Proceedings of the
2nd International Workshop on Mining Urban Data

Emerging Learning Paradigms and Applications for Smart Cities

Editors

Ioannis Katakis, National and Kapodistrian University of Athens

Francois Schnitzler, Technion

Thomas Liebig, TU Dortmund

Dimitrios Gunopulos, National and Kapodistrian University of Athens

Katharina Morik, TU Dortmund

Gennady Andrienko, Fraunhofer IAIS and City University London

Shie Mannor, Technion

List of Authors

Contents

2nd International Workshop on Mining Urban Data (Preface)

Ioannis Katakis KATAK@DI.UOA.GR
National and Kapodistrian University of Athens, Greece

François Schnitzler FRANCOIS@EE.TECHNION.AC.IL
Technion - Israel Institute of Technology, Haifa, Israel

Thomas Liebig THOMAS.LIEBIG@TU-DORTMUND.DE
University of Dortmund, 44221 Dortmund, Germany

Abstract

This paper presents an overview of the second International Workshop on Mining Urban Data (MUD2). The MUD2 workshop was held in conjunction with the 32nd International Conference on Machine Learning (ICML 2015) in Lille, France, July 11, 2015.

1. Introduction

We are gradually moving towards a smart city era. Many innovative applications arise daily utilizing massive urban data streams. Technologies that apply machine learning algorithms to urban data will have significant impact in many aspects of the citizens' everyday life. Examples of such applications include managing disastrous events, understanding the city's sentiment and opinion, tracking health issues, monitoring crucial environmental factors as well as improving energy efficiency and optimizing traffic. Unfortunately, urban data have some characteristics that hinder the state of the art in machine learning algorithms. Such are diversity, privacy, distributed and partitioned data, lack of labels, noise, complimentary of multiple sources and requirement for online learning. Many smart city applications require to tackle all these problems at once. This workshop aimed at discussing a set of new Machine Learning applications and paradigms emerging from the smart city environment. MUD2 will focus on presenting novel approaches that target some of the following applications: a) Traffic Management, b) Public Transport Adjustment, c) Accident Prevention, d) Resource Allocation, e) Energy Efficiency, f) Sentiment Analysis, g) Environment.

History A successful first edition of the MUD workshop was organized and co-located with EDBT/ICDT conference in March 24-28, 2014[1]. During the workshop 15 full and short papers were presented (50% full paper acceptance ratio). Approximately 30 people attended the workshop. As a follow-up, a special issue on Mining Urban Data has been organized (currently in the review phase) at the Journal of Information Systems[2]. 43 papers were submitted to the Special Issue. Workshops with a similar focus have been held recently: senseML, at ECML/PKDD 2014, and SenseMine, at ACM SenSys 2013. Finally, a workshop series dedicated to Computational Transportation Science is also organized.

2. Scope

The second version of MUD aimed at raising the awareness of the Machine Learning community on the challenges and opportunities of the Urban Data research arena. Mining Urban Data is a multidisciplinary field. The Data Management and Knowledge Discovery communities have already started working towards this direction (see first version of MUD at EDBT/ICDT conference[3]), however even though there are some first efforts from Machine Learning perspective, a greater ML involvement is required. ML will contribute to a better understand-

Proceedings of the 2nd International Workshop on Mining Urban Data, Lille, France, 2015.

[1] http://www.insight-ict.eu/mud/
[2] http://goo.gl/vUejYf
[3] http://www.insight-ict.eu/mud

ing and long term prediction of the urban sensor-citizen environment. The engagement of the community was achieved by inviting researchers and industries to present their work in the workshop. The workshop was open, but not limited to, the following topics:

Online learning - data generated from sensors can only partially/temporarily be stored. Thus, a major requirement is to process and analyse them as they arrive from the sources. Algorithms should be on-line and adaptive.

Large Scale Learning - the massive volume of data demands distributed / parallel processing technologies. Other issues include the complexity of the data coming from different sources with different spatial and temporal references or granularity.

Learning in Mobile Environment - special techniques are required for storing and learning in mobile environments.

Heterogeneous Data and Information Fusion - in many smart-city applications, different types of information (GPS, weather, Twitter, traffic data) should be analysed and combined in order to draw conclusions.

Learning with Social media - the main issue in mining micro-blogging data (e.g. Twitter) is that the text is very short, cursorily written and in different languages.

Event Detection - a very interesting research issue that arises from such data is the identification of real world events (e.g. "traffic jam", "accident", "flood").

Learning with Uncertain/Noisy Data - data generated by a smart city are typically very noisy. Uncertainty management procedures as well as crowdsourcing techniques might be required in order to aid the data models disambiguate the information.

Learning without Labels - with the size of the data sets and the associated area, labeling the full data set can be prohibitively expensive. Therefore, learning must typically be done with originally no or extremely few labels. Semi-supervised or active learning approaches could be very interesting for such applications.

Computer vision - CCTV cameras are a rich source of information. They can be used to count pedestrians, detect accidents, security etc.

3. Data Sets

We especially welcomed contributions based on data that can be reused by the community. Some published data sets are the following (the list is incomplete - please refer to the web site):

Dublin Bus GPS sample data from Dublin City Council. This data set comprises GPS traces of public buses within Dublin for a period of 1 month.
http://dublinked.ie/datastore/datasets/dataset-304.php

Traffic Volume Data for Dublin City. This data set originates from the traffic management system installed at Dublin city and contains performance values (traffic flow and saturation) at about 900 junctions for a period of 3 months.
http://dublinked.ie/datastore/datasets/dataset-305.php

Sales Prices in Helsinki. The dataset records advertised sales prices of 8337 residential apartments in Helsinki region along with features describing real estate characteristics, location characteristics, and accessibility in terms of travel distances and travel times. For more details see the paper Zliobaite et al (2015) published at the MUD2 workshop.
http://www.zliobaite.com/datahel.zip

An updated list of available data sets is available at the workshop website[4].

4. Invited Talks

This year, we welcome three invited speakers at the MUD workshop:

- Dr. Eleni Pratsini – Lab Director Smarter Cities Technology Center, IBM Research, Ireland, *"Using Big Mobile Data to Analyze Social Events in Cities"*
- Prof. Kristian Kersting – Fraunhofer IAIS and Technical University of Dortmund, Germany, *"Poisson Dependency Networks: Gradient Boosted Models for Multivariate Count Data"*
- Prof. Sharad Mehrotra – University of California, Irvine, USA *"Towards 'on the fly' data cleaning"*

5. Submissions and acceptance

We received 18 submissions from 11 countries. The number of authors per country is depicted in Figure 1. 15 submissions were accepted, 9 for regular talks and 6 for short talks. In the following, we present an overview of the submissions, grouped by topic.

