Real-Time Highways Traffic Monitoring System, a Lightweight Model

Andrea Galloni, Balázs Horváth, and Tomáš Horváth

Faculty of Informatics
Eötvös Loránd University
Budapest, Hungary
{alfred.hofmann,ursula.barth,ingrid.haas,frank.holzwarth,
anna.kramer,leonie.kunz,christine.reiss,nicole.sator,
erika.siebert-cole,peter.strasser,lncs}@springer.com
http://www.springer.com/lncs

Abstract. The abstract should summarize the contents of the paper and should contain at least 70 and at most 150 words. It should be written using the *abstract* environment.

Key words: We would like to encourage you to list your keywords within the abstract section

1 Introduction

In this paper we describe the implementation of the developed framework for analyzing real-time status of the mobility traffic on Hungarian highways through the exploitation of mobile telecommunications logs of one of the major mobile telecommunication providers operating in Hungary.

The goal of the research is to develop a lightweight model which works efficiently and in real-time on small computing infrastructures highlighting the fact that not always such big data infrastructures or frameworks are needed given some specific problem statements or circumstances. In fact we managed to develop such a framework capable to process the daily data logs generated from mobile telecommunication events in about 10 minutes on a single process application covering all the highways within the Hungarian country. Experiments done on real but anonymized data shows promising results based on which it is worth further development of the proposed framework.

The rest of this paper is organized as follows: In section TODO REF we illustrate the related work and the state of the art of the field, in section TODO REF the dataset, the data pre-processing procedures and preparation steps are described, section TODO REF illustrates the user interface of the system, while in section TODO REF the overall architecture of the system is introduced, follows section TODO REF where we introduce the core statistical model while finally in section TODO REF experimental results are illustrated and discussed.

2 Related Work

TBD

3 The Data-Set, Preparation and Pre-processing

As is usual in machine learning, data mining and data analytics applications, the available data need to be pre-processed and prepared for further processing. In this chapter, the main steps on the server as well on the client side concerning data pre-processing and preparation are introduced.

3.1 Information from Other Available Sources

Before the the research could focus on the data provided by Magyar Telekom for the purposes of this project, the team had to re-consider the necessary information and the format of the data. First of all, Hungary "as geodata" was needed for visualization and to set borders to the algorithm later on. For that purpose, there are several solutions, the most suitable is to use the Open street map, to extract information and data from it. OpenStreetMap[1] (OSM) is an open source tool built by the mapper community. It provides data about roads, trails, cafes, railway stations and other basic map features from all around the world.

Fig. 1. OpenStreetMap

With this tool detecting the objects needed for the visualization is much easier. After those objects are detected, the next step is to create polygons out of

[1] `https://www.openstreetmap.org/`

them. For this task OSM provides a tool[2] called "polygons". This tool can generate the whole geometry of the given OSM relation id, with the corresponding sub-relations. When the geometry is available, it is possible to generate simplified geometries from this one, and export them as .poly, GeoJSON, WKT or image formats. The result, in this case, will be a GEOJSON file which is containing the borders and the area of Hungary. To validate the GEOJSON there are numerous solutions. The result of validation can be seen on Figure 2.

Fig. 2. geojson.io validation

After the validation of the country was accepted, the next step is to detect and model the highways. For this purpose multiple applications were used. First step is to gather the data about the highways. The website Overpass Turbo[3] provides a tool which is able to run queries on map data, and export them.

```
(
  node["highway"="motorway"]({{bbox}});
  way["highway"="motorway"]({{bbox}});
  relation["highway"="motorway"]({{bbox}});
);

out body;
>;
```

[2] http://polygons.openstreetmap.fr/
[3] https://overpass-turbo.eu

```
out skel qt;
```

[Sample query configuration]

On the code snippet **??** above, the configuration of the query to get the highways be seen. The result of the query is geolocation data in a square shaped area. This data even contains additional information about each road segment, like speed limits. An example of the output can be seen on the Figure 3. The application allows to download this information in GEOJSON format. Inside the GEOJSON each road segment is a "Feature" in a "FeatureCollection". Each Feature has the same attributes as on the following code snippet **??**:


```
{
      "type": "Feature",
      "properties": {
        "@id": "way/275732",
        "highway": "motorway",
        "lanes": "2",
        "lit": "yes",
        "maxspeed": "130",
        "name": "Westautobahn",
        "oneway": "yes",
        "ref": "A1",
        "source:maxspeed": "AT:motorway",
        "toll": "yes"
      },
      "geometry": {
        "type": "LineString",
        "coordinates": [
          [
            16.2327096,
            48.2030833
          ],
          [
            16.2323309,
            48.2033167
          ],
          [
            16.2313232,
            48.2039551
          ],
          [
            16.2308245,
            48.2042339
          ]
        ]
```

```
    },
    "id": "way/275732"
}
```


[A sample Feature in FeatureCollection]

In order to focus on the Hungarian highways, steps of preprocessing had to be done. As it is visible and mentioned before, the exported data is the form square shaped areas, so the result of it contains highways from other countries, too. This phase of preprocessing was manual work, the needless features had to be removed.

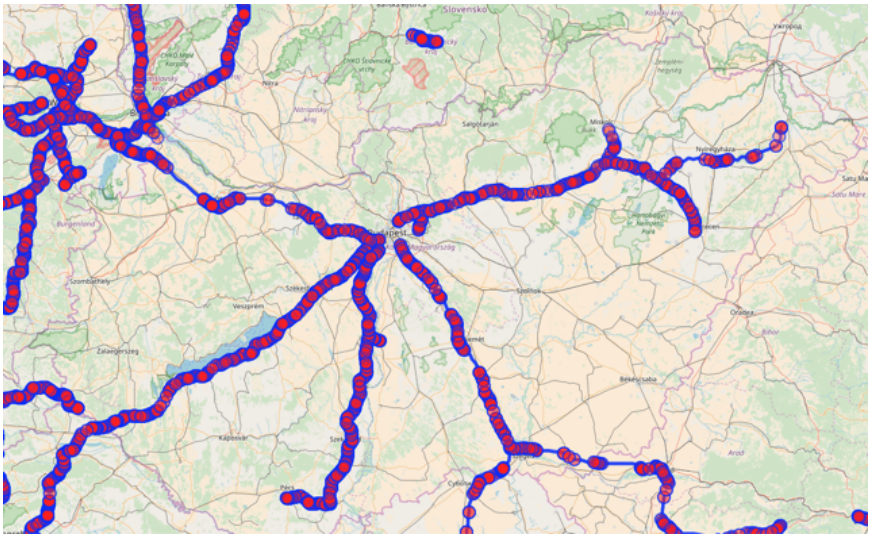

Fig. 3. Highway data visualized with overpass

After the highways have been filtered, the next step was to merge both the borders of the country, the area of the country and the highways into one GEOJSON, which makes the access of them easier for the visualization and the model as well. The validated visualization of the ready GEOJSON file is visible on the Figure 4, which was made by *geosjon.io*.

3.2 Detection of Interesting Towers

Since (as previously stated) the aim of this project is to infer the traffic status of the Hungarian highways infrastructure through the analysis and correlation of the status of the mobile telecommunication infrastructure, first, before developing the application server, in order to create a joint model we needed to perform some data filtering and preprocessing tasks.

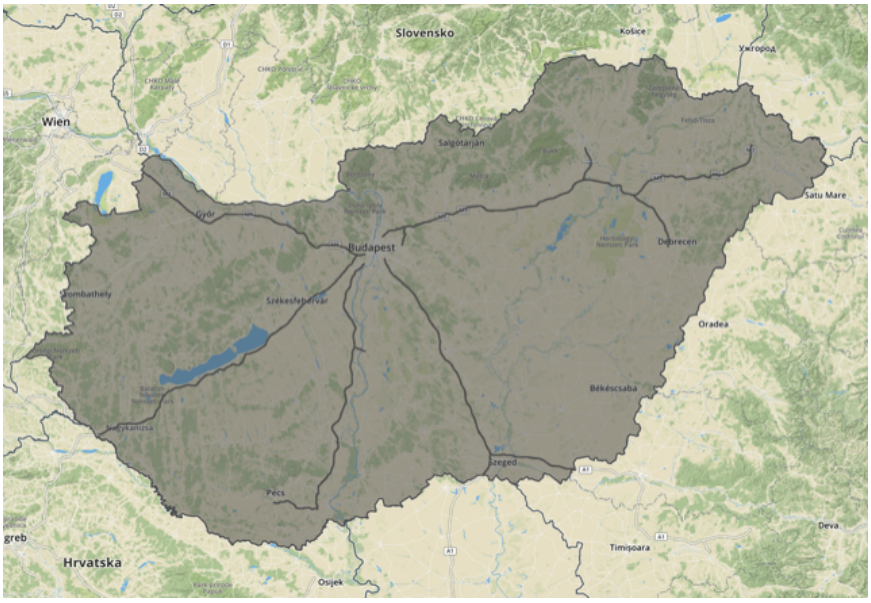

Fig. 4. The merged data visualized with geojson.io

About Tower Cells: Data-Set Findings and Core Concepts In order to monitor the highways infrastructure traffic and exclude irrelevant information from the data model we needed to filter and select just the relevant cell towers that have a strict correlation with the highways infrastructure, thus excluding all the remaining ones. Nevertheless, before to proceed with this task some observations and assumptions on the data-set have been made.

Firstly we plotted all the cell towers (as suggested from the provided dataset the number of these cell towers displaced in the whole Hungarian territory is 79796). In order to do so, in this preliminary stage, for matters of time and efforts, we did not develop any in-house tool, instead, we just used an online free tool[4] for plotting points in a map. The aforementioned tool, given a set of geospatial coordinates allows to plot a set of points on a map.

From Figures 5 and 6 representing the result of the plotting process, it is easy to observe and get a general idea of the high number of cell towers displaced in the whole country (Figure 5) and in detail (Figure 6) the city of Budapest and its east country side area. Looking at both images, especially at Figure 5 is clear that, provided the mostly flat characteristics of the Hungarian landscapes, thus assuming that the cell towers displacement somehow is not conditioned by any of the topographic properties of the surrounding areas. At a glance, the density of the cells follows exactly the density of the population over the whole territory. This characteristic of the cell towers placement is due to several factors

[4] `http://www.hamstermap.com`

such as scalability, laws of physics and signal processing theory. In fact the cell towers placement has not a constant density, instead it looks to be more dense within urban areas rather than in the countryside. All these observations can be confirmed having a look at Figures 7 and 8. This characteristic increases the overall localization resolution in urban areas rather than in the rest of the territory. Given this latter statement, the density of the cell tower placement and its spacial characteristics represent a crucial issue in terms of space-resolution within the monitoring system we developed.

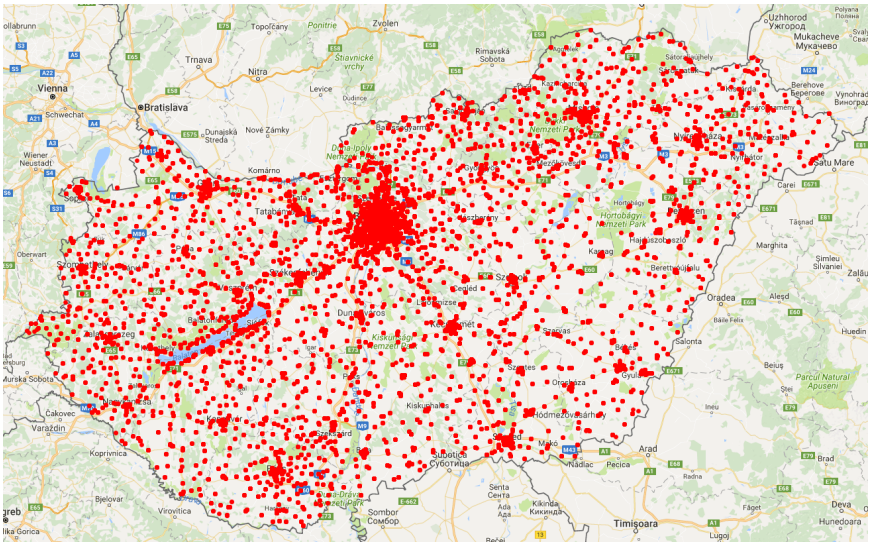

Fig. 5. Cell Towers Placement in Hungary

In order to understand the main concepts on which the whole project has its roots it is of extremely importance to note that in non urban environments cell towers appear to be placed according to the mobility infrastructure, highways included. This latter property is of a fundamental importance in the dataset, in fact it makes easier to monitor the highway mobility activity from the telecommunication perspective, making feasible the process of selecting the relevant towers that could generate relevant information logs, furthermore the activity of the cells displaced far from cities and close to highways can provide data about travelers with a significantly low level of what we define static noise (telecommunication activities performed by non traveling subscribers) increasing the correlation between the two systems thus creating a good ground for the model we are going to describe in the following part of this document.

