified tracks

Web server

We now need an interface that queries the data and feeds it to our front-end. The RESTful state is achieved with the creation of a Python based web server on the back-end. The web server runs in the port 5000 of the *localhost*, and waits for requests. Now what is missing is the ability to connect to the database and feed each visualization with their data.

Multiple visualizations connected to this web server, each with their specific type and format of data, need multiple functions in order to be supplied. Flask offers resourceful routing. So we have as many URL's as there are visualizations. The responses from each resource are always encoded in the JSON format. A set composed by a URL and a Python class that contains a method that produces the particular data for a visualization, is called an endpoint. We have 13 endpoints. Each visualization may have more than one endpoint. We also have two endpoints to feed the dates do the slider. Each endpoint deals with the geographic information stored in the database, or directly handles the LIFE file, producing a formatted JSON response.

With the load testing of our solution, previously explained, we also used that data to test the scalability of both the back-end and front-end. This would show if our solution is scalable enough to handle a lifetime of semantic annotations and personal geolocation data. For the back-end, all of our endpoints correctly handled and processed the requests, returning the responses in an appropriate time interval (averaging less than half a second). Only one endpoint took more than the average time (under 10 seconds). Although considering the large amounts of data used, it is still a slow time to process. To make our back-end fully interactive it would need to get the results in under one second. Nonetheless our solution continues to be responsive, but more preprocessing is needed. Having these tests in mind and except this particular endpoint, overall we believe our back-end is scalable enough to deal with a lifetime of semantic notes and personal geolocation data.

FRONT-END

In this chapter we will explain our approach regarding the front-end of our solution. Due to the context of our line of investigation and the features (RESTful, etc.) we needed for our back-end, the choice of the framework to create it was rather simple. However, for the front-end the decision was not as straightforward. As we have seen in the previous chapter, we decided to use the AngularJS framework to build the front-end, because it is still the industry standard framework, has bigger and better community support, and has a massive amount of add-ons (in the form of directives and modules) that can be used to enhance an application.

User interface

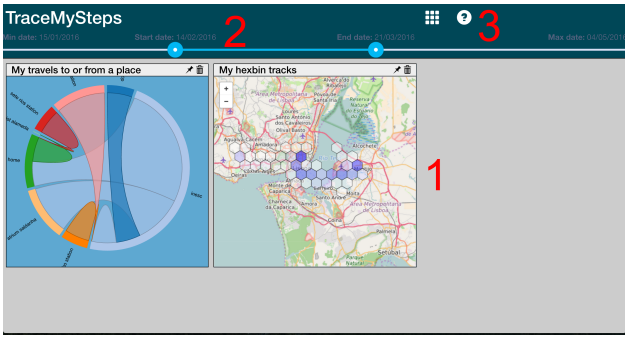

Figure 5. TraceMySteps user interface

Our interface is divided in three areas: visualizations area, timeline slider area and menu area (figure 5).

In the visualizations area is where the added visualizations are drawn. Each visualization is wrapped in a widget. Users can drag the visualization to any position on the grid, using the header of the widget. This grid layout was created using a directive for AngularJS called "angular-gridster" (https://github.com/ManifestWebDesign/angular-gridster). Widgets can also be resized to a maximum or minimum defined size. Users simply need to click and drag any border of the chosen widget. The size of the visualization inside the widget changes accordingly, maintaining its consistency. This grid layout obviously allows for multiple widgets to be active at the same time, with them never overlapping each other.

In the timeline slider area, the time interval in which the user wants to observe the data is adjusted. There are two handles that the user can click and drag to adjust the minimum and maximum dates of the interval. The majority of the visualizations react accordingly to the changes in the slider and redraw to show the corresponding information. Some visualizations provide a more general view, showing only all-time data or data from the last year. Therefore these visualizations do not react to changes in the slider. It is useful to have these two types - this way users can compare certain periods of their life with the general picture, making it easier to find relevant patterns and happenings.