5.1. Traffic

- Distributed Traffic Flow Prediction with Label Proportions: From in-Network towards High Per-

[4] http://www.insight-ict.eu/mud2/data.html

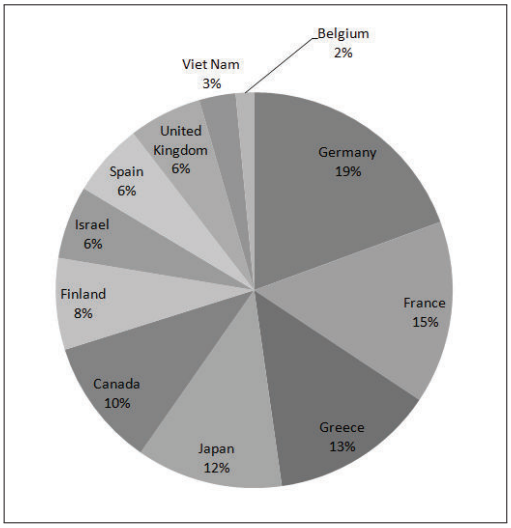


Figure 1. Distribution of MUD2 authors per country.

formance Computation with MPI, *Thomas Liebig, Marco Stolpe and Katharina Morik*

- Towards detection of faulty traffic sensors in real-time, *Nikolas Zygouras, Nikolaos Panagiotou, Nikos Zacheilas, Ioannis Boutsis, Vana Kalogeraki, Ioannis Katakis and Dimitrios Gunopulos*
- Car-traffic forecasting: A representation learning approach *Ali Ziat, Gabriella Contardo, Nicolas Baskiotis and Ludovic Denoyer*

5.2. Social Media

- Event-based Clustering for Reducing Labeling Costs of Incident-Related Microposts, *Axel Schulz, Petar Ristoski, Johannes Fürnkranz and Frederik Janssen*
- Modelling Time and Location in Topic Models, *Christian Pölitz*
- Stresscapes: Validating Linkages between Place and Stress Expression on Social Media, *Martin Sykora, Colin Robertson, Ketan Shankardass, Rob Feick, Krystelle Shaughnessy, Becca Coates, Haydn Lawrence and Thomas W. Jackson*

5.3. Trip Planning

- Automatic Extrapolation of Missing Road Network Data in OpenStreetMap, *Stefan Funke, Robin Schirrmeister and Sabine Storandt*
- Improved Trip Planning by Learning from Travelers' Choices, *Boris Chidlovskii*
- Report from Dagstuhl: SocioPaths - Multimodal Door-to-Door Route planning via Social Paths, *Thomas Liebig, Sabine Storandt, Peter Sanders, Walied Othman and Stefan Funke*

5.4. Bicycles and Public Transport

- Profiling users of the Velo'v bike sharing system, *Albrecht Zimmermann, Mehdi Kaytoue, Marc Plantevit, Céline Robardet and Jean-François Boulicaut*
- Accessibility by public transport predicts residential real estate prices: a case study in Helsinki region, *Indrė Žliobaitė, Michael Mathioudakis, Tuukka Lehtiniemi, Pekka Parviainen and Tomi Janhunen*
- On Predicting Traveling Times in Scheduled Transportation, *Avigdor Gal, Avishai Mandelbaum, François Schnitzler, Arik Senderovich and Matthias Weidlich*

5.5. Open, Mobile Data and Environment

- Analyzing Open Data from the City of Montreal, *Joelle Pineau and Pierre-Luc Bacon*
- Evaluating distance measures for trajectories in the mobile setting, *Nikolaos Larios, Christos Mitatakis, Vana Kalogeraki and Dimitrios Gunopulos*
- Airvlc: An application for real-time forecasting urban air pollution, *Lidia Contreras Ochando, Cristina I. Font Julián, Francisco Contreras Ochando and Cèsar Ferri*

6. Organization

Members of the Organizing Committee were: Ioannis Katakis (*National & Kapodistrian University of Athens*), François Schnitzler, (*Technion*), Thomas Liebig, (*TU Dortmund*), Gennady Andrienko, (*Fraunhofer IAIS and City University London*), Dimitrios Gunopulos, (*National & Kapodistrian University of Athens*), Katharina Morik, (*TU Dortmund*), Shie Mannor, (*Technion*). In addition, we would like to thank Marco Cuturi, workshop chair of ICML 2015 for the excellent collaboration.

Acknowledgemnts

The organizers received funding from the European Union's Seventh Framework Programme under grant agreement number FP7-318225, INSIGHT.

Analyzing Open Data from the City of Montreal

Joelle Pineau JPINEAU@CS.MCGILL.CA
McGill University, Montreal, CANADA

Pierre-Luc Bacon PBACON@CS.MCGILL.CA
McGill University, Montreal, CANADA

Abstract

There is a significant effort towards moving much of the data from the city of Montreal into an Open Data format. In this short paper, we report on a recent initiative to analyze this data using machine learning techniques in the context of a graduate course project. We review the approach, summarize accomplishments, and provide several recommendations for improving the impact from such efforts.

1. Introduction

Many cities worldwide have started to devote significant efforts and resources to publicly releasing data relating to their operations and situations. There is an opportunity for machine learning practitioners to use this data to answer several questions of interest for citizens, administrators, businesses, and researchers.

A course project was assigned in the context of a graduate course of Applied Machine Learning at McGill University. The stated goal of the project was to use open data from the city of Montreal's website to identify an interesting prediction question that can be tackled using machine learning methods, and solve the problem using appropriate machine learning algorithms and methodology. Previously, students had received 2 months of instructions on machine learning methods [1]. The course involved 65 students at various levels of their studies, from advanced undergraduate to Masters and PhD, 1 course in structure and 2 graduate teaching assistants. Course participants came from a diverse set of backgrounds, including computer science, electrical, mechanical and biomedical engineering, mathematics and statistics, epidemiology, neuroscience, environmental science. They worked in teams of 3 for this project.

1.1. Context and project instructions

According to instructions, participants were not restricted to using only the data from the city of Montreal website, though needed to use some of it. In particular, when appropriate, students were encouraged to incorporate data from other sources (e.g. equivalent data from other cities), or collect additional data (e.g. a new test set) to deepen their investigation.

The choice of prediction task and dataset to use was open. The goal was to pick a prediction question that is relevant and important to the citizens or administrators of the city. Particular attention was given to designing a prediction task that was well suited to the choice of dataset; and vice versa, picking the right data for tackling the chosen prediction question. The choice of algorithms and software systems was left open, including allowing use of existing machine learning toolboxes. The emphasis was on proper scientific methodology for computational analysis of urban data, rather than on the implementation of machine learning algorithms.