When the number of non moving subscribers is low, in general, the cell log activity result to be generally lower than the city's one. In addiction, it is also well correlated with the highways daily rhythm and its real-time status making

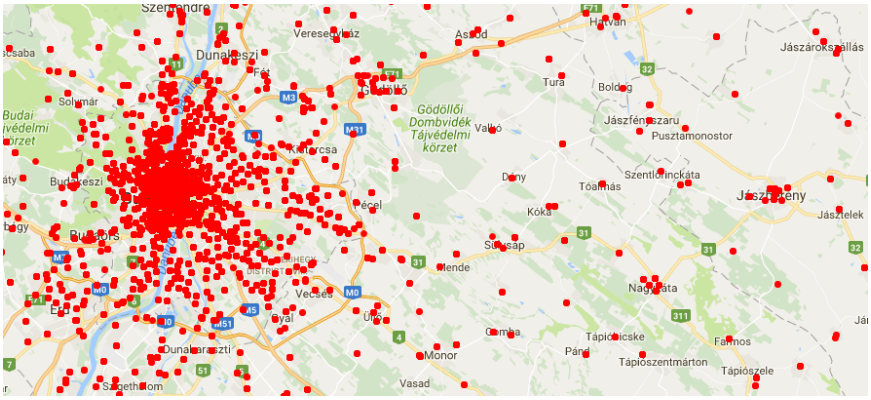

Fig. 6. Budapest Urban Area and Rural Area Comparison

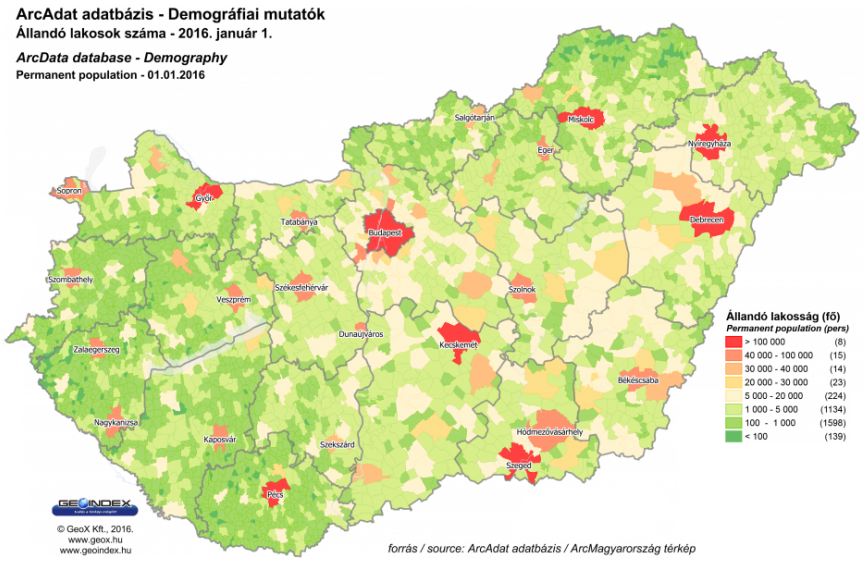

Fig. 7. Permanent population in Hungarian settlements 2016, *http://www.geoindex.hu/*

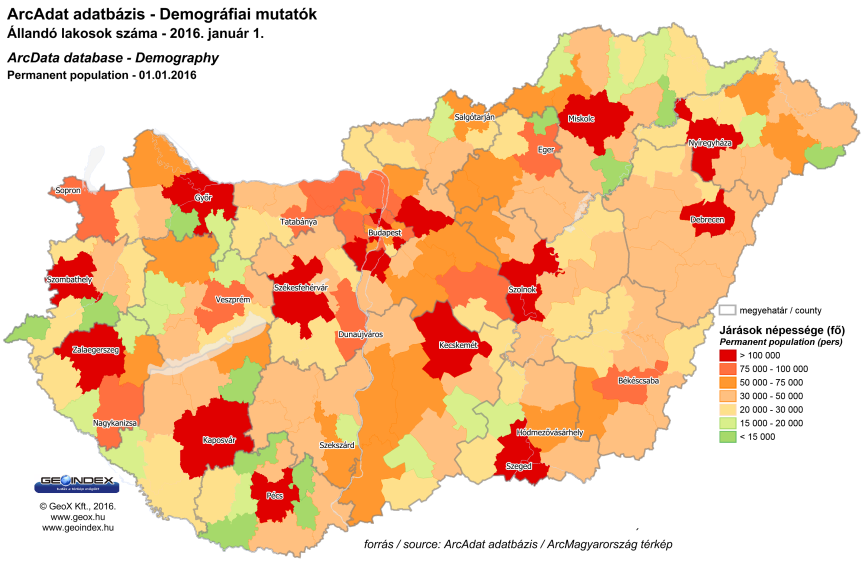

Fig. 8. Permanent population in Hungary 2016, *http://www.geoindex.hu/*

relatively easy to infer the status of a system (the mobility one) from the other (the telecommunication one).

Before describing the process of matching both the infrastructures and selecting the relevant towers for our project, it is necessary to describe the main characteristics of the data we made use of.

As described in the previous section through the use of OSM data we selected and isolated all the map features concerning all the Hungarian borders and highways by mean of the .geojson file format (the standard format used by OpenStreetMap to formally describe all the objects on maps). The remaining part of the data related to the communication infrastructure is contained within the file *cells.csv* namely a file containing all the Cell Towers placed within the Hungarian country and their specification.

hungarianhighways.geojson Here, in the Code Snipped **??** we can observe the structure of a sample of *hungarianhighways.geojson* file. As we can observe this file represents a json object containing a collection of features. An important characteristic of this file we utilized to match the two data-sets is that every feature contains an unique id (namely: *@id*) along with a *geometry object* containing the coordinate points (longitude and latitude) of every slice of the highway represented by the unique id. In simple words every slice of an highway is composed by a set of coordinates points graphically represented by several little segments passing through all of them, important to remark is that all this little segments share the same unique id.

```
[caption=hungarianhighways.geojson Sample, label=code:geojsonsample]
{
  "type": "FeatureCollection",
  "generator": "overpass-ide",
  "copyright": "Data [...] from www.openstreetmap.org. [...]",
  "timestamp": "2017-08-08T12:38:02Z",
  "features": [
    {
      "type": "Feature",
      "properties": {
        "@id": "way/4315827",
        "highway": "motorway",
        "int_ref": "E 65;E 75",
        "lanes": "2",
        "maxspeed": "130",
        "oneway": "yes",
        "ref": "D2",
        "toll": "yes",
        "toll:hgv": "yes"
      },
      "geometry": {
        "type": "LineString",
        "coordinates": [
          [
            17.1723507,
            48.0177768
          ],
          ...
          [
            17.1016405,
            48.074648
          ]
        ]
      },
      "id": "way/4315827"
    },
    ...
    ]
}
```

[hungarianhighways.geojson Sample]

cells.csv Table **??** describes a sample content of the file *cells.csv* This file contains details about the cell towers displaced within Hungary. As shown, the file is composed by four columns.

Follows the description of the file content:

- **Gen (Generation):** This column indicates which technology is powering the cell tower signal. This record contains the number of mobile telecommunications technology the domain of this column can be respectively: 2G, 3G or 4G.
- **MSC ID:** This value represent the id of the cells contained on the MSC files logs. In fact for every interaction between a mobile phone and the mobile switching center the identifier string of the voice call cell where the phone is connected is logged and stored. This value is unique for every record and it resulted to be useful to detect the geo-localization of the call event.
- **NGPRS ID:** This value represent the id of the cells contained on the NGPRS files logs. In fact for every interaction between a mobile phone and the gprs data network the identifier string of the cell where the phone is connected is collected and stored. This value is unique for every record and it resulted to be useful to detect the geo-localization of the data event.
- **Lat (Latitude):** Latitude is a geographic coordinate that specifies the northsouth position of a point on the Earth's surface. Latitude is an angle which ranges from 0° at the Equator to 90° (North or South) at the poles. In this case the information is represented on Decimal degrees (DD). This value represents the exalt latitude of the specific cell tower.
- **Long (Longitude):** Longitude is a geographic coordinate that specifies the east-west position of a point on the Earth's surface. By convention, the Prime Meridian, which passes through the Royal Observatory, Greenwich, England, was allocated the position of zero degrees longitude. Also in this case the information is represented on Decimal Degrees (DD). This value represents the exalt longitude of the specific cell tower.

Matching the Datasets The process of merging both the relevant parts of the two datasets is aimed to get a correspondence between a set of highway slices and the cell tower responsible to highlight the behavior of that part of highway.

Since to a specific coordinate point may correspond to several towers (generally two, one for the NGPRS data and one for the MSC data) we decided to generate another set of unique ids grouping the cells sharing the same exact coordinate points. This operation can be justified given the fact that we are not interested to preserve the nature of the event or the specific id of the receiver rather we are interested just on the location of the event and the unique id of the user who generated such event.

In order to do so, an ad-hoc Python module have been developed, namely *psyspacexy*. The functionality of this module is to build up a tree-based data structure (*QuadTree*) representing a two-dimensional space that let us to find the closest neighbor of a point in an efficient time, the complexity is namely $O(logN)$ per query plus $O(NlogN)$ for the tree building task.

The generated tree contains all the coordinates of the cell towers that are listed in the *cells.csv* file. Then, for every feature (thus for every unique @id in the .geojson) the closest cell has been found querying the tree structure. Finally

a .json file (competence.json) containing the matching is generated and stored (code snippet **??**).