Our interface features two different menus in the menu area, with each one associated to one button. By clicking the question mark button, the "Help" menu slides from the left. By clicking the menu button, the "Add visualizations" menu slides from the right. This menu allows users to add the widgets containing the visualizations they wish to analyze. They can add as many copies of each one as they want. When many copies of one visualization are present, they all react the same, with no information loss. There is also a button to clear all the widgets from the grid.

Data Manager

The intermediate between the visualizations in our interface and the endpoints of our back-end is the data manager. It acts as a broker, receiving requests from the visualizations, gathering the data they need and sending it back to them. To achieve this, we made use of another AngularJS tool: services. In our case we use the HTTP service, one of the most common used services in AngularJS apps. The service makes a request to the server, and lets the application handle the response. So we can consider our data manager to be a service that contains extra verifications so as to correctly cache and send the data.

The data manager receives that first request, and using the name of the endpoint that came with it, connects to the respective web server endpoint URL (ex: "*localhost: 5000/ endpoint_name"*). The web server will respond to the manager with the correct data, already prepared for the visualization. Lastly, that data is sent from the manager to the visualization. A cache mechanism returns the data without having to do any new requests. So we avoid doing an excessive number of requests to the server.

Visualizations

Each visualization is contained in a directive. Together with caching and control variables/mechanisms (such as loading times) all the directives have similar methods: the method to communicate with the data manager; the methods that make a visualization connect and relate with others, where applicable; the methods that deal with the timeline slider, where applicable; and finally the essential methods that draw the visualizations, using D3 or Leaflet. There are nine different directives: two of them use Leaflet maps with an hexagonal binning algorithm in order to show common paths and places; one uses a Leaflet map with a plugin to draw the tracks for a certain time period; a calendar is used to show the stays of the last year, with yearly, daily, monthly and weekly overviews; a stays visualization acts as an all-time calendar heatmap; a bar chart shows both the most visited places and the most time spent in places during a certain time period; an area graph shows for a certain time period, the number of minutes a user spent moving; an arc diagram that shows the all-time most frequent trips; and a chord diagram that shows the trips for a certain time period.

Connecting the visualizations

As the user places the desired widgets on the grid and analyzes each one, the information they show needs to be consistent between each other. By linking the visualizations, users can see the various aspects surrounding their data such as the location associated to a place.

Figure 6 illustrates an example. We can see in the figure that the user selected "*inesc*" in the Chord Diagram. When this happened, besides the normal behavior of the diagram, the map automatically zoomed and centered on the location of "*inesc*". The Places visualization also highlighted the corresponding label. From here, the user can observe how that specific location positions itself in the other visualizations, allowing for quick comparison and pattern finding. These links exist between almost all possible combinations of visualizations.

The link between visualizations is possible thanks to an AngularJS tool: broadcasts. This broadcast mechanism tinkers with the scopes of the app. In the case of the

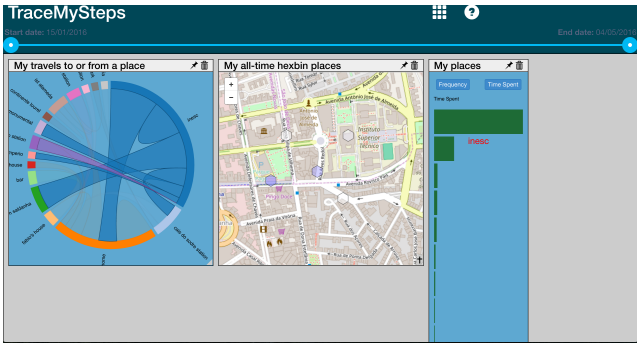

Figure 6. Three linked visualizations

broadcast, it is the root scope of the app. Every AngularJS app has one root scope. All other scopes are child scopes.

This mechanism is straightforward: when we call each directive (that contains each visualization), each one is created with its own scope. These scopes inherit the root scope of our application. Changes made on the root scope will be reflected in its children (the directives). Therefore, to pass the data between visualizations, we need to call the root scope in the directive. Using figure 8 again as an example, when a user hovers on the location in the Chord Diagram, we call the root scope and its broadcast method on the diagram. That method dispatches two parameters: a string identifying the specific directive broadcast and the data we wish to pass, wrapped as a JSON object. In that case we send the label of the place. Then, the two other visualizations contain a listener for the Chord Diagram, and when they get notified that the Chord Diagram sent information to them via the root scope, they do a series of visual operations to highlight the corresponding data in the different visualizations. When a widget is removed, it means the directive is also removed and its scope is deleted. We use a special listener for that event, in order to erase the broadcast listeners and avoid memory leaks.