1.2. Characteristics of the city of Montreal dataset

The city of Montreal's Open Data resource[2] currently contains 177 datasets, organized under different themes, as listed in Table 1. Some datasets are re-listed under several themes, for example a dataset on the location and dimensions of community gardens appears under both the Environment and Housing and urban planning.

Several of these datasets include descriptive data, for example the list of municipal buildings in a particular borough, with their respective addresses, or a document describing the yearly accomplishments in terms of universal accessibility of buildings (municipal and others). In many

[1] The course syllabus:
http://www.cs.mcgill.ca/~jpineau/comp598/

[2] The data can be accessed here:
http://donnees.ville.montreal.qc.ca/

Proceedings of the 2^{nd} International Workshop on Mining Urban Data, Lille, France, 2015.

Table 1. Themes and number of datasets from the City of Montreal open data website.

Theme	Number of datasets
Organization and administration	54
Sports, leisure, culture and development	43
Infrastructures	28
Environment	27
Housing and urban planning	21
Financial resources	19
Election and referendum	17
Information management	16
Public safety and security	13
Communication and public relations	12
Material resources and services	9
Buildings and land	8
Economic development	7
Human resources	3
Legal affairs	2
Property assessment	1

respects, the data is not systematically or uniformly available: the list of municipal buildings is available for only one of the 19 boroughs of the city. The data is available in several formats (PDF, TXT, XLS, ODT, CSV, DOC, XML, KML, KMZ, GML, SHP, DXF, JSON, 3DM, ZIP), though each dataset is provided in a (small) subset of these formats.

2. Overview of projects and results

A total of 22 projects were completed, across a range of topics. Project titles are listed in Table 2. The primary challenge for most teams was to identify a dataset that contained enough data to perform a substantial machine learning analysis. This proved harder than expected, and thus several teams converged on using similar datasets from the set of 177 available. The most popular datasets pertained to the usage of the Bixi bike-sharing service, and data on the location of bicycling accidents. In some cases, participants complemented the available data with similar data from other cities, for example a project doing a comparative analysis of bicycle accidents in Montreal and New York.

A second challenge for many teams was to identify an appropriate prediction question, which was both feasible (i.e. sufficient available data) and interesting (i.e. with impact for citizens or administrators of the city). In some cases, the prediction question arose naturally out of the data, for example predicting the loan rate of library books. On the other hand, some participants were particularly creative with their choice of task. Good examples of this were found in the analysis of city images, which included a project aiming at the automatic colouring of historical images (originally taken in black&white).

Table 2. List of projects

Real estate
- Montreal Real Estate Pricing
- Prediction of Real Estate Property Prices in Montreal
- Location, Location, Location!

Transportation
- Estimating Traffic Levels in Montreal using Computer Vision and Machine Learning Techniques
- Predicting STM Bus Intervals Using Vehicles, Bicycles and Pedestrian Traffic Data
- Predicting Method of Transportation
- Biking Lane Usage Prediction
- BIXI Montreal
- Modeling imbalance in Bike Share Networks
- Predicting Bike Counts for BIXI Stations in Montreal
- Prediction Problems on Bike Accident and Usage Data in Montreal
- Prediction of Bicycle Accidents in Montreal
- Prediction of Bike Accidents, a Comparison of New York and Montreal
- Load Forecasting for Smart City with Possible Electrical Vehicle Penetration

Reconstruction/analysis of city images
- Where am I? Predicting Montreal Neighbourhoods from Google Street View Images
- Patch-Wide Classification of Historical Aerial Images of the Island of Montreal
- Reviving Old Montreal
- Object Recognition of Historical Datasets

Food safety
- Smart System for Restaurant Rating
- Predicting Severe Food Safety Violations in Toronto, Ontario

Library usage
- Predicting Montreal Library Book Loans
- Book Recommender Systems for Montreal Libraries

The choice of machine learning method to solve the chosen task was left open to the participants. In most cases, they needed to tackle the full pipeline, from feature extraction, to training the learner, to setting up a valid evaluation protocol. Many teams used common software libraries (e.g. scikit-learn (Pedregosa et al., 2011)) to assist with some portion of the work.

We now highlight a few of the projects.

2.1. Sample project: Prediction of real estate property prices in Montreal

This project aimed to predict the price of houses in Montreal. A total of 25,000 records were extracted from online listings of real estate brokers. Complementary infrastructure and geographical information for each listing was acquired from additional open data sources from the city of Montreal and Statistics Canada. Pre-processing was ap-

plied, for example removing properties with an asking price less than $10,000. Principal components analysis was used to project the feature space to a lower-dimensional space. Several machine learning algorithms were considered: linear regression, support vector regression, k-nearest neighbours, and random forest regression. Algorithms were implemented using the scikit-learn package (Pedregosa et al., 2011). The most promising results were obtained by an ensemble of k-nearest neighbour and random forest, achieving a prediction error on par with previous literature on similar datasets for other cities. In the case where the asking price of a house is included, prediction error of the selling price can be further reduced. Such a tool could be used by citizens to get a more accurate estimate of a property's market value. It may also be used by municipalities to assess property value for tax purposes. Finally, it may be used to inform economic indices.

2.2. Sample project: Biking lane usage prediction

This project aimed to predict the number of cyclists passing through different streets in Montreal on a given day. The analysis focused on ten different streets, and learned from daily counts obtained from sensors installed on the streets, over a period of dates between 2009 and 2013, with a total of 1722 records. Several features were considered, including the day of the week, weather, air quality index, price of gas, special events (festivals, football and hockey games), for a total of 47 features. This complementary data was extracted from various online sources. Several machine learning algorithms were considered: linear regression, k-nearest neighbours, boosted decision trees, and support vector regression. Prediction performance was assessed using the mean absolute error, as well as the ratio between the mean squared error for a given method and the mean squared error of a baseline (dummy) predictor. The boosted decision trees yielded the best performance. A complementary analysis of the feature impact using Lasso regression suggested that the day of the week was one of the most important features, possibly because the bicycle usage varies greatly between weekdays and weekends.

3. Discussion

In this section we discuss several opportunities and challenges that arose during the project.

3.1. Opportunities

From app design to data science. Many early open data efforts from large cities have focused on releasing descriptive data, amenable to app design, often used in the context of hackaton events. While such activities continue to be exciting and worthwhile endeavours, we believe that many communities have much to gain from also considering an open data strategy that leads to the release of urban data suitable for machine learning analysis. To meet this goal, the teams designing the open data platforms and controlling the information flow may need to acquire expertise about the goals and challenges of machine learning, in order to offer appropriate datasets. Computer scientists and statisticians have a role to play in informing these teams about the benefits that machine learning can bring to our society, and in providing convincing examples of cases where machine learning has enhanced the quality of life of citizens, and productivity of organizations.