After this process a specific id-translation delegate file is stored as a Python's *.pickle* file (for efficient loading and reading operations) containing an HashMap (code snippet **??**).

```
[language=python, caption=Data-Sets Matching, label=code:competence_generation]
#!/usr/bin/python3

from psyspacexy.QuadTree import QuadTree
from psyspacexy.point import Point

from collections import defaultdict

import json
import csv

def main():
    tower_set = load_towers_csv("res/data/hack_cells.txt")
    # Creates a list of tuples containing the coordinates of the cell
    myTuple = tuple((float(p.split(",")[0]),float(p.split(",")[1]))
        for p in tower_set)

    # creates an efficient data structure to find the closest point neighbour
    qTree = QuadTree(myTuple)
    # loads the gejson given the file path
    geoj = load_geoj("res/data/hungarianhighways.geojson")

    # creates a default dic of empty lists
    resDic = defaultdict(lambda:[])

    # for every feature contained in the .geojson gets the closest tower.
    for feature in geoj:
        resDic[tower_set[get_closest_tower(feature,qTree)]]
         .append(feature['properties']['@id'])

    # stores the coupled data in competence1.json file
    with open('res/data/competence1.json', 'w') as outf:
        json.dump(resDic, outf)

def get_closest_tower(feature,qTree):
    dDic = defaultdict(lambda:0)

    for i in feature['geometry']['coordinates']:
        closest_point = qTree.findNeighbourPoint(Point(i[1],i[0]))
```

```
        point_ID = "{:2.10f},{:2.10f}"
         .format(closest_point.x,closest_point.y)
        dDic[point_ID]+=1

    return max(dDic, key=dDic.get)


# loads the gejson given the file path
def load_geoj(path):
    # open the file in ReadOnly mode
    with open(path, 'r') as fileReader:
        data = json.load(fileReader)
    # returns jhust the features list of the .geojson file
    return data["features"]


# load the towers from a file passed as path returning a dictionary
def load_towers_csv(path):
    tower_dic = {} # create an empty py dictionary (HashMap)
    # open the file in ReadOnly mode
    with open(path, 'r') as fileReader:
        csvReader = csv.reader(fileReader, delimiter=';')
        i = 0
        for line in csvReader:
            if line[3] and line[4]:
                tower_dic[("{},{}".format(line[3],line[4]))] = i
                i += 1

    return tower_dic

if __name__ == '__main__':
    main()

# EOF
```

[Data-Sets Matching]

```
[language=python, caption=ID Translator generator, label=code:ids_generation]

#!/usr/bin/python3

import json
import pickle

with open('competence1.json','r') as forig:
    orig = json.load(forig)
```

```
translator = {}
new_ids_dict = {}
i = 1
for key,val in orig.items():
    translator[key] = i
    new_ids_dict[i] = val
    i+=1

with open('competence_int.json','w') as cint:
    json.dump(new_ids_dict,cint)

with open('translator.pkl', 'wb') as outp:
    pickle.dump(translator,outp)

#EOF
```

[Data-Sets Matching]

Conclusions In this section we managed to describe the data models at the root of our system implementation and their merging operations. We created a correspondence between the highways infrastructure and the cell towers placed along the territory. Furthermore this operation will let us to develop a lightweight system on top of it since we reduced a lot the number of cells that are of our interest. Furthermore taking into account just the relevant towers will lead us to faster discard these irrelevant logs.

Finally with the help of Figure 9 (just for illustrative proposes) we can get a visual idea of the merging process previously described, here, the red spots represent the cell towers whose logs have no direct correlation with the highway status, while the other colored dots represent the cell towers whose signal supposedly reaches the travelers within the highway (based on our assumptions). The colored parts of the highway represent the competence of each tower coupled with a piece of highway so the tower cells and slices of highways holding the same color represent the outcome of the data-merging process.

3.3 Available Data

Within this section we are going to provide all the details and some statistics about the specific data-set on which all the experiments have been performed.

Call Detail Records A Call Detail Record (CDR) is a data record produced by a telephone exchange or other telecommunications equipment that documents the details of a call or other telecommunications transaction (e.g., Short Message

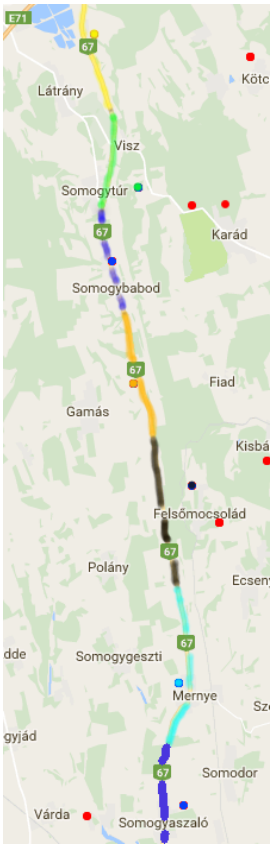

Fig. 9. Competence Sample

Service also known as SMS) that passes through a telecom provider infrastructures.

The records contain various attributes of the call, such as time, duration, completion status, source number, and destination number. Call detail records are useful for many operational tasks providing useful information that can help to identify suspects, call data logs can reveal details such as some individual's relationships with other individuals, communication and/or behavior patterns, and even location data that can establish the estimated location of an individual during the time spent on a mobile phone call. Of course, there are restrictions on the availability of some types of data and on the projected applications. First,communication's contents are not stored by the operator; second, while mobile phone operators have access to all the information filed by their customers and the CDRs, they may not give the same access to all the information to a third party (researchers included), depending on their own privacy policies and the laws that apply in the country of application.

The provided dataset is composed by several *csv* (comma separated values) source files; all the information contained on such files have been organized on a daily basis. The entire information contained within the dataset covers a range of 31 days, more precisely between 15th September 2016 and 15th October 2016, where all the unique ids referencing to the users have been anonymized on a daily basis.

The Data-Set Structure The available dataset is composed by several files accounting a size of 500GB. The data is mainly divided in four types of files:

- Cell Towers Global Positioning System (*cells.csv*), already described.
- CRM Customer Relationship Management Data (*crm_YYYYMMDD.csv*)
- MSC Mobile Switching Center (*msc_YYYYMMDD.csv*)
- NGPRS Network General Packet Radio Service (*ngprs_YYYYMMDD.csv*)

Data-Set General Information and Statistics For each category (CRM,MSC and NGPRS) the number of files provided is equal to 30 (one for each day).

CRM The CRM files contain all the details about users and some generic data about profiling information. Provided the daily anonymization of the data, the dataset does not contain just a file about subscribers but there is a file for every day covered by the dataset.
Follows the description of the file content:

- **IMSI:** The International Mobile Subscriber Identity or IMSI (daily anonymized) is used to identify the user of a mobile network and is a unique identifier associated with the cellular network. It is represented as a 64 bit infromation field and is sent by the phone to the network. It is also used for acquiring

other details of the mobile in the home location register (HLR) or as locally copied in the visitor location register. The IMSI identifier is usually composed by 15 digits, but can be shorter.
- **ZIP:** The ZIP code is a postal code, in hungary it is composed by a series of four digits, it is usually included in a postal address for the purpose of sorting mail. The assignment of this code is aimed to divide geographical areas in smaller sections. It gives an idea of the home site of the subscriber. The total number of zip codes contained in the dataset is 4205.
- **City:** The name of the city where the subscriber has its residence. The number of the unique records is exactly 5353. Provided that the number of unique ZIP codes is higher than the number of cities in this case the dataset might have some inconsistency or ZIP codes might be the same for several little villages.
- **Sex:** The sex column contains the gender of the subscriber, it can have two values *M* for male and *F* for female, in several records this field have been observed to be missing.
- **Age:** This value contains the age of the subscriber expressed in years. The median value of the age resulted to be 44.
- **Cat.:** This field contains a variable describing the client's contract type basically it helps to define if the subscriptions are for business proposes or not providing details on the size of the business. The possible values are:
  - *LAK* for private individuals
  - *SH+* for private individual entrepreneurs
  - *SMB* for small and medium business
  - *TSS* TSS for T-systems subscribers
- **SS:** This record is used to classify users if they have both a fixed line subscription and a mobile subscription with the same operator.
- **ARPU:** Average revenue per unit, usually abbreviated to ARPU, is a measure used primarily by consumer communications and networking companies, this value defines how the subscriber is used to spend on average. This column has a discrete co-domain, in fact the values are in a range between 1 and 5. In this case the user is classified based on its best fitting category 5 means that the user is very valuable for the company revenues.
- **4G:** This field simply indicates if the SIM card of the customer supports 4G or not.
- **SED:** Namely: Subscription End Date, this field indicates the year and the month when the customer's subscription will expire.

MSC The MSC (namely *mobile switching center*) files contain all the details about users interactions with the voice infrastructure such as voice call sent/received or SMS sent/received. This kind of information represents the core of our research. As it is coupled with the GPRS data, it gives to us a real perspective of the users activity with the network. This kind of events stored in the MSC files always contain time logs. The number of daily logs contained into each file is about 30 millions.
Follows the description of the file content:

- **EID:** This column represent the identifier of the event. This value is unique and is an integer.
- **SIMSI:** This value represents the International Mobile Subscriber Identity of the subject performing the action.
- **IMSI:** This value represents the International Mobile Subscriber Identity of the object of the action.
- **TAC:** The TAC namely Type Allocation Code is the initial eight-digit portion of the 15-digit IMEI code used to uniquely identify mobile devices. The Type Allocation Code identifies the specific model of the mobile telephone for use on a GSM, UMTS networks. The TAC number is interesting because can give insight on the actual mobile handsets market. Since the IMEI code is unique for every device it have been truncated for privacy reasons.
- **EID:** This field provides information about the kind of event that triggered the msc log. The possible values are:
  - *1* Outgoing call
  - *3* Incoming call
  - *7* Incoming SMS
  - *9* Outgoing SMS
- **Date T.:** This field in terms of our research is the most valuable one, in fact it provides the a temporal coordinate about the time when the event have been triggered.
- **CID:** This field represents the specific cell tower identifier, even in this case this record is fundamental for our research, in fact if merged with the information contained in the cells file provides an estimation of the geographic location of the described event.

NGPRS The NGPRS file (namely *Network General Packet Radio Service*) files contain all the details about users interactions with the data services such as data requests sent or received. This kind of information is the most valuable in our research as it is coupled with the MSC data and gives to us a real perspective of the users activity with the network. This kind of events stored in the NGPRS files always contain time logs. The number of daily logs contained into each file is about 200 millions.
Follows the description of the file content:

- **EID:** This column represent the identifier of the event. This value is unique and is an integer.
- **IMSI:** This value represents the International Mobile Subscriber Identity of the subject performing the action.
- **TAC:** As for the MSC file the TAC namely Type Allocation Code is the initial eight-digit portion of the 15-digit IMEI code used to uniquely identify mobile devices. The Type Allocation Code identifies the specific model of the mobile telephone for use on a GSM, UMTS networks. The TAC number is interesting because can give insight on the actual mobile handsets market. Since the IMEI code is unique for every device it have been truncated for privacy reasons.

- **Event T. (Event Type):** This field describes the type of the event, it can contains three values:
  - *SGSN* The Serving GPRS Support Node (SGSN) is a main component of the GPRS network, which handles all packet switched data within the network, e.g. the mobility management and authentication of the users. This event is logged every authentication process is triggered.
  - *SGW* The SGW routes and forwards user data packets, while also acting as the mobility anchor for the user plane during inter-Node handovers. This event is enabled every time the user equipment changes node.
  - *PGW* The PDN Gateway provides connectivity from the UE (User Equipment) to external packet data networks by being the point of exit and entry of traffic for the UE. This event is logged every time the user performs data requests or receives data packets.
- **Date T.:** This field in terms of our research is the most valuable one, in fact it provides the a temporal coordinate about the time when the event have been triggered.
- **Cell ID:** This field represents the specific cell tower identifier, even in this case this record is fundamental for our research, in fact if merged with the information contained in the cells file provides an estimation of the geographic location of the logged event.

CRM Statistics The files related to the CRM (Customer Relationship Management) are organized in a daily basis for at least two reasons: First because of the anonymization process and, second, because the number of customers changes daily. In fact subscription are subject to expiration, customers might resign from contracts and some customers could migrate from a mobile provider to another on a daily basis. The average number of subscriptions over time is 5.608.677 (increasing over time) while the average size per file is 233.93 MB.

MSC Statistics The average number of logs for MSC related files over time is 28.010.199 (increasing over time) while the average size per file is 2068.93 MB (around 2GB).

Follow some tables and graphs useful to get an idea of the size of the related data:

NGPRS Statistics The average number of logs for NGPRS related files over time is 184.377.183 (increasing over time) while the average size per file is 14214.68 MB (around 14GB).