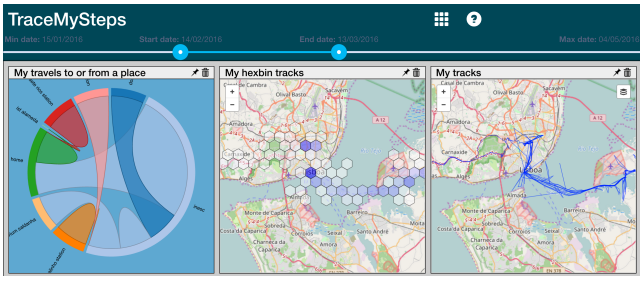

Figure 7. Observing data from the 14th of February to the 13th of March

The timeline slider can be considered one visualization, although of small proportions. When the user ends the slider dragging, the root scope broadcasts the minimum and maximum date of the time period, together with its

identifier. Then, the visualizations redraw. Figure 7 shows the timeline slider in action.

The first time visualizations are added to the grid, they start listening to the changes in the slider and react accordingly. But here, when a visualization is deleted we do not delete its timeline slider listener. This way each one saves state and updates its dataset in the background. A visualization is always consistent with the time period.

To test the scalability of our front-end, we also used the dataset from the load testing. The loading times for all visualizations are about the same, regardless the dataset size. The timeline slider works without any problem. The visualizations are still correctly linked. Regarding the drawing of the visualizations, eight out of nine adequately scale and show information without cluttering the widget. One visualization is slower in the display the tracks, but the speed is acceptable. The arc diagram is the one that does not scale well. It uses the correct information, but the space it uses is bigger than the provided. We consider the scalability of our front-end to be as good as our back-end.

EVALUATION

Here we evaluate our system, assessing if our solution reaches the objective we have set.

Experimental protocol

Twenty users were part of this evaluation. Each session was composed by four stages: initial questionnaire; demonstration of the system; a set of tasks presented to the user. They consist on actions to be performed on our solution, covering its main components; final questionnaire.

We used the same dataset for all users: five months of mobility tracking and semantic notes, collected by the author. The tasks are divided in four different groups, related to components of the solution: evaluating the menus and widgets, the connected visualizations, the timeline slider, and overall use of the solution. We decided to use the following metrics to collect data: task duration time, completion within a time interval, number of errors, number of clicks, correct answers and other annotations.

The task set consists of 16 different tasks, divided in the aforementioned four groups. Also, we asked users to express their doubts and concerns at the end of each task.

Results

Our system had an overall SUS score of 75.5. This means our system is good and we are on the right track. Some users had problems with the removal of the widgets and the closing of the menus. Users did not have any serious problems observing the data between the different linked visualizations. They all responded quickly to user actions. Regarding the timeline slider there were not any problems, since users found the timeline easy to use and the visualizations also responded quickly to the temporal filter. Overall, apart from some color scale problems, users found the interface to be working efficiently, with no excessive loading times of freezes. We found the results of our experiments positive and people found our system to be quick and well designed. Tasks were also adequate, despite a few problems. User tests shown us that we have a good quality system, with users being able to customize, filter and find information and patterns, regarding personal semantics and geolocation data.