Use of urban data to enhance transportation models. Several of the projects targeted the use of the city of Montreal data to predict various aspects of urban transportation, from the usage of the bike sharing service, to the expected timing of buses and automobiles. We observe that those datasets yielded some of the most interesting analysis because they were more extensive than other datasets, in terms of number of data points. The projects completed to date targeted specific aspects of the transportation network in isolation of others, however there is significant potential to combine such results into a coherent model of urban transportation, and eventually to use this model to evaluate different transportation strategies (e.g. adding bicycle lanes, changing bus routes, etc.)

Use of machine learning to enhance delivery of goods and services. Several of the projects attempted to use the available data to predict usage of various services, from the above-mentioned Bixi bike sharing service, to the borrowing of library books. Such analysis can be useful to make more efficient use of available municipal resources. However these cases pose particular challenges because the observed demand often depends on the availability of goods or services. So for example, one will not observe any demand for a particular book if that book was not available at the library. Similarly, it is difficult to accurately predict the real demand for the shared Bixis at a particular location once that station has no more bicycles available, and it is difficult to accurately predict demand at a new location. Some of the technical recommendations below relate to this aspect.

Use of machine learning to enhance human perception of urban data. One of the most original projects targeted the automatic re-coloration of old grey-scale images of the city. While the results so far were not fully satisfying, there is potential, as the methods improve, to use this technology to allow people to gain a new perspective on historical material. Some of the other projects relating to analysis of images have similar potential to enhance human understanding of the urban landscape, past or present.

Use of urban data as complementary data. A frequent use of the city of Montreal open data in the projects

listed above was as a supplement to other more extensive datasets. An example of this are the three projects pertaining to Real estate, where a large amount of data was first retrieved from real estate brokerage websites, and then complemented (via geo-location features) with city of Montreal data on local municipal infrastructure. Additional supplementary information was also considered, from sources such as Statistics Canada (for sociodemographic indicators), the YellowPages (for location of grocery stores, medical clinics, yoga studios, etc.) and public transit authorities (for bus and subway access locations).

3.2. Teaching challenges

Methods beyond the curriculum. Several of the projects required students to tackle machine learning methods that were beyond the basic course curriculum. The lectures for the course were not designed with the final project in mind, but rather to provided good coverage of basic algorithms and methods for applied machine learning in general. Fortunately, online resources are plentiful, and most students were able to acquire the necessary material in areas pertinent to their topic. In many cases however, understanding of that material seemed to be very superficial, and more opportunity for one-on-one learning would have improved the quality of the analysis.

Managing multiple projects. One of the familiar challenges with open-topic course projects is the load it creates in terms of supervision. The instructor and teaching assistants must have the time to provide individualized advice to each project team. We observed the most intense needs during the project definition phase, with some teams requiring up to 3-4 half-hour long meetings to properly define their scope and aims.

Scope of conclusions. We observed two challenges pertaining to the interpretation of the results. First, as with any data analysis, the urge can be strong to interpret the results in ways that are not warranted by the methodology used. For example, reporting results indicating that old aerial images of the city can be classified in terms of usage type (farmland, forest, residential, water) with 80% accuracy, but failing to state that the accuracy is in fact much lower for farmland and forests, but higher for water and residential areas. Second, while quantitative results are typically the preferred metric of performance, it is often the qualitative results that speak most to the human imagination. There is a tendency to pick a few select qualitative results to "tell a story"; this can be a powerful way of showing results, but it can easily be used to mis-characterize the expected performance of a system across the full range of events.

Presentation format. Two components were used for evaluation: an in-class 3-minute spotlight talk and a written report. The spotlight talks were preferred over long talks due to the number of projects. It proved difficult to provide accurate detailed evaluations from such short presentations, and so most of the feedback was qualitative. The spotlights talks were held roughly 2 weeks before the final report was due, and thus focused more on the problem definition and methods, with few results. The final report was formatted as a research paper, max. 8 pages in length, and provided a more accurate account of the project accomplishments. In previous years, a poster session was held, instead of the spotlights and written report. This format offers more opportunity for interaction between participants. The option was not retained this year due to scheduling constraints.

3.3. Practical challenges

Language of dataset. Most of the data available for the city of Montreal is in French. Few of the resources have been translated. Even in the case of quantitative data, the lack of English-language description posed an important problem for some of the young researchers.

Design of the prediction task. When working with previously used supervised machine learning benchmarks, the target problem (i.e. *output variable*) of interest has already been identified. When working with new datasets, it can be challenging to identify the right target variable. For example in the case of the projects pertaining to transportation, it may at first seem useful to predict the number or location of bicycle accidents within the city. However these events are relatively rare, and dealing with rare events is often challenging from a statistical and algorithmic perspective (especially in small datasets). An alternative may be to predict the number of close encounters between cyclists and vehicles, which are less rare, but such data is not typically available. Alternately, predicting the flow of larger vehicles (cars, buses, trucks) may be more fruitful, since it can be reliably estimated, and can be used within a larger predictive model on urban transportation.

Lack of parallel datasets. Comparative analyses (between years, between neighbourhoods) can yield rich information. This can only be tackled if data from parallel settings is available. The well-known Boston housing dataset (Harrison & Rubinfeld, 1978) was used as a comparison for some of the projects pertaining to real estate. In general, it is useful to keep this in mind when planning for additional releases of urban open data.

3.4. Machine learning challenges

Small data. The typical ICML attendee may be tempted to believe that all the interesting tasks for machine learning deal with so-called big-data. Yet several important problems occur in the small data setting. The challenges in this case are different, possibly less computational and

more statistical. There remains many opportunities to connect to the big-data community through the use of auxiliary datasets.

Sparse, incomplete, noisy datasets. As with most real-world datasets, a major problem with urban data remains the poor quality and uniformity of the data published. Often, the data is not curated by a person familiar with machine learning methods. There exists many statistical and machine learning methods to overcome problems of data quality, such as expectation maximization (Demptster et al., 1977), multiple imputation (Rubin, 1987). However the effective application of these approaches to complex datasets generally requires a good understanding of the methods (e.g. to construct a good model of imputation).

Feature coding for heterogenous data. Several projects observed that the choice of coding method for the data had a significant impact on the performance of their machine learning algorithm. For example in the bike lane usage prediction, an important feature was the day of the week. Encoding this as 7 binary features reduced the error rate by more than 5%, compared to using a single 7-valued categorical feature. Another similar effect was seen in the real estate price prediction task, where a logarithmic function was used to re-scale prices. Typically, the choice of encoding can be validated using standard methods for feature selection.