Follow some tables and graphs useful to get an idea of the size of the related data:

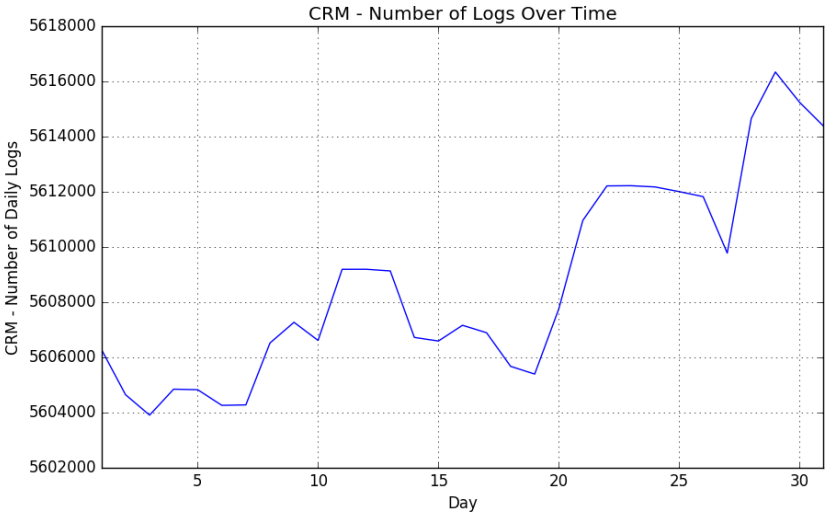

Fig. 10. CRM - Number of Daily Logs Over Time

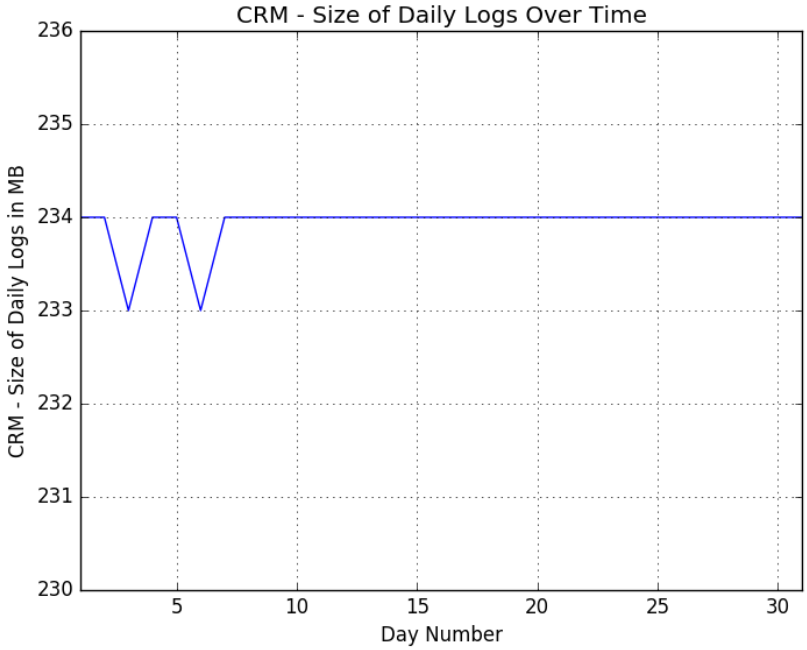

Fig. 11. CRM - Size of Daily Logs Over Time (MB)

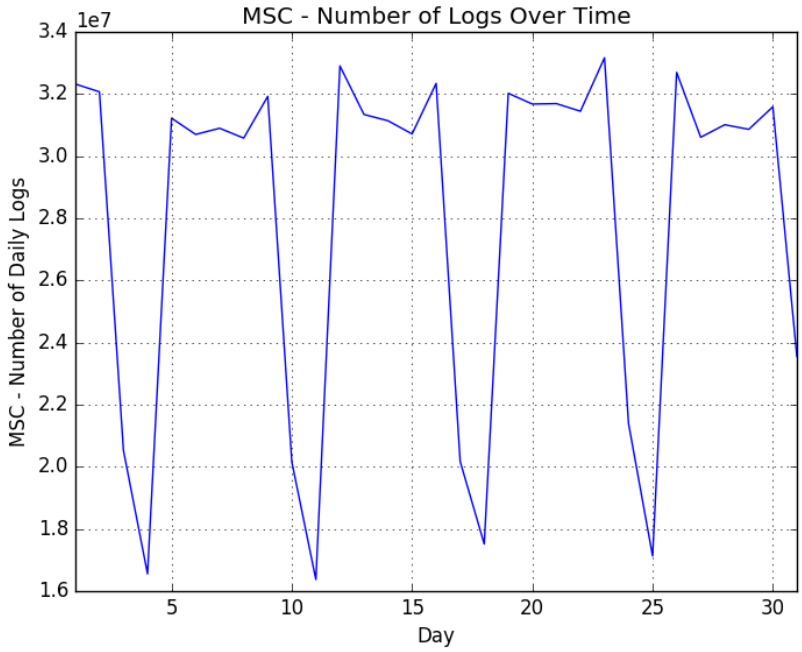

Fig. 12. MSC - Number of Daily Logs Over Time

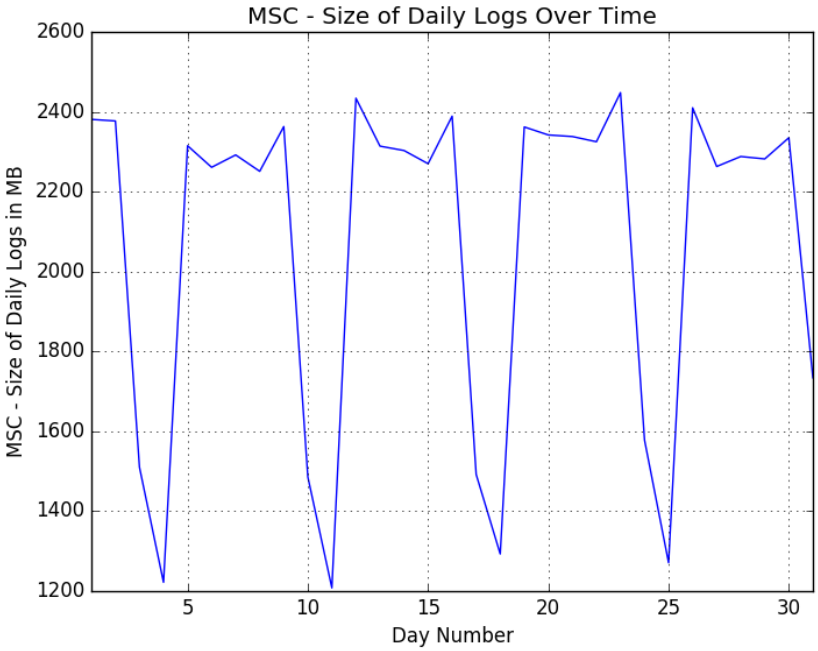

Fig. 13. MSC - Size of Daily Logs Over Time

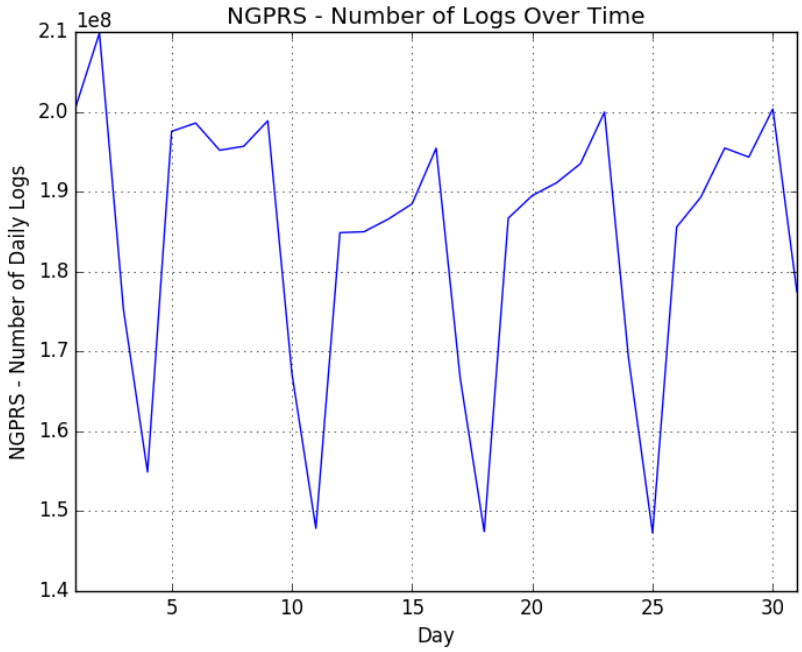

Fig. 14. NGPRS - Number of Daily Logs Over Time

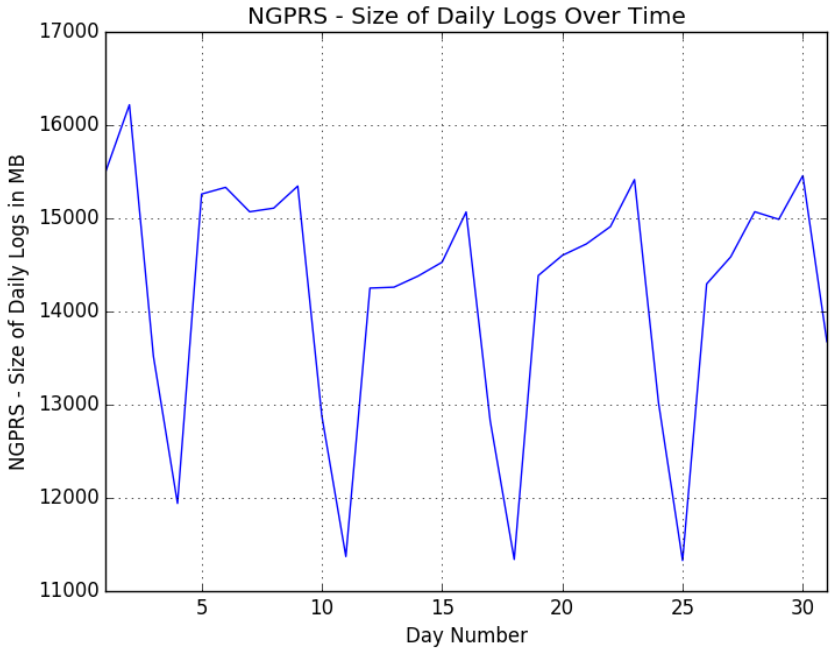

Fig. 15. NGPRS - Size of Daily Logs Over Time

Dataset General Comments In this section we described how the dataset is composed. The CRM contains anonymized information about customers. The localization data is contained on the cells file. Finally the communication events are stored in two separate files the MSC for voice data (and SMS) and NGPRS for the mobile broadband data connection traffic events. Interesting to note that for both MSC and NGPRS log files we can observe that the number of logs is compliant with the human rhythms, in fact the overall number of logs during the weekend (when business interaction are fewer if compared to the weekdays) is significantly lower than in the other days observing relative minimums on Sundays, this characteristic can be observed even in the graphs describing the size of files since the correlation between the two is strong (the more logs the more space needed to store them).

Given the the presented characteristics of the data, the location obtained by merging the information contained in the MSC and NGPRS logs and the cell towers location does not provide the exact location of the subscriber but simply the location of the cell tower from where the event was registered, this characteristic of the data-set has a fundamental impact on the precision (in terms of resolution) of the entire application system because the area covered by cell towers (in rural areas) can be about several square kilometers. The range covered by a specific cell tower depends on several factors such as: the average "crowdedness" of the specific area, the generation and the geological properties of the area.

3.4 Data-Set Inconsistencies

After getting general overview of the data while performing some deeper analysis on its content, we realized that our dataset contained some inconsistencies, in fact, some records (both for MSC and NGPRS files) contained unique cells identifiers that did not match with our list of cell towers identifiers or in some cases the logs contained a null subscriber identifier.

In order to better understand and get a quantitative measure of this inconsistencies and the number of affected records, we decided to write an ad hoc test. This analysis have been performed to understand if these inconsistencies would have an relevant impact in our research or not. During this process we iterated over the whole dataset files and tried to match the identifiers. In case this does not happen, we collect some information about it generating the following statistics. The test scanned the whole dataset involving billion of records (namely 6,584,008,885) the whole process took more than five hours (exactly 5h, 31m and 17s) to run in a single threaded Python process.

Table **??** shows the statistics reported by our test; the table reports the total number of processed files, the total number of records contained in our dataset, the total number of the cells found on MSC and NGPRS files but not found in the cells file, the number of cells listed in the cells file, the total number of all the cells found in the CRM and NGPRS files and finally the number of the so called *phanthom cells* that is the number of cells contained in the cells file but not referenced at all from both the MSC or NGPRS files.