CONCLUSION AND FUTURE WORK

Most approaches to the exploration of mobility tracking still do not cope with the personal aspect of data. So we found the need to design and create an application for users to understand and analyze their spatio-temporal information, with focus on the personal aspects of their data. We analyzed several different works that provided different ways for users to analyze mobility data. Few works included approaches to the personal aspect of the data, confirming the lack of solutions for this issue. Then, we made a survey with potential users, in order to understand the requirements. Then we established the architecture of our solution. Our back-end comprises a database and a web server. We also approach data collection, its problems and processing. For the front-end, we identified the three components we needed. We also study the scalability, by load testing a large dataset. The results were very satisfactory and so we can consider our solution to be quite scalable. Our objective of *designing a visualization that allows users to understand and analyze their spatio-temporal information, with focus on personal semantics* was attained, as our evaluation describes. Still, there are some issues that can be addressed in the future, such as improving the scalability of the Trips Network (arc-diagram) visualization, perfect the widget removal and menu closing buttons, add a login system, perfect the back-end processing (more optimization for GPS tracks) and the addition of a visualization based on (Otten 2015).

ACKNOWLEDGMENTS

This work would not have been achievable without the support of my family, professors and friends. To my family, thank you all for giving me the grit and inspiration. I am principally indebted to my parents and brother, who supported me emotionally and financially. Also, I would like to take this opportunity to express my gratitude and admiration towards my supervisor, Daniel Jorge Viegas Gonçalves for his commendable teaching, guidance and incentive throughout the progress of this thesis. To all my friends, and especially my closest ones, thank you for the advice, help, support, insights and understanding.

REFERENCES

1. Andrienko, Gennady, et al. 2013. Extracting Semantics of Individual Places from Movement Data by Analyzing Temporal Patterns of Visits. In *Proceedings of The First ACM SIGSPATIAL International Workshop on Computational Models of Place* (COMP '13). ACM, New York, NY, USA, , Pages 9 , 8 pages. DOI=http://dx.doi.org/10.1145/2534848.2534851
2. Andrienko, Gennady, et al. 2011. Challenging problems of geospatial visual analytics. In *Journal of Visual Languages & Computing. 22.4.* p. 251-256. https://www.researchgate.net/profile/Gennady_Andrienko/publication/220578976_Challenging_problems_of_geospatial_visual_analytics/links/546c6d7c0cf257ec78ffeb82.pdf