Feature selection for complex data. For some domains, the set of features that can be considered is very large, thus an important problem is in selecting the right set of features. Furthermore, it is often possible to enhance the feature set by incorporating supplementary data sources. It can be difficult to select the sufficient and necessary set of features for a given prediction task. Cross-validation methods can be used to automatically compare different feature sets. But this can be problematic in the case of small datasets where only limited data is available for validation of the feature set. An effective method in those cases is usually to use domain knowledge and expert advice to narrow down the candidate features to a manageable set (or small number of candidate sets). Another possible approach to tackle this problem is to use data from another city to predict the right set of features. Considering the case of Food Safety analysis, while Montreal has released only 750 records of food inspections (Montreal food data), San Francisco has released 10,000 records (San Francisco food data). Therefore one could optimize the choice of features using the San Francisco data and then apply the model and learn a simple prediction strategy on the Montreal data. More sophisticated methods for transfer learning are also worth investigating. Finally, it is worth pointing out that the choice of features can be key not just for building a good predictor, but also for building a good model for missing data imputation.

Choice of machine learning algorithm. There is a tendency among novice machine learning practitioners to spend significant efforts on testing several machine learning algorithms, with the belief that the choice of algorithm is the dominant factor in achieving good prediction performance. Another tendency is to assume that the most advanced methods will necessarily outperform more naive methods. In practice, several algorithms may perform equivalently, or simple methods may outperform more complicated ones, for example when there is insufficient data to properly train a complex hypothesis space, or the hyper-parameters are not properly optimized. Similar to the choice of features, algorithms can be compared using an appropriate cross-validation methodology.

Interpretability of results Methods such as linear regression, decision tree and naive bayes classifiers, are often preferred to more complex methods such as neural networks or kernel methods, in the case where interpretability of the results is necessary. In some applications, the knowledge of which features are most predictive of a particular outcome (e.g. finding which municipal amenities are best predictors of higher real estate prices) is of utmost interest. Several newer models have been proposed that combine rich hypothesis spaces with interpretability (Letham et al., To appear).

From supervised learning to decision-making So far we have been mostly concerned with *supervised* learning, where the goal of the learner is to predict a given quantity (the output) from observed variables (the input). In some cases, the goal may be to use the analysis to change a decision strategy. For example, by correctly predicting which restaurants may be found in violation of the health and safety laws, it may be possible to more efficiently deploy food safety agents. It is important to be aware of the fact that such a change in policy may result in a shift in the observed data. In the case where one wants to optimize the decision strategy, it may be more appropriate to phrase the problem under the framework of reinforcement learning (Sutton & Barto, 1998).

Off-policy learning A related case for concern arises when the data was acquired under a particular decision strategy, and the results of the analysis are used to change that decision strategy; in such case it can be difficult to accurately predict what will happen under the new decision policy. This is known as the *off-policy learning* problem in the machine learning literature (Sutton & Barto, 1998). Consider for example analyzing the usage data from Montreal's Bixi bike sharing service, then using the predictions derived from this analysis to determine which stations have lower demand, and then reducing bicycle availability at those stations. If those stations had low demand because

they were already subject to reduced availability, then the further shift to reduces availability likely would not result in more satisfied customers overall.

4. Conclusion

This paper presents a recent initiative to apply machine learning techniques to analyze open data from the City of Montreal data, conducted in the context of a graduate course project. Several of the challenges and opportunities identified are commonly known in the machine learning community. Our goal in presenting this work is to illustrate how such challenges arise in the context of analyzing urban data, and in doing so, facilitate collaboration with interested parties from other communities. While the City of Montreal was not involved in the elaboration of the course project, we have since communicated results of the projects with them. We have also received inquiries from officials of other cities. There is clearly significant interest in the outcomes of such initiatives.

Acknowledgements

Much of the credit for this paper goes to the students of the Fall 2014 edition of the course *COMP-598: Applied Machine Learning*, at McGill University. The first sample project on the prediction of real estate property prices was realized by Nissan Pow, Emil Janulewicz, and Liu Liu. The second sample project on the biking lane usage prediction was realized by Robert Wenger, Haomin Zheng, and Stefan Dimitrov. Several of the issues highlighted in the discussion were extracted directly from those and other students' project reports. Additional thanks go to Angus Leigh who acted as a teaching assistant for the course, jointly with Pierre-Luc Bacon.

References

Demptster, A.P., Laird, N.M., and Rubin, D.B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39, 1977.

Harrison, D. and Rubinfeld, D.L. Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management*, 1978.

Letham, B., Rudin, C., McCormick, T., and Madigan, D. Buildling interpretable classifiers with rules using bayesian analysis. *Annals of Applied Statistics*, To appear.

Montreal food data. http://donnees.ville.montreal.qc.ca/dataset/inspection-aliments-contrevenants.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: machine learning in python. *Journal of Machine Learning Research*, 12, 2011.

Rubin, D.B. *Multiple Imputation for Nonresponse in Surveys*. J. Wiley & Sons, 1987.

San Francisco food data. https://data.sfgov.org/health-and-social- services/restaurant-scores/stya-26eb?

Sutton, Richard S. and Barto, Andrew G. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998. ISBN 0262193981.

Improved Trip Planning by Learning from Travelers' Choices

Boris Chidlovskii BCHIDLOVSKII@XRCE.XEROX.COM
Xerox Research Center Europe, 6 chemin Maupertuis,38240 Meylan, France

Abstract

We analyze the work of urban trip planners and the relevance of trips they recommend upon user queries. We propose to improve the planner recommendations by learning from choices made by travelers who use the transportation network on the daily basis. We analyze individual travelers' trips and convert them into pair-wise preferences for traveling from a given origin to a destination at a given time point. To address the sparse and noisy character of raw trip data, we model passenger preferences with a number of smoothed time-dependent latent variables, which are used to learn a ranking function for trips. This function can be used to re-rank the top planner's recommendations. Results of tests for cities of Nancy, France and Adelaide, Australia show a considerable increase of the recommendation relevance.

1. Introduction

Most cities and agglomerations around the world propose their trip planners, in the form of a web or mobile application. Upon a user travel request, they recommend trips using a static library of roads and public transportation network and services. Although these planners are increasingly reliable in their knowledge of transportation network and available services, they all share the same static-world assumptions. In particular, they make a general assumption of *constancy and universality* (Letchner et al., 2006), that the optimal trip is independent of the time of day of the actual journey and of the passengers' preferences.

In reality, constancy and universality rarely hold. Most urban travelers can verify that the best trip between work and home at midnight is not necessarily the best choice to make between the same locations at 8am. Similarly, different passengers may choose different ways to travel between the same origin and destination points.

While the personal knowledge plays an important role, in many cases passengers simply have different preferences about the trip planning. For example, one passenger may avoid multiple changes, by extending the duration of her journey by a few minutes, while another passenger simply wants to arrive as quickly as possible to the destination.