The number of affected records is considerably high, more than one third of the data-set appear to be damaged. Anyway, we decided to go further in the research, in fact we realized the missing values are not randomly distributed but they involve precise locations (the information about one is fully present or is not present at all).

For illustration proposes we report some additional tests performed over four files, two from MSC and two from NGPRS:

```
==============================
FILE: cells.csv
==============================
Number of records: 79.796
Number of null IDs: 4.352
Number of valid records: 75.444
==============================
FILE: ngprs_.20160915
==============================
Number of records: 200.481.896
Number of records having a null user ID: 5.232.836
Number of records with missing towers IDs: 68.300
Number of cells not present in cells.csv: 320.983
Number of records with inconsistent cells IDs: 83.183.560
Number of distinct cells present in cells.csv: 29.743
==============================
FILE: ngprs_20160916
==============================
Number of records: 209.846.749
Number of records having a null user ID: 5.393.040
Number of records with missing towers IDs: 66.856
Number of cells not present in cells.csv: 314.688
Number of records with inconsistent cells IDs: 86.502.907
Number of distinct cells present in cells.csv: 29.772
==============================
FILE: msc_20160915
==============================
Number of records: 32.322.009
Number of records having a null user ID: 1.007.472
Number of records with missing towers IDs: 19
Number of cells not present in cells.csv: 2.521
Number of records with inconsistent cells IDs: 502.431
Number of distinct cells present in cells.csv: 27.312
==============================
FILE: msc_20160916
==============================
Number of records: 32.071.631
Number of records having a null user ID: 1.108.897
```

```
Number of records with missing towers IDs: 9
Number of cells not present in cells.csv: 2.521
Number of records with inconsistent cells IDs: 528.308
Number of distinct cells present in cells.csv: 27.361
```

3.5 Data Pre-processing

In order to reshape and clean the database, all the power and simplicity of the Linux *Burning Shell* have been used. In fact tools like *awk*, *sort*, and other text manipulation commands have been used. Since all the file have been provided as compressed *.tar* archives All the files have been decompressed using the GNU *tar* utility. During the preprocessing phase all the non-necessary information have been removed, only the strict necessary data have been kept such as event log subscriber id, time and cell position. All the customer data have been filtered out and reordered in fact since the order of logs appeared to be pretty casual all the logs have been sorted in a chronological order maintaining the daily basis of the files. Provided these premises our observation perspective is not centered on the single user activity but rather we finally aggregate the user's activity around the cells.

The temporal resolution of the logs is in terms of seconds, but, as we will see further in our research, a more aggregated time resolution provided a better understanding of the data both during the visualization process and the experimental one.

The simplicity and the "programmer-friendly characteristics" of Python made it the best choice to develop the code to validate our techniques in short time and without many language related issues. Thank to this we gained time and resources to study and to develop the proposed solutions.

Python is a free and open-source general-purpose, high-level interpreted programming language. The Python's dynamic type system, the automatic memory management and the software design in general make it simple to learn and certainly a good instrument for prototyping in short time terms. Python interpreter is available for many platforms, this make it widely portable. As any general-purpose programming language it does not come with specific built in modules to operate with *csv* files.

In this section we are going to provide technical details about the preprocessing phase.

The Decompression Phase Since the original dataset files have been delivered in a compressed *.tgz* format a bash script have been wrote for the decompressing phase (Code Snippet **??**). The script is subdivided in two main parts: a for loop and a *parallel()* function. For every file except CRM ones (regular expressions have been used to filter file names) a new process that executes the *parallel()* function is spawned.

The `tar -xvf <compressed_file_path>` command decompresses the file (passed as parameter) in the script execution folder. Finally after the decom-

pression phase all the decompressed files are moved and renamed from the temporary decompression folder in the proper directory based on the nature of the file (MSC or NGPRS).

The compressed MSC files average dimension was around 500MB while decompressed about 2.5GB with a compressing ratio around 500% containing on average about 30 million logs per day, while NGPRS files originally before decompression on average occupy 2.5GB, and after about 15/16GB with a compression ratio close to 1000% with an average of 200 million logs per file.

```
[language=Bash, caption=Parallel Decompressing Script, label=code:decompressor.sh]
#!/bin/bash
# FILENAME: decompressor.sh


RES_FOLDER=/[path_to_the_archive]/;
TMP_FOLDER=/[path_to_tmp_working_directory]/;
DEST_FOLDER=/[main_path_of_destination_folder]/;

parallel(){

    # the file name to decompress is passed as parameter
    file=$1

    # decompressing command -x: extract, -v: verbose, -f folder name
    tar -xvf $file

    # building the name of the file
    fname_0=${file#$RES_FOLDER}
    fname=${fname_0%.tgz}

    # creating the name of the final file
    s_tmp=${file#$RES_FOLDER'motionl_MT_'}
    new_f_name=${s_tmp%.tgz}

    # Different destination folders depending on the type of file #RegEx
    if [[ $file =~ .*msc.* ]]; then
        # Move the file to the right folder
        mv -v  $TMP_FOLDER$fname $DEST_FOLDER"msc_temp/"$new_f_name

    elif [[ $file =~ .*ngprs.* ]]; then
        # Move the file to the right folder
        mv -v $TMP_FOLDER$fname $DEST_FOLDER"ngprs_temp/"$new_f_name

    else
        # Debugging log
```

```
        echo "NO MATCH! Skipping $file"

    fi

}

# for every file in the data resource folder
for f in $RES_FOLDER*
    do
        # if not a CRM file then (#RegEx)
        if !  [[ $f =~ .*crm.* ]]; then
            # parallel decompresses the file;
            parallel $f >> out.log &

        else
            # Debugging log
            echo "SKIPPED: $f"

        fi
    done
```

[Parallel Decompressing Script]

The Data Filtering Phase After the previously described decompression and the inconsistency test phases we started working on removing useless information, reshaping the data (both for the MSC and NGPRS data logs) trying to reduce the information at the minimum in order to make it properly fit with our specific project-related needs and minimize the disk overhead wile reading the data, these requirements has to be of a particular interest, in fact, in a real deployment scenario the network overhead has to be considered as the probable worst bottleneck on the whole system. We decided to do so both for matters of commodity and for performances reasons.

After this phase the files contained just the subscriber id, the time-stamp and the unique identifier of the cell tower.
Follow the data filtering and sorting scripts:

```
[language=Bash, caption=File Selection Script for\_files\_launch\_script.sh, label=code:
#!/bin/bash

f_list=($(ls motionl_MT_ngprs_T*))

for i in "${f_list[@]}"; do

printf $i"_sorted_4\n";
./minimize_and_sort.sh $i $i"_sorted_4";
```

```
done
```

[Data-Sets Matching]

```
[language=Bash, caption=File Selection Script for\_files\_launch\_script.sh, label=code:
#!/bin/bash

f_list=($(ls motionl_MT_ngprs_T*))

for i in "${f_list[@]}"; do

printf $i"_sorted_4\n";
./minimize_and_sort.sh $i $i"_sorted_4";

done
```

[Data-Sets Matching]

Code Snippet **??** we can observe *awk* and *sort* tools in action, awk for every line of the file passed as parameter to the script will filter-in just columns 2, 8 and 9 while the sort command will sort out all the lines based on the second column taking ';' as a column separator and using a memory buffer of 1GB.

After this phase the size of the files resulted to be way smaller in fact, the average NGPRS file from the original size of 15GB became around 7GB.

4 The User Interface

Map visualization In order to visualize the anomalies and statuses of the highway traffic, first a good map representation is needed. For the initial map, according to our research, Mapbox[5] is the suitable tool. Mapbox is lightweight and fast, renders at a high frame-rate, allowing the maps to fluidly respond to user feedback or scripted events and opening up a whole new class of apps. It is a vector based map and vector maps are roughly one-fourth the size of traditional raster implementations what means greater performance in low-bandwidth environments and greater cost savings where bandwidth is expensive. That means 75% size decrease compared to traditional raster maps.

Highway layers As a first step, the previously mentioned Hungary layer is placed on top of the map. After the Hungary layer is ready, the highway layer initialization can be started. As it have been stated in the preprocessing section, the highway GEOJSON contains a lot of short segments of the highways, keeping meta informations about them. Each of these segments/features is initialized with the normal traffic color. These initialized highways can be seen in Figure 17

[5] `https://www.mapbox.com`

Fig. 16. Initial map visualization

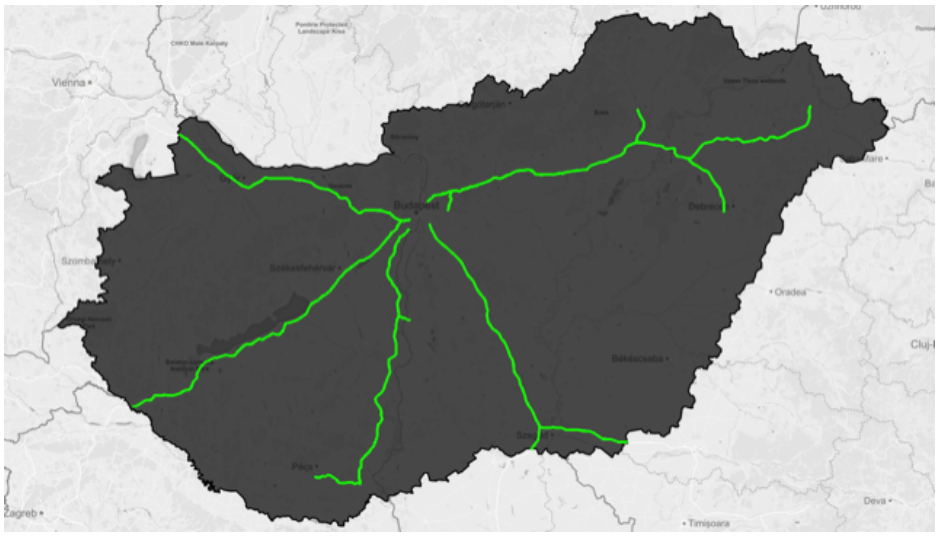

Fig. 17. Initial highway layers

After the connection is established with the server, the client starts to receive messages from the server. These messages have a list of statuses, each one of them has three attributes. These attributes are giving the information about the traffic situation changes. The *id* attribute identifies the tower, where the change happened. These *id*-s are mapped to the highway segments/layers through the competence file, which has been sent from the server as an initialization of the connection. After the mapping is done, the next attribute is the *value*, which is an Integer. It can go from 1 to 10. This value is showing the traffic load, according to the average traffic load, on that time and day of the week. The system is changing the colors of those layers according to their values. If it is a heavy traffic w.r.t. the average than it gets the shade of red. The shade is based on the difference from the average. Same with the low traffic load, it is going to turn into blue. An example of a lower traffic load, from a real life use case can be seen on Figure 18 As an opposite, a real life scenario, but heavier traffic load can be seen on the Figure 19

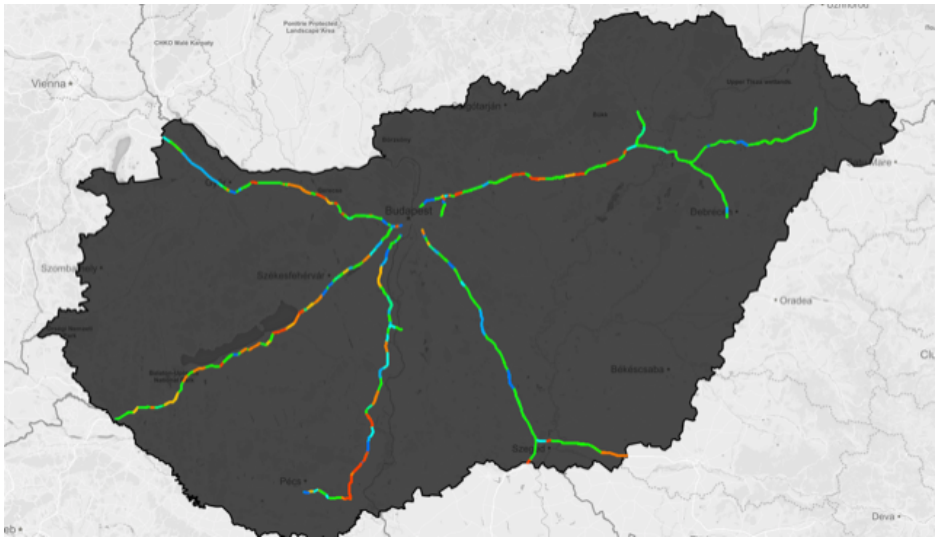

Fig. 18. Real life scenario: lower traffic

The third attribute is the *anomaly*, the purpose of this attribute is to show if there is a high difference. This can be used for alerts later on, and those alerts can trigger different functions. The attribute is not processed yet on the frontend layer, just preparing the field of the further work on signals and warnings.

