POLITECNICO di TORINO

Dipartimento di Elettronica e Telecomunicazioni

ICT4TS LABORATORY DESCRIPTION

Marco Mellia

E-mail: mellia@polito.it

Table of content

Lab goals

The laboratory part of the ICT4TS focuses on the analysis of free floating car sharing data collected from open systems in the Internet. Data has been collected from websites of FFCS systems like car2go or Enjoy in Italy, and made available through a MondoDB database. The goal of the laboratory is twofold:

- to allow students to get used to ICT technologies typically used in the backend of smart society applications – database and remote server access, and writing analytics using simple scripts
- to allow students to work on real data, trying to extract useful information specific to the application domain – transport system in this case – and get used data pre-processing and filtering.

Accessing the data

Data is stored in a MongoDB server running on the server bidgatadb.polito.it:27017. It offers read only access to clients connected a network to the following database:

You can use command line interfaces (e.g., the mongo shell) or GUIs (e.g., the Robo3T application) if properly configured. For the mongo shell, you can use

```
mongo --host bigdatadb.polito.it --port 27017 --ssl \
      --sslAllowInvalidCertificates -u ictts -p \
      --authenticationDatabase carsharing
```

Collections

The system exposes 4 collections for **Car2Go**, which are updated in real time. Those are

- "ActiveBookings": Contains cars that are currently booked and not available
- "ActiveParkings": Contains cars that are currently parked and available
- “PermanentBookings": Contains all booking periods recorded so far
- "PermanentParkings": Contains all parking periods recorded so far

The same collections are available for **Enjoy** as well. Names are self-explanatory:

- "enjoy_ActiveBookings": Contains cars that are currently booked and not available
- "enjoy_ActiveParkings": Contains cars that are currently parked and available
- ”enjoy_PermanentBookings": Contains all booking periods recorded so far
- "enjoy_PermanentParkings": Contains all parking periods recorded so far

For Torino and Milano, the system augments the booking information with additional information obtained from Google Map service: walking, traveling, and public transportation alternative possibilities. Not all of them are available, due to the limited number of queries google allows.

Lab description

Students work in group of 3 colleagues. Each group is assigned three cities to analyse, as found on the google drive document. Each group has to work on the project assignment, and submit a report of max 5 pages which describes the finding. Code, scripts, etc., must be added in an appendix.

Step 1 – Lab #1 - Preliminary data analysis

To get used to both MongoDB and the data at disposal, investigate first the collections and get used to the document and field stored in each.

- How many documents are present in each collection?
- Why the number of documents in PermanentParkings and PermanentBooking is similar?
- For which cities the system is collecting data?
- When the collection started? When the collection ended?
- What about the timezone of the timestamps?

Considering each city of your group, check

- How many cars are available in each city?
- How many bookings have been recorded on the November 2017 in each city?
- How many bookings have also the alternative transportation modes recorded in each city?

For each question, write the MongoDB query, and the answer you get. Add a brief comment, if useful, to justify the result that you get.

Step 2 – Lab # 1 - Analysis of the data

Consider each city of your group, and the period of time of November 2017.
Consider the time series (city, timestamp, duration, locations). Process it to further analyse it by producing the following plots and results:

1. Derive the Cumulative Distribution Function of booking/parking duration, and plot them. Which consideration can you derive from the results?
    a. Which of the CDF is longer? Are there some outliers?
    b. Does the CDF change per each city? Why?
    c. Does the CDF change over time (e.g., aggregate per each week of data, or per each day or the week)? Why?
2. Consider the system utilization over time: aggregate rentals per hour of the day, and then plot the number of booked/parked cars (or percentage of booked/parked cars) per hour versus time of day.
3. Derive a criterion to filter possible outliers (booking periods that are too short/too long), so to obtain rentals from bookings, filtering system issues or problems with the data collection.
4. Filtering data as above, consider the system utilization over time again. Are you able to filter outliers?
5. Filtering the data as above, compute the average, median, standard deviation, and percentiles of the booking/parking duration over time (e.g., per each day of the collection).
    a. Does it change over time?
    b. Is it possible to spot any periodicity (e.g., weekends vs week days, holidays versus working periods)?

6. Consider one city of your collection and check the position of the cars when parked, and compute the density of cars during different hours of the day.
    a. Plot the parking position of cars in different times using google map. You can use the Google Fusion Tables to get the plot in minutes -- check https://support.google.com/fusiontables/answer/2527132?hl=en.
    b. Divide the area using a simple squared grid of approximatively 500mx500m and compute the density of cars in each area, and plot the results using a heatmap (i.e., assigning a different colour to each square to represent the densities of cars).
    c. Compute then the O-D matrix, i.e., the number of rentals starting in area *O* and ending in area *D*. Try to visualize the results in a meaningful way.
7. [Optional] For the city of Torino or Milano, try to correlate the probability of a rental with the availability of other transport means.
    a. Extract those valid rentals for which there is also the data for alternative transport systems.
    b. Consider one alternative transport system, e.g., public transports. Take the duration, and divide it into time bins, e.g., [0,5)min, [5,10)min, [10,15)min, …
    Compute then the number of rentals for each bin, i.e., the probability of seeing a rental given the duration of public transport would be in a given interval. Plot the obtained histogram, and comment the results.

Step 3 – Lab #2 - Prediction using ARIMA models

Consider the time series of rentals that you obtained in the previous steps, for each city. Consider a time period of 30 days for which you have data (the November 2017 is not a good dataset has you have found out).

The goal of this lab is to experiment with predictions using ARIMA models, and in particular to check how the error changes with respect to hyper-parameters. For this, you have to consider the various parameters in the ARIMA model training, including both the model parameters (p,d,q), and the training process, i.e., the training windows size N (how many past samples are used for training), and the training policy, i.e., expanding versus sliding windows. A possible outline of the work could be

1. For each city, consider the selected time period of 30 days, extracting the number of rentals recorded at each hour – after proper filtering of outliers.
2. Check that there are no missing samples (recall – ARIMA models assume a regular time series, with no missing data). In case of missing samples – define a policy for fitting missing data. For instance, use i) the last value, ii) the average value, iii) replace with zero, iv) replace with average value for the given time bin, etc.
3. Check if the time series is stationary or not. Decide accordingly whether to use differencing or not (d=0 or not).
4. Compute the ACF and PACF to observe how they decrease. This is instrumental to guess possible good values of the p and q parameters.
5. Decide the number of past samples N to use for training, and how many for testing. Given you have 30 days of data (each with 24 hours) you can consider for instance training during the first week of data, and test the prediction in the second week of data.
6. Given N, (p,d,q), train a model, compute the error. Consider the MPE and/or MSE – so that you can compare results for different cities (absolute errors would obviously be not directly comparable).
7. Now check the impact of parameters:
    a. Keep N fixed, and do a grid search varying (p,d,q) and observe how the error changes. Choose the best (p,d,q) parameter tuple.
    b. Given the best parameter configuration, change N and the learning strategy (expanding versus sliding window). Keep always the testing on the same amount of data. For instance, if you use 1 week for testing, you can use from 1 day to 3 weeks for training.
    c. Compare results for the different cities. How the relative error changes w.r.t. the absolute number of rentals?
8. [optional] Try to see how the time horizon h of the prediction impact the performance. Instead of predicting the number of rentals at t+1, use the model to predict the future rentals at t+h, h in [1:24].