When a user queries a planner for a journey from origin o to destination d starting at time t_s, there are often a large number of trips satisfying the query. Planners are designed to provide the k-top recommendations according to a set of predefined criteria, such as the minimal transfer time, the minimal number of changes, etc.. Their work is similar to any information retrieval system, where the goal is to place the most relevant documents among the k-top answers. Therefore, it is highly desirable that a trip planner behaves intelligently and suggests k-top trips which reflect the real passengers' preferences.

In this paper we closely analyze the cases of divergence between the planner recommendations and real choices made by urban travelers. We collect two sets of individual trips extracted from fare collection systems in cities of Nancy, France and Adelaide, Australia (see Figure 1). We compare these data to the city planners' recommendations; and in the of case of divergence, we propose a novel method to rank the trips that better reflects the reality.

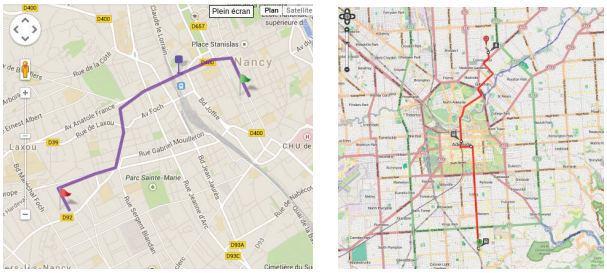


Figure 1. Trip planners of Nancy (left) and Adelaide (right).

Our method relies on two main contributions. First, we consider any individual trip as a set of explicit preferences made by the traveler during the trip. We use this set of pairwise preferences to learn a ranking function of trips.

Proceedings of the 2^{nd} International Workshop on Mining Urban Data, Lille, France, 2015.

This function is then used on the top of the trip planner, to re-rank the k-top recommendations. Second, we model passenger preferences of choosing a specific service or a change point in a way that reflects their dynamic nature. To address the sparse and noisy character of the raw trip, we model the user preferences by a set of dynamic latent variables. We estimate these variables by a smoothed dynamic non-negative factorization of service and transit counts.

The remainder of this paper is organized as follows. In Section 2 we briefly review the state of art in urban trip planning. Section 3 introduces the trip ranking problem by analyzing individual trips for Nancy city case. Learning to rank for trip planning is presented in Section 4. Then Section 5 proposes to model user preferences by dynamic latent variables and develop an estimation method by smoothed dynamic non-negative factorization of service and transit counts. In Section 6, we report results of evaluation on trip re-ranking for two city datasets. Section 7 concludes the paper.

2. Prior Art

Trip planners. Public transport (PT) trip planners are designed to provide information about available journeys in the transport system. The application prompts a user to input an origin o, a destination d and a departure time t_s (or arrival time t_f), it then deploys a trip planning engine to find a sequence of available PT services from o to d starting at time t_s (or ending at time t_f).

Trip planners often retrieve multiple trips for a user query. They typically use a variation of the time-dependent *shortest path algorithm* to search a graph of nodes (representing access points to the network) and edges (representing possible journeys between points) (Casey et al., 2014). Different weightings such as distance, cost or accessibility are often associated with each edge and node. Search may be optimized on different criteria, for example, the fastest, least changes or cheapest ones (Pelletier et al., 2009).

Planning high quality realistic trips remains difficult for several reasons (McGinty & Smyth, 2000). First, available General Transit Feed Specification (GTFS) sources rarely contain all information useful for constructing realistic plans. Second, the notion of "service quality" is difficult to define and is likely to change from person to person. Consequently, in real-world trip planning, the shortest trip is rarely the best one for a given user.

Multiple efforts have been made to improve the trip planning (Lathia & Capra, 2011; Liebig et al., 2014; Mokhtari et al., 2009; Trepanier et al., 2005; Yuan et al., 2011). Analysis of trip planner log files (Trepanier et al., 2005) can help improve transit service by providing better knowledge on transit users. Log files were useful for identifying new locations to be assessed for better understanding user behaviors, and for guiding updates of the PT information system.

Personalization of trip planning took into account user preferences and tries to identify the best trips among a set of possible answers. In (Mokhtari et al., 2009), the fuzzy set theory was used to model complex user preferences. A typology of preferences was proposed to explicitly express the preferences and integrate them in a query language.

Trip personalization by mining public transport data has been addressed in (Lathia & Capra, 2011). It established a relation between urban mobility and fare purchasing habits in London public transport network (Seaborn et al., 2010), and proposed personalized ticket recommendations based on the estimated future travel patterns and matching travelers to the best fare.

Integrating real time information in trip planners has been another research trend. (Yuan et al., 2011) presented a cloud-based system computing customized and practically fast driving routes for an end user using (historical and real-time) traffic conditions and driver behavior. GPS-equipped taxicabs are used as mobile sensors constantly probing the traffic rhythm of a city and taxi drivers' intelligence in choosing driving directions. The real time trip planning has also been extended to multi-modality (Casey et al., 2014; Seaborn et al., 2010). It used data from GPS-enabled vehicles to produce more accurate plans in terms of time and transit vehicles.It incorporates the delays into the transit network at real-time to minimize the gap with respect to the prediction model.

Learning to Rank. In document retrieval, to ranking documents based on their degrees of relevance to a query has been the key question for decades. Much effort has been placed on developing document ranking functions. Early methods used a small number of document features (e.g., term frequency, inversed document frequency, and document length), with an empirical tuning of the ranking function parameters. To avoid the manual tuning, the document retrieval was proposed to be regarded as *learning to rank* (Burges et al., 2005; 2006; Cao et al., 2006; Liu, 2011). Click-through data are used to deduce pair-wise training data for learning ranking functions.

In learning to rank, a number of categories are given and a total order is assumed to exist over the categories. Labeled instances are provided, and each instance is represented by a feature vector, and each label denotes a rank. Existing methods can be categorized as point-wise, pair-wise and list-wise (Liu, 2011). In point-wise methods, each instance with its rank is used as an independent training example. The goal of learning is to correctly map instances into intervals. In pair-wise methods, each instance pair is used as a training example and the goal of training is to correctly

find the differences between ranks of instance pairs, and ranking is transformed into pairwise classification or pairwise regression (Herbrich et al., 2000). This model formalizes learning to rank as learning for classification on pairs of instances and can deploy any classification method. In list-wise methods, the loss function is defined on a ranked list with respect to a query (Xia et al., 2008).

3. Individual trips analysis

We consider a public transportation system that offers a number of services (buses, trams, trains, etc.) to urban travelers. Any individual passenger trip J represents a sequence of PT services and changes between the services. Service legs of $\mathcal{J}$ form a sequence $\mathcal{S}_{\mathcal{J}} = \{l_1, \ldots, l_n\}, n \geq 1$, where leg l_i is a tuple $(s_i, b_i, a_i, t_i^b, t_i^a)$, s_i is a service identifier (a bus number, for ex.); b_i and a_i are boarding and alighting stops, t_i^b and t_i^a are boarding and alighting timestamps. Trip is *direct* if $n = 1$, and *transit* otherwise.