Each of these layers are getting replaced when a new status message is arriving with their tower id. This is handled by an "updateFeature" function, which is first creating a layer on the top of the previous one and then deleting the old

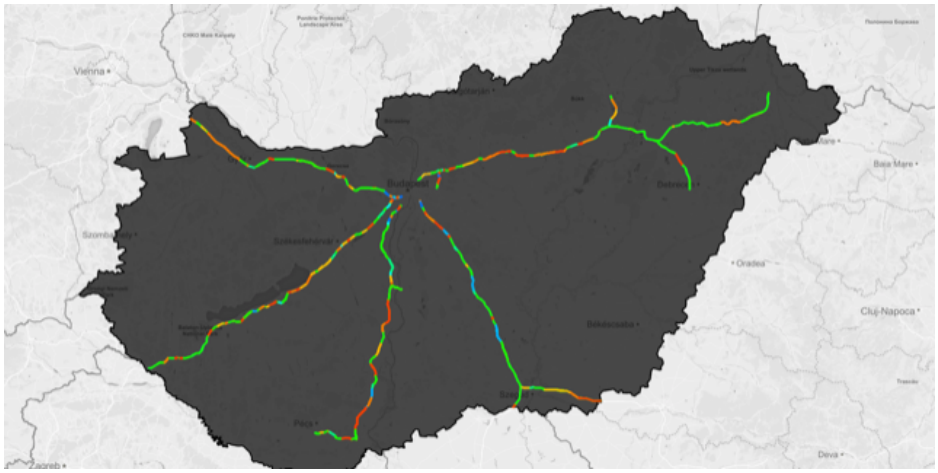

Fig. 19. Real life scenario: heavier traffic

one. This is the provided solution of the Leaflet library, but this method can be optimized later in the further work.

5 System Architecture

The data-flow in the whole process can be summarized in few points:

1. Past Data Loading
2. Data Structures Creation
3. Log Loading
4. Server Publishing
5. Continuous Log Processing*
6. Continuous Client Updating*

*where these operations continuously alternate each other. The action diagram describing the application system from the startup to the client serving is presented in the Figure 20.

Client-Server Communication: The communication from the two entities is always opened by the client. In fact, as a standard procedure the client first tries to open a WebSocket connection with the server. Once the server approves the connection and allows the communication, the client sends a message (namely: "getCompetence") asking to be sent a json file describing all the IDS (of the cell towers) and their set of highways slices they are competent for. Once this preliminary information exchange has finished the server will continuously update the client with the new statuses and the client will never send new messages to the server. Follows a sample of the *getCompetence* response and a diagram of the communication schema.

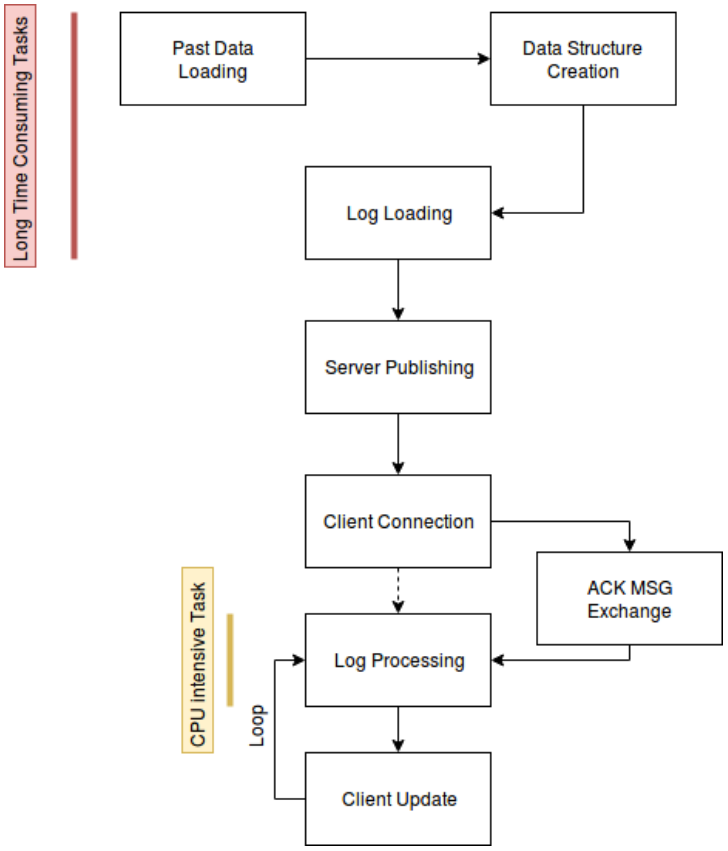

Fig. 20. Application System Action Diagram

```
{
    "1": ["way/4380568",...,"way/302433071"],
    "2": ["way/107536563",...,"way/319211427"],
    "3": ["way/109040038",...,"way/396567519"],
    ...
    "204": ["way/149403340",...,"way/312923558"],
    "205": ["way/90516720",...,"way/450062500"]
}
```


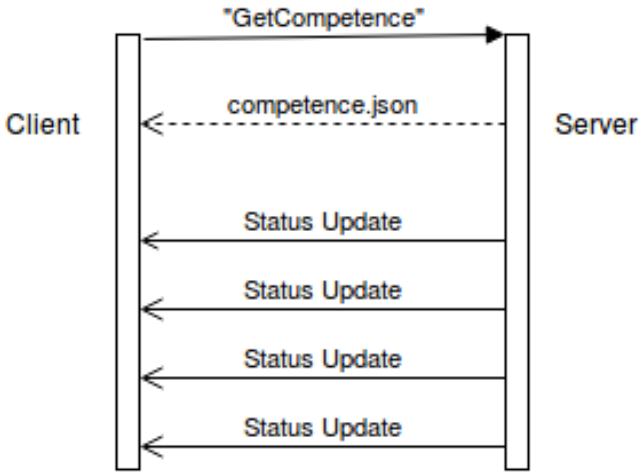

Fig. 21. Client-server communication

5.1 The Backend

Implementation details of the backend side of the application are introduced on this section.

The Application System Components The application-system we are going to introduce has its roots on several components, i.e. the following few data structures and logical units:

– **RTD (Relevant Towers Dictionary):** This Python's Dictionary is one of the main data-structure from which most of the others are based on. In fact, after the process of selecting the towers relevant to our system (*Relevant Towers*, namely the cell towers whose logs are strictly connected with the highways activities) the RTD have been stored and utilized. The RTD represents an HashMap having as keys all the Relevant Towers (represented by Tower Location ID denoted as TLIDs) and as values the geographical coordinates of the tower as a string.

- **URTD (UUID in Relevant Towers Dictionary):** This Python's Dictionary is an HashMap and it has as keys all the UUIDs (Unique User ID) of the users whose previous logs were triggered from Relevant Towers namely these logs who had the TLID (Tower Location ID) appearing in the RTD values. At the startup of the application system, this data structure will be empty and at the beginning of a new day it will be reinitialized.
- **TIDCD (Tower ID Counter Dictionary):** This Python's Dictionary is an HashMap and it has as keys all the Relevant Towers (namely TLID) whose appear to be in the incoming data-logs while it has as values counters initialized to 0 and representing the number of users whose the last log have been registered in that specific TLID (Tower Location ID) that is provided by querying the RTD. In fact, it can happen that given the multiplicity of antennas' generations to several different TID corresponds to the same TLID thus these sets of Cell Towers will be sharing the same counter.
- **Historical Data Dictionary:** This Python's Dictionary is an HashMap and it has as keys all the historically encountered TLIDs for every TLID corresponding to a list of values representing the crowdedness at a specific time-frame (the list's length depends on the size of the time frames).
- **Status Maintainer Module:** As soon as a subscriber performs an actions that triggers a log event inside or outside the area covered by a Relevant Tower, the Status Maintainer Module is delegated to keep track of the subscriber location (attaching him or her to the proper TLID in the URTD) and increases or decreases the counter of the proper TLID within the TIDCD accordingly to the situation. This module acts as a supervisor moving the subscribers from a tower to another, keeping the model up to date with the information provided by the event logs.
- **Evaluator Module:** The evaluator is the core module of the Application System responsible for classifying the status of the business of a specific highway segment. It uses as an evaluation model the model described in the previous section. The number of classes is parametric.
- **Notification Delegate Module:** The notification delegate module is the part of the Application System responsible to constantly collect the result of the Evaluator and update the clients about the highways status sending all the needed informations as a json payload.

Beyond the Status Maintainer (SM) As mentioned before, the Status Maintainer (SM) Module is delegated to interpret and take decisions based on the information provided from the logs with the following functionality:

As soon as the application is started, both the URTD and the TIDCD HashMaps will be empty. As the first incoming log having a TID present in the RTD key-set is processed, suddenly, the SM will fill the URTD with the UUID as key and the TLID as value (beyond the lines the TLID is inferred querying the RTD HashMap), then, it will initialize the counter of the TLID within the TIDCD to 1. If a second log from the same user will come but this time with a different *relevant* RTD and different TLID, then the counter of the

old TLID within the TIDCD will be decreased by one unit and suddenly the corresponding URTD's value will be updated to the actual TLID inferred from the log querying the RTD. In the last case, if the subscriber's handset will trigger a log which is not related to a Relevant Tower the counter of the old TLID within the TIDCD will be decreased by one and the key within the URTD corresponding to the specific UUID will be removed from the HashTable. Figure 22 provides a graphical illustration of the SM logic.

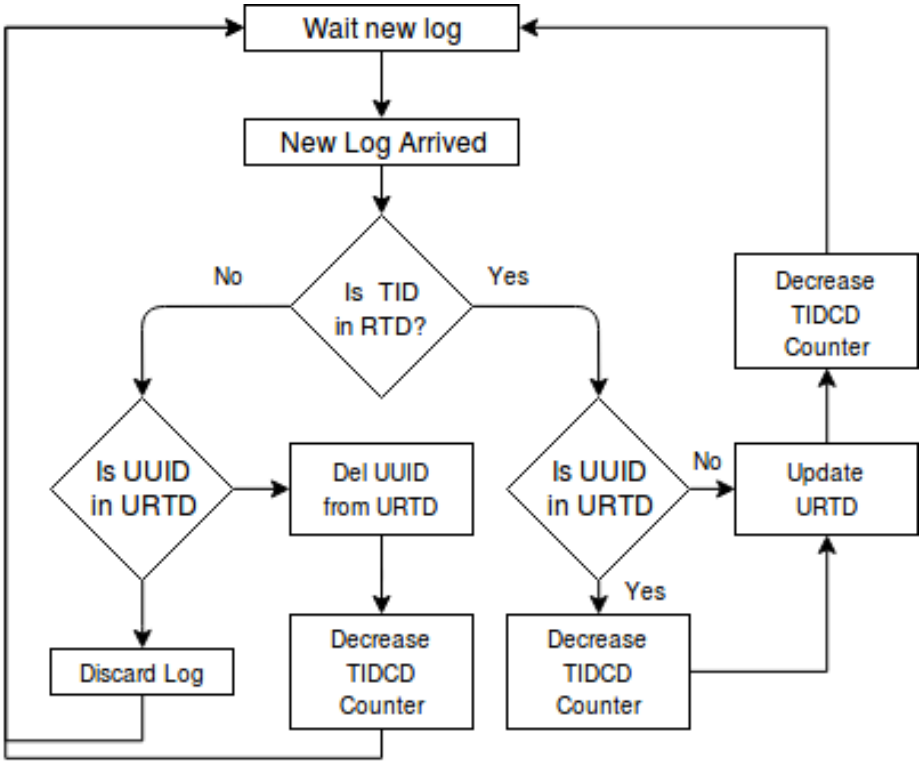

Fig. 22. Status Maintainer Logic

Beyond the Evaluator Module The evaluator module is triggered every time a time-frame (a parameter) expires. At this point it is time to evaluate the status of the whole system. At this stage the Evaluator iterates over all the TIDCD, computes all the means and standard deviations using the past data necessary to evaluate the actual status of all the TLID and then finally classifies every highway segment assigning a number to each of them. While this process takes place, an id conversion is performed. In fact, the backend and the frontend (for

security reasons) do not share the same internal ids for representing the TLID. Once finished, the control is passed to the Notification Delegate.