3. Andrienko, Gennady, et al. 2011. An event-based conceptual model for context-aware movement analysis. Int. J. Geogr. Inf. Sci. 25, 9 (September 2011), 1347-1370. DOI=http://dx.doi.org/10.1080/13658816.2011.556120
4. Andrienko, Natalia; Andrienko, Gennady. 2013. Visual analytics of movement: an overview of methods, tools and procedures. In *Information Visualization 12*, 1 (January 2013), 3-24. DOI=http://dx.doi.org/10.1177/1473871612457601
5. Attfield, Simon, et al. 2014. Patterns of life visual analytics suite for analyzing GPS movement patterns. In *Visual Analytics Science and Technology (VAST), 2014 IEEE Conference on. IEEE*. p. 343-344. DOI=http://dx.doi.org/10.1109/VAST.2014.7042557
6. Chen, Siming, et al. 2014. MovementFinder: A Multi-filter Visual Analytics Design for Movement Data Investigation. In *Visual Analytics Science and Technology (VAST), 2014 IEEE Conference on. IEEE*. p. 345-346. http://vis.pku.edu.cn/people/simingchen/docs/vastchallenge14-mc2.pdf
7. Guo, Chen, et al. 2014. Dodeca-rings map: Interactively finding patterns and events in large geo-temporal data. In *Visual Analytics Science and Technology (VAST), 2014 IEEE Conference on. IEEE*. p. 353-354. DOI=http://dx.doi.org/10.1109/VAST.2014.7042562
8. Hundt, Michael, et al. 2014. Visual analytics for detecting behaviour patterns in geo-temporal data. In *Visual Analytics Science and Technology (VAST), 2014 IEEE Conference on. IEEE*. p. 355-356. DOI=http://dx.doi.org/10.1109/VAST.2014.7042563
9. Jeon, Jae Ho, et al. 2014. Exploratory visualization of smarphone-based life-logging data using Smart Reality Testbed. In *Big Data and Smart Computing (BIGCOMP)*. 2014 International Conference on. IEEE, p. 29-33. DOI=10.1109/BIGCOMP.2014.6741400
10. Kraak, Menno-Jan. 2003. The space-time cube revisited from a geovisualization perspective. In *Proc. 21st International Cartographic Conference*. p. 1988-1996. http://www.itc.eu/library/Papers_2003/art_proc/kraak.pdf
11. Krueger, Robert, et al. 2013. TrajectoryLenses - a set-based filtering and exploration technique for long-term trajectory data. In *Proceedings of the 15th Eurographics Conference on Visualization (EuroVis '13)*. The Eurographs Association & John Wiley & Sons, Ltd., Chichester, UK, 451-460. DOI=http://dx.doi.org/10.1111/cgf.12132
12. Krueger, Robert, et al. 2014. Visual analysis of movement behavior using web data for context enrichment. In *Pacific Visualization Symposium (PacificVis), 2014 IEEE*. p. 193-200. DOI=http://dx.doi.org/10.1109/PacificVis.2014.57
13. Larsen, Jakob Eg, et al. 2013. QS Spiral: Visualizing periodic quantified self data. In *CHI 2013 Workshop on Personal Informatics in the Wild: Hacking Habits for Health & Happiness*. http://orbit.dtu.dk/files/73859557/chi2013_pi.pdf
14. Liu, He, et al. 2011. Visual analysis of route diversity. In *Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on. IEEE*. p. 171-180. DOI=http://dx.doi.org/10.1109/VAST.2011.6102455
15. Lu, Min, et al. 2015. OD-Wheel: Visual Design to Explore OD Patterns of a Central Region. In *2015 IEEE Pacific Visualization Symposium (PacificVis)*, Hangzhou, China. http://vis.pku.edu.cn/research/publication/PacificVis15_ODWheel.pdf
16. Lu, Min, et al. 2015. Trajrank: Exploring travel behaviour on a route by trajectory ranking. In *Visualization Symposium (PacificVis), 2015 IEEE Pacific. IEEE*. p. 311-318. DOI=http://dx.doi.org/10.1109/PACIFICVIS.2015.7156392
17. Lundblad, Patrik; Jern, Mikael. 2013. Geovisual Analytics and Storytelling Using HTML5. In *Information Visualisation (IV), 2013 17th International Conference. IEEE*. p. 263-271. DOI=http://dx.doi.org/10.1109/IV.2013.35
18. Otten, Heike, et al. 2015 Are there networks in maps? An experimental visualization of personal movement data. In *IEEE VIS 2015*. http://mariandoerk.de/papers/personalvis2015.pdf
19. Saraf, Parang, et al. 2014. Safeguarding Abila: Spatio-temporal activity modeling VAST 2014 Mini Challenge 2 award: Honorable mention for effective presentation. In *Visual Analytics Science and Technology (VAST), 2014 IEEE Conference on. IEEE*. p. 359-360. DOI=http://dx.doi.org/10.1109/VAST.2014.7042565
20. Thudt, Alice, et al. 2013. Visits: A spatiotemporal visualization of location histories. In *Proceedings of the eurographics conference on visualization*. p. 79-83. http://innovis.cpsc.ucalgary.ca/innovis/uploads/Publications/Publications/visits.pdf
21. Van Der Spek, S. C. Activity patterns in public space; a tool for assessing city centres. 2010. In *Walk 21, 11th conference, The Hague.* Nov. 16-19, 2010. https://www.researchgate.net/profile/Stefan_Van_der_SPEK/publication/254822298_Activity_patterns_in_public_space_a_tool_for_assessing_city_centres/links/0deec525f96d73d4e5000000.pdf
22. Wang, Zuchao; Yuan, Xiaoru. 2014. Urban trajectory timeline visualization. In *Big Data and Smart Computing (BIGCOMP), 2014 International Conference on. IEEE*. p. 13-18. DOI=http://dx.doi.org/10.1109/BIGCOMP.2014.6741397
23. Wood, Jo. 2014. Visual analytics of GPS tracks: From location to place to behaviour. In *Visual Analytics Science and Technology (VAST), 2014 IEEE Conference on. IEEE*. p. 367-368. DOI=http://dx.doi.org/10.1109/VAST.2014.7042569