A transit trip includes $n-1$ changes which refer to *waiting* and/or *walking* between the services. The sequence of changes is defined as $\mathcal{C}_{\mathcal{J}} = \{c_1, \ldots, c_{n-1}\}, n \geq 1$, where c_i is uniquely defined by two successive service legs l_i and l_{i+1}, as $c_i = (a_i, b_{i+1}, t_i^a, t_{i+1}^b)$.

We make the following association between individual trips and trip recommendations. We consider a trip $\mathcal{J}$ as an *explicit answer* to an *implicit travel query* $Q = (o = b_1, d = e_n, t_s = t_1^b)$ or $Q = (o = b_1, d = a_n, t_f = t_n^a)$.

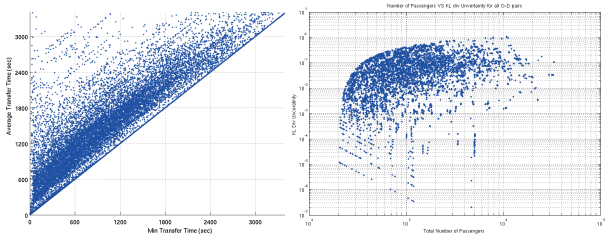


Figure 2. a) Minimal travel time vs average travel time. b) Trip uncertainty.

We analyze sets of individual trips collected from the automated fare collection systems (Mezghani, 2008) installed in Nancy, France and Adelaide, Australia; we mined these data to understand how passengers' choices differ from the planner recommendations.

For every pair of locations (o, d) in a network, we extract all real trips from o to d and analyze their travel time distribution. Figure 2.a shows the distribution of the *minimal* versus the *average* travel time for every (o, d) pair in Nancy. The high density zone suggests that the average travel time is far longer than the minimal time which is conventionally assumed by the planners.

Trip datasets expose a very large variety of paths for any (o,d) pair; the maximum number of different paths observed is 46 for Nancy and 37 for Adelaide; the average number of paths between two locations is 2.71 and 3.12, respectively. We measure the uncertainty of choosing one or another path from an origin o to a destination d, by using the Kullback-Leibler divergence $KL(q||p)$ of the trip distribution q from the uniform distribution p. The higher KL values indicate the higher certainty and a clear domination of one trip over others. Figure 2.b plots the KL divergence values for all (o,d) pairs in Nancy using the log-log scale. Again, the high density zone suggests that a large part of (o,d) pairs is dominated not by one but by 2 to 5 different paths of high frequency.

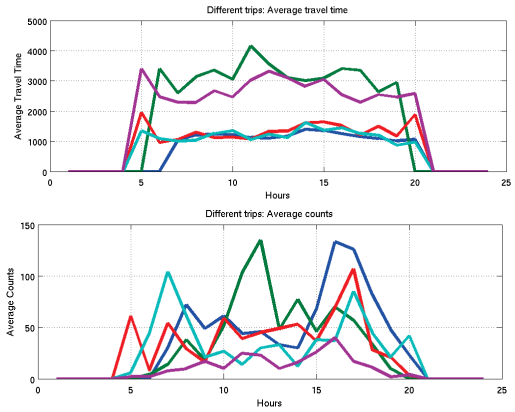


Figure 3. a) 5-top trips for one (origin, destination) pair in Nancy. b) The travel time and trip count distributions for top 5 trips.

It is important to recall that travelling preferences change during the day. Figure 3.a shows 5-top transit trips for an example (o, d) location pair in Nancy. Figure 3.b shows the travel time and average trip counts for 5-top trips for this example. All 5 trips are transit ones with one change. The figure reveals how the user preferences vary during the day. The trip planner recommends the trip shown in *red* for the fastest trip query. First, this recommended trip is not the fastest nor the most frequent one. Second, the trip shown in green is the most frequent during the lunch, despite it is far from being fast.

Figure 4 gives a more general picture. It shows 240 most frequent (o,d) pairs in Nancy. For each pair, Figure 4.a uses the different colors to show changing user preferences. The most frequent trip is colored in dark blue. Second, third, forth and fifth preferences are shown in blue, green, orange and brown colors, respectively. Trips are sorted by the distance between the origin and destination (see Figure 4.b).

Short trips expose a higher variability than longer ones. As the figure shows, the second choices are more visible (blue

color) during the morning rush hours. Figure 4.c shows the trip planner recommendations for the same pairs. The recommendations are static and do not reflect the user preferences.

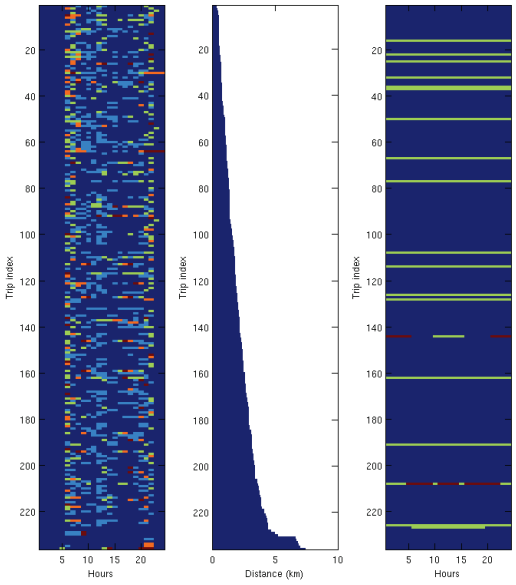


Figure 4. a) Changing user preferences for most frequent (o,d) pairs in Nancy. b) Trip distances. c) Trip recommendations by the planner.

We conclude this section by Figure 5 which shows how the user preferences vary between the PT services. It presents the total passenger counts for all Nancy change points, at 8am, 1pm and 6pm.

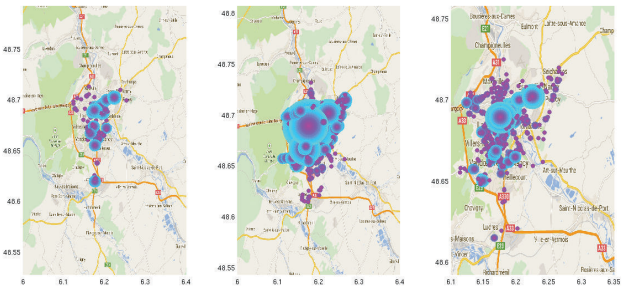


Figure 5. Change counts in Nancy at 8am, 1pm and 6pm.