The Notification Delegate This module collects all the results, format them into a json file according to the specifics of the client and finally it sends the result through a WebSocket. Interesting to note is that the notification Delegate makes use of Python's await (asynchronous wait) feature, thus, in case the payload is pretty big, or the network suffers from speed issues, this operation won't block the whole Application System. In fact, in the meanwhile the application will be able to process new incoming logs. Follows a sample of the *json* payload:

```
[
  {"value": 5, "anomaly": false, "id": 236},
  {"value": 7, "anomaly": true, "id": 164},
  {"value": 5, "anomaly": false, "id": 288},
  ...
  {"value": 4, "anomaly": false, "id": 295},
  {"value": 2, "anomaly": false, "id": 14},
  {"value": 1, "anomaly": false, "id": 188}
]
```

Historical Data Representation The data from the past is stored in an efficient to load binary format making use of Python's Pickle module. Files are stored in a daily format in the form of a dictionary of arrays having as keys the TLIDs. Once loaded, the files are reshaped in a MxN matrix where M is the number of time-frames and N represents the number of days to consider in the past. All the tests were performed setting the time frames at 15 minutes and considering the last 15 days in the past. This representation have been chosen because once the system will be running in real-time with a continuous data-stream then it will be easy to shift the matrix data and recompute all the means and standard deviations efficiently. This shifting and re-computation operation would be performed once per day at a given time point (e.g. midnight) when the system is subject of a reinitialization.

5.2 The Frontend

Implementation details of the frontend side of the application are introduced on this section.

Communication For the frontend part, communications are handled with WebSockets which is an advanced technology that makes an interactive communication session between the client and the server side. It is an API, which can send messages to a server and receive event-driven responses without having to poll the server for a reply. It has three different interfaces, namely the

WebSocket, the CloseEvent and the MessageEvent. The WebSocket interface is responsible for opening the connections to the WebSocket server and then sending and receiving data on that connection. The CloseEvent is an event sent by the WebSocket object in case of connection lost. The MessageEvent is the most interesting interface for this project, which is an event triggered by the received message from the server. The Websocket is implemented in JavaScript using AJAX (asynchronous JavaScript and XML) technology.

On the Figure 23, the bidirectional communication is pointed out. This is much more useful than the traditional master-slave connection, where just one side can initialize any event. In this way the client can accept numerous status messages from the state of the highways and to make it interactive later on, the client is able to send messages to the server. From the other side, if anything exceptional occurs then the server can send messages about it to inform the client.

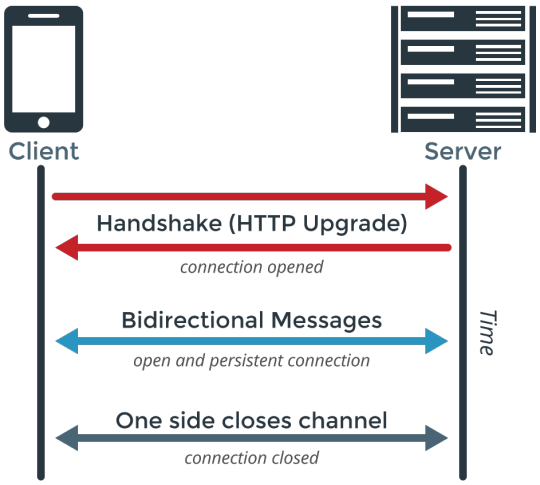

Fig. 23. Websockets diagram

As it was mentioned above, the MessageEvent interface is the most important int this project. In the following code snippet, a possible solution is visible for the communication protocol mentioned earlier in the backend section. In this solution, the three main event is shown: the CloseEvent which is the function to run, when the connection got lost. The OpenEvent which is the function to run by the client when the connection with the server is established, based on the communication protocol of the backend, the open event is sending a getCompetence message for the server. After the connection is ready, the server is sending the competence file, which is a mapping to the highway segments to the cell towers. When the client received the competence file, it is ready to process

the status messages from the server.

```
[language=java, caption=Communication protocol, label=code:commprotocol]
trafficSocket = new WebSocket("ws://CHOSEN IP:PORT");

    trafficSocket.onclose = function(){
      console.log("attempting again...")
      setTimeout(initConnection, 3333);
    }

    trafficSocket.onopen = function (event) {
      console.log("sent getCompetence!");
        trafficSocket.send("getCompetence");
    };

    trafficSocket.onmessage = function (event) {
      console.log("got competence");
      //competence = JSON.parse(event.data);
        trafficSocket.onmessage = function (myevent) {
          console.log("got update");
          processData(myevent.data);
      }
    }
```

6 The Model

The proposed model for detection of traffic jams and its implementation details are introduced in this section. In fact, the task at hand is to classify each part of the highway into several classes according to their load such that “full”, “more loaded”, “normal”, “less loaded” and “empty”.

6.1 The Classification Model

In order to classify the busyness of a segment of highway the best approach we find out to use is to consider the past data as a reference for the evaluation of the new entries. In order to explain and formalize this concept we need to provide some observations and definitions.

Provided that the usual activity of a cell tower close to the highway has a low static noise, the majority of the logged events come from the travelers it is easily possible to observe that the number of travelers in a given time-range tend to be similar for different days.

Provided the latter observation, we can build up a model such that in order to classify the busyness of a specific road segment in a precise time-range of the day (e.g.: between 12:00 and 12:15) we can compare the value to be classified with the past value of business within the same time range of the previous days.

All the data over time can be represented as a time series.

Provided that a real-valued time series can be represented in a symbolic representation through an arbitrary discretization of its values making use of the Arithmetic Mean and the Standard Deviation of its values we decided to use this technique as a classifier. In order to archive this objective we exploited some of the standard steps to compute the SAX (Symbolic Aggregate approXimation) representation of a time-series [**?**].

SAX allows a time series of arbitrary length n to be reduced to a string of arbitrary length w, ($w < n$, typically $w \ll n$). The alphabet size is also an arbitrary integer a, where $a > 2$. Our discretization procedure is unique in that it uses an intermediate representation between the raw time series and the symbolic strings. We first transform the data into the Piecewise Aggregate Approximation (PAA) representation and then symbolize the PAA representation into a discrete string.

We simply reduce the time series from n dimensions to w dimensions, the data is divided into w equal sized frames. The mean value of the data falling within a frame is calculated and a vector of these values becomes the data-reduced representation. Figure 24 gives a graphical description of the concept.

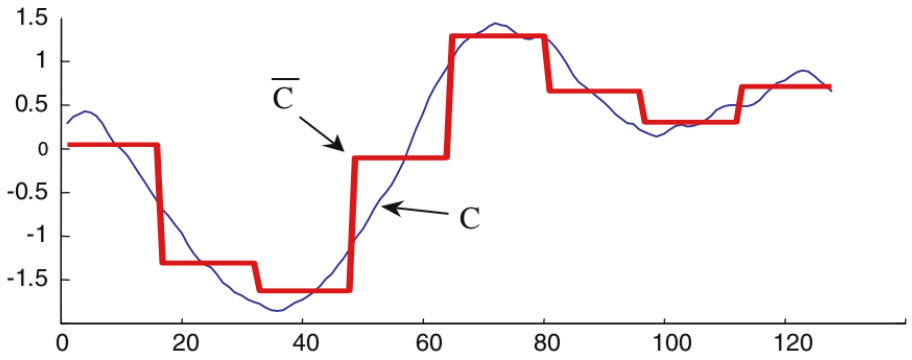

Fig. 24. PAA Example

Assuming that the distribution of the values over the time-series follow a normal distribution is possible to subdivide the time-series into so called *breakpoints* B. In our specific case we already have such a discrete representation of the time-series since our dataset is composed by discrete values over discrete time ranges.

These breakpoints may be determined by looking them up in a statistical table. Figure 25 illustrates a lookup table that contains the breakpoints that divide a Gaussian distribution in an arbitrary number (from 3 to 10) of equiprobable regions.

β_i \ a	3	4	5	6	7	8	9	10
β_1	−0.43	−0.67	−0.84	−0.97	−1.07	−1.15	−1.22	−1.28
β_2	0.43	0	−0.25	−0.43	−0.57	−0.67	−0.76	−0.84
β_3		0.67	0.25	0	−0.18	−0.32	−0.43	−0.52
β_4			0.84	0.43	0.18	0	−0.14	−0.25
β_5				0.97	0.57	0.32	0.14	0
β_6					1.07	0.67	0.43	0.25
β_7						1.15	0.76	0.52
β_8							1.22	0.84
β_9								1.28

Fig. 25. Statistical Table

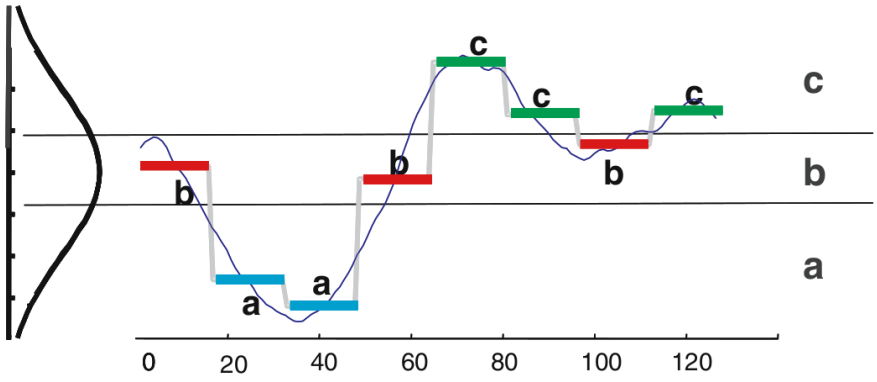

Fig. 26. Sax Representation of a Time-Series

As we can infer from 26 it is possible to classify every frame of a Time-Series within a discrete range of classes. The final key-concept of our model is that we are not going to represent the data in the past with a SAX representation, however we are going to classify a new entry using the breakpoints generated making use of the data recorded in past in a SAX-like fashion.

In order to describe the process of classification we need to formalize three more concepts: *Arithmetic Mean, Standard Deviation and Normalization*

Depending on the number of classes on which we would like to classify the new entry in the dataset, the right column of the statistical table has to be selected and then looked up, we sat up this value in a parametric manner in order to give a wider freedom of tuning to our model.

7 Experimental Results

TBD

You are strongly encouraged to use LaTeX 2$_\varepsilon$ for the preparation of your camera-ready manuscript together with the corresponding Springer class file `llncs.cls`. Only if you use LaTeX 2$_\varepsilon$ can hyperlinks be generated in the online version of your manuscript.

The LaTeX source of this instruction file for LaTeX users may be used as a template. This is located in the "authors" subdirectory in `ftp://ftp.springer.de/pub/tex/latex/llncs/latex2e/instruct/` and entitled `typeinst.tex`. There is a separate package for Word users. Kindly send the final and checked source and PDF files of your paper to the Contact Volume Editor. This is usually one of the organizers of the conference. You should make sure that the LaTeX and the PDF files are identical and correct and that only one version of your paper is sent. It is not possible to update files at a later stage. Please note that we do not need the printed paper.

We would like to draw your attention to the fact that it is not possible to modify a paper in any way, once it has been published. This applies to both the printed book and the online version of the publication. Every detail, including the order of the names of the authors, should be checked before the paper is sent to the Volume Editors.

7.1 Checking the PDF File

Kindly assure that the Contact Volume Editor is given the name and email address of the contact author for your paper. The Contact Volume Editor uses these details to compile a list for our production department at SPS in India. Once the files have been worked upon, SPS sends a copy of the final pdf of each paper to its contact author. The contact author is asked to check through the final pdf to make sure that no errors have crept in during the transfer or preparation of the files. This should not be seen as an opportunity to update or copyedit the papers, which is not possible due to time constraints. Only errors introduced during the preparation of the files will be corrected.

This round of checking takes place about two weeks after the files have been sent to the Editorial by the Contact Volume Editor, i.e., roughly seven weeks before the start of the conference for conference proceedings, or seven weeks before the volume leaves the printer's, for post-proceedings. If SPS does not receive a reply from a particular contact author, within the timeframe given, then it is presumed that the author has found no errors in the paper. The tight publication schedule of LNCS does not allow SPS to send reminders or search for alternative email addresses on the Internet.

In some cases, it is the Contact Volume Editor that checks all the final pdfs. In such cases, the authors are not involved in the checking phase.

7.2 Additional Information Required by the Volume Editor

If you have more than one surname, please make sure that the Volume Editor knows how you are to be listed in the author index.

7.3 Copyright Forms

The copyright form may be downloaded from the "For Authors" (Information for LNCS Authors) section of the LNCS Website: `www.springer.com/lncs`. Please send your signed copyright form to the Contact Volume Editor, either as a scanned pdf or by fax or by courier. One author may sign on behalf of all of the other authors of a particular paper. Digital signatures are acceptable.

8 Paper Preparation

Springer provides you with a complete integrated LaTeX document class (`llncs.cls`) for multi-author books such as those in the LNCS series. Papers not complying with the LNCS style will be reformatted. This can lead to an increase in the overall number of pages. We would therefore urge you not to squash your paper.

Please always cancel any superfluous definitions that are not actually used in your text. If you do not, these may conflict with the definitions of the macro package, causing changes in the structure of the text and leading to numerous mistakes in the proofs.

If you wonder what LaTeX is and where it can be obtained, see the "*LaTeX project site*" (`http://www.latex-project.org`) and especially the webpage "*How to get it*" (`http://www.latex-project.org/ftp.html`) respectively.

When you use LaTeX together with our document class file, `llncs.cls`, your text is typeset automatically in Computer Modern Roman (CM) fonts. Please do *not* change the preset fonts. If you have to use fonts other than the preset fonts, kindly submit these with your files.

Please use the commands `\label` and `\ref` for cross-references and the commands `\bibitem` and `\cite` for references to the bibliography, to enable us to create hyperlinks at these places.

For preparing your figures electronically and integrating them into your source file we recommend using the standard LaTeX `graphics` or `graphicx` package. These provide the `\includegraphics` command. In general, please refrain from using the `\special` command.

Remember to submit any further style files and fonts you have used together with your source files.

Headings. Headings should be capitalized (i.e., nouns, verbs, and all other words except articles, prepositions, and conjunctions should be set with an initial capital) and should, with the exception of the title, be aligned to the left. Words joined by a hyphen are subject to a special rule. If the first word can stand alone, the second word should be capitalized.

Here are some examples of headings: "Criteria to Disprove Context-Freeness of Collage Language", "On Correcting the Intrusion of Tracing Non-deterministic Programs by Software", "A User-Friendly and Extendable Data Distribution System", "Multi-flip Networks: Parallelizing GenSAT", "Self-determinations of Man".

Lemmas, Propositions, and Theorems. The numbers accorded to lemmas, propositions, and theorems, etc. should appear in consecutive order, starting with Lemma 1, and not, for example, with Lemma 11.

8.1 Figures

For LaTeX users, we recommend using the *graphics* or *graphicx* package and the `\includegraphics` command.

Please check that the lines in line drawings are not interrupted and are of a constant width. Grids and details within the figures must be clearly legible and may not be written one on top of the other. Line drawings should have a resolution of at least 800 dpi (preferably 1200 dpi). The lettering in figures should have a height of 2 mm (10-point type). Figures should be numbered and should have a caption which should always be positioned *under* the figures, in contrast to the caption belonging to a table, which should always appear *above* the table; this is simply achieved as matter of sequence in your source.

Please center the figures or your tabular material by using the `\centering` declaration. Short captions are centered by default between the margins and typeset in 9-point type (Fig. 27 shows an example). The distance between text and figure is preset to be about 8 mm, the distance between figure and caption about 6 mm.

To ensure that the reproduction of your illustrations is of a reasonable quality, we advise against the use of shading. The contrast should be as pronounced as possible.

If screenshots are necessary, please make sure that you are happy with the print quality before you send the files.

Please define figures (and tables) as floating objects. Please avoid using optional location parameters like "`[h]`" for "here".

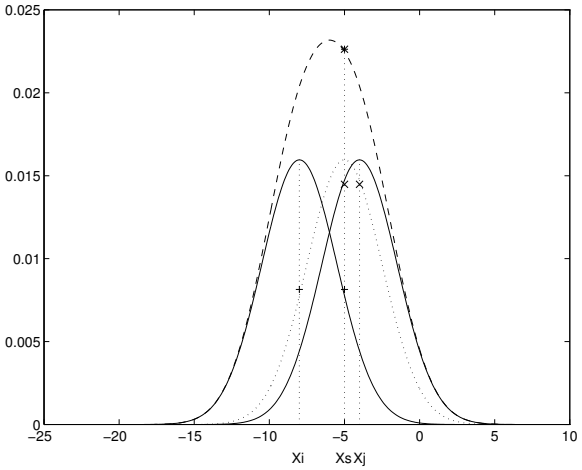

Fig. 27. One kernel at x_s (*dotted kernel*) or two kernels at x_i and x_j (*left and right*) lead to the same summed estimate at x_s. This shows a figure consisting of different types of lines. Elements of the figure described in the caption should be set in italics, in parentheses, as shown in this sample caption.

Remark 1. In the printed volumes, illustrations are generally black and white (halftones), and only in exceptional cases, and if the author is prepared to cover the extra cost for color reproduction, are colored pictures accepted. Colored pictures are welcome in the electronic version free of charge. If you send colored figures that are to be printed in black and white, please make sure that they really are legible in black and white. Some colors as well as the contrast of converted colors show up very poorly when printed in black and white.

8.2 Formulas

Displayed equations or formulas are centered and set on a separate line (with an extra line or halfline space above and below). Displayed expressions should be numbered for reference. The numbers should be consecutive within each section or within the contribution, with numbers enclosed in parentheses and set on the right margin – which is the default if you use the *equation* environment, e.g.,

$$\psi(u) = \int_o^T \left[\frac{1}{2} \left(\Lambda_o^{-1} u, u \right) + N^*(-u) \right] dt \; . \tag{1}$$

Equations should be punctuated in the same way as ordinary text but with a small space before the end punctuation mark.

8.3 Footnotes

The superscript numeral used to refer to a footnote appears in the text either directly after the word to be discussed or – in relation to a phrase or a sentence –

following the punctuation sign (comma, semicolon, or period). Footnotes should appear at the bottom of the normal text area, with a line of about 2 cm set immediately above them.[6]

8.4 Program Code

Program listings or program commands in the text are normally set in typewriter font, e.g., CMTT10 or Courier.

Example of a Computer Program

```
program Inflation (Output)
  {Assuming annual inflation rates of 7%, 8%, and 10%,...
   years};
   const
     MaxYears = 10;
   var
     Year: 0..MaxYears;
     Factor1, Factor2, Factor3: Real;
   begin
     Year := 0;
     Factor1 := 1.0; Factor2 := 1.0; Factor3 := 1.0;
     WriteLn('Year  7% 8% 10%'); WriteLn;
     repeat
       Year := Year + 1;
       Factor1 := Factor1 * 1.07;
       Factor2 := Factor2 * 1.08;
       Factor3 := Factor3 * 1.10;
       WriteLn(Year:5,Factor1:7:3,Factor2:7:3,Factor3:7:3)
     until Year = MaxYears
end.
```

(Example from Jensen K., Wirth N. (1991) Pascal user manual and report. Springer, New York)

8.5 Citations

For citations in the text please use square brackets and consecutive numbers: [1], [2], [4] – provided automatically by LaTeX's `\cite` ... `\bibitem` mechanism.

8.6 Page Numbering and Running Heads

There is no need to include page numbers. If your paper title is too long to serve as a running head, it will be shortened. Your suggestion as to how to shorten it would be most welcome.

[6] The footnote numeral is set flush left and the text follows with the usual word spacing.

9 LNCS Online

The online version of the volume will be available in LNCS Online. Members of institutes subscribing to the Lecture Notes in Computer Science series have access to all the pdfs of all the online publications. Non-subscribers can only read as far as the abstracts. If they try to go beyond this point, they are automatically asked, whether they would like to order the pdf, and are given instructions as to how to do so.

Please note that, if your email address is given in your paper, it will also be included in the meta data of the online version.

10 BibTeX Entries

The correct BibTeX entries for the Lecture Notes in Computer Science volumes can be found at the following Website shortly after the publication of the book:
`http://www.informatik.uni-trier.de/~ley/db/journals/lncs.html`

Acknowledgments. The heading should be treated as a subsubsection heading and should not be assigned a number.

11 The References Section

In order to permit cross referencing within LNCS-Online, and eventually between different publishers and their online databases, LNCS will, from now on, be standardizing the format of the references. This new feature will increase the visibility of publications and facilitate academic research considerably. Please base your references on the examples below. References that don't adhere to this style will be reformatted by Springer. You should therefore check your references thoroughly when you receive the final pdf of your paper. The reference section must be complete. You may not omit references. Instructions as to where to find a fuller version of the references are not permissible.

We only accept references written using the latin alphabet. If the title of the book you are referring to is in Russian or Chinese, then please write (in Russian) or (in Chinese) at the end of the transcript or translation of the title.

The following section shows a sample reference list with entries for journal articles [1], an LNCS chapter [2], a book [3], proceedings without editors [4] and [5], as well as a URL [6]. Please note that proceedings published in LNCS are not cited with their full titles, but with their acronyms!

References

1. Smith, T.F., Waterman, M.S.: Identification of Common Molecular Subsequences. J. Mol. Biol. 147, 195–197 (1981)

2. May, P., Ehrlich, H.C., Steinke, T.: ZIB Structure Prediction Pipeline: Composing a Complex Biological Workflow through Web Services. In: Nagel, W.E., Walter, W.V., Lehner, W. (eds.) Euro-Par 2006. LNCS, vol. 4128, pp. 1148–1158. Springer, Heidelberg (2006)
3. Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, San Francisco (1999)
4. Czajkowski, K., Fitzgerald, S., Foster, I., Kesselman, C.: Grid Information Services for Distributed Resource Sharing. In: 10th IEEE International Symposium on High Performance Distributed Computing, pp. 181–184. IEEE Press, New York (2001)
5. Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The Physiology of the Grid: an Open Grid Services Architecture for Distributed Systems Integration. Technical report, Global Grid Forum (2002)
6. National Center for Biotechnology Information, `http://www.ncbi.nlm.nih.gov`

Appendix: Springer-Author Discount

LNCS authors are entitled to a 33.3% discount off all Springer publications. Before placing an order, the author should send an email, giving full details of his or her Springer publication, to `orders-HD-individuals@springer.com` to obtain a so-called token. This token is a number, which must be entered when placing an order via the Internet, in order to obtain the discount.

12 Checklist of Items to be Sent to Volume Editors

Here is a checklist of everything the volume editor requires from you:

☐ The final LaTeX source files

☐ A final PDF file

☐ A copyright form, signed by one author on behalf of all of the authors of the paper.

☐ A readme giving the name and email address of the corresponding author.