4. Learning to rank trips

When a passenger travels from an origin o to a destination d at time t_s, she implicitly prefers the trip $\mathcal{J}$ she takes to all other trips $\mathcal{J}', \mathcal{J}' \neq \mathcal{J}$. Our approach is to transform this implicit feedback into an explicit set of pair-wise trip preferences and to learn the ranking function f from them.

Algorithm 1 below uses the trip planner and a set $\mathcal{T}$ of individual passengers' trips. For any trip $\mathcal{J} \in \mathcal{T}$ matching the query $Q = (o, d, t_s)$, the algorithm retrieves the k-top candidates for Q and retains that $\mathcal{J}$ has been preferred to any of these candidates, except $\mathcal{J}$ itself if it happens to be in this set. Real trip $\mathcal{J}$ matches a recommended trip $\mathcal{J}'$, if it has the same number of legs and following the same sequence of services. If $\mathcal{S}_\mathcal{J} = \{l_1, \ldots, l_n\}$ and $S_{\mathcal{J}'} = \{l'_1, \ldots, l'_n\}$, then $\mathcal{J}$ *matches* $\mathcal{J}'$ iff $s_i = s'_i \wedge b_i = b'_i \wedge a_i = a'_i$, for all $i = 1, \ldots, n$.

Algorithm 1 Rank learning algorithm.

Require: Collection $\mathcal{T}$ of passenger trips $\mathcal{J} = (\mathcal{S}, \mathcal{C})$
Require: Trip planner P with k-top recommendations
$S = \emptyset$; `set of pairwise preferences`
for each $\mathcal{J} \in \mathcal{T}$ **do**
  Form a query $Q = (o = b_1, d = a_n, t_s = t^b_1)$
  Query the planner P with query Q
  Retrieve k-top trips as a list L
  for each $\mathcal{J}' \in L, \mathcal{J}' \neq \mathcal{J}$ **do**
    Add $(Q, \mathbf{x}(\mathcal{J}) \succ \mathbf{x}(\mathcal{J}'))$ to S
  end for
end for
Learn the ranking model f from S
Ensure: f

Once the ranking function f is learned, it can be used to improve the relevance of trip planner recommendations acording to the *re-ranking scenario*. The trip planner does not change the way it works. And for a new user query Q, the trip planner first generates k-top candidate trips. Then these candidates are re-ranking using the function f.

To learn a ranking function f, Algorithm 1 requires every trip $\mathcal{J}$ be described by a feature vector $\mathbf{x}(\mathcal{J})$. In the following sections, we first describe a method for learning the ranking function f and then how to extract relevant and dynamic features from individual trips.

4.1. Gradient Boosting Rank

We used individual trips to form a set pairwise preferences, a ranking function f can be learned from. For each individual trip $\mathcal{J} \in \mathcal{T}$, we generate a set of labeled data $(\mathbf{x}_{i,1}, y_{i,1}), \ldots, (\mathbf{x}_{i,m_i}, y_{i,m_i})$, $i = 1, \ldots, |\mathcal{T}|$, which are preference pairs of feature vectors. If $\mathbf{x}_{i,j}$ has a higher rank than $\mathbf{x}_{i,k}$ ($y_{i,j} > y_{i,k}$), then $\mathbf{x}_{i,j} \succ \mathbf{x}_{i,k}$ is a preference pair, which means that $\mathbf{x}_{i,j}$ is ahead of $\mathbf{x}_{i,k}$. The preference pairs can be viewed as instances and labels in a new classification problem, where $\mathbf{x}_{i,j} \succ \mathbf{x}_{i,k}$ is a positive instance.

Any classification method can be used to train a classifier $f(\mathbf{x})$ which is then used for ranking. Trips are assigned scores by $f(\mathbf{x})$ and sorted by the scores. Learning a good

ranking model is realized by training of a model for pairwise classification. The loss function in learning is pairwise because it is defined on a pair of feature vectors.

The pairwise approach is adopted in many methods, including Ranking SVM (Herbrich et al., 2000), RankBoost (Freund et al., 2003), RankNet (Burges et al., 2005), IR SVM (Tsai et al., 2007), GBRank (Zheng et al., 2007), LambdaRank (Burges et al., 2006), and others. In the following we adopt GBRank as one of popular pairwise methods currently used.

GBRank takes preference pairs as training data, $\{\mathbf{x}_i^1, \mathbf{x}_i^2\}, \mathbf{x}_i^1 \succ \mathbf{x}_i^2, i = 1, \ldots, N.$ and uses the parametric pairwise loss function

$$L(f) = \frac{1}{2}\sum_{i=1}^{N}(\max\{0, \tau - (f(\mathbf{x}_i^1) - f(\mathbf{x}_i^2)\})^2,$$

where $f(\mathbf{x})$ is the ranking function and τ is a parameter, $0 < \tau \leq 1$. The loss is 0 if $f(\mathbf{x}_i^1)$ is larger than $f(\mathbf{x}_i^2) + \tau$, otherwise, the incurred loss is $\frac{1}{2}(f(\mathbf{x}_i^2) - f(\mathbf{x}_i^1) + \tau)^2$.

To optimize the loss function with respect to the training instances, the Functional Gradient Decent is deployed. Treating all $f(\mathbf{x}_i^1), f(\mathbf{x}_i^2), i = 1, \ldots, N$ as variables; the gradient of $L(f)$ is computed with respect to the training instances as follows

$$-\max\{0, f(\mathbf{x}_i^2) - f(\mathbf{x}_i^1) + \tau\}, \max\{0, f(\mathbf{x}_i^2) - f(\mathbf{x}_i^1) + \tau\}$$
$$i = 1, \ldots, N.$$

If $f(\mathbf{x}_i^1) - f(\mathbf{x}_i^2) \geq \tau$, the corresponding loss is zero, and there is no need to change the ranking function. If $f(\mathbf{x}_i^1) - f(\mathbf{x}_i^2) < \tau$, the loss is non-zero, and the ranking function is updated using the Gradient Descent:

$$f_k(\mathbf{x}) = f_{k-1}(\mathbf{x}) - \nu\Delta L(f_k(\mathbf{x})),$$

where $f_k(\mathbf{x})$ and $f_{k-1}(\mathbf{x})$ denote the values of $f(\mathbf{x})$ at k-th and $(k-1)$-th iterations, respectively, ν is the learning rate.

At the k-th iteration of the learning, GBRank collects all the pairs with non-zero losses $\{(\mathbf{x}_i^1, f_{k-1}(\mathbf{x}_1^2) + \tau), (\mathbf{x}_i^2, f_{k-1}(\mathbf{x}_i^1) - \tau)\}$ and employs Gradient Boosting Tree (Friedman, 2000) to learn a regression model $g_k(\mathbf{x})$ that can make prediction on the regression data. The learned model $g_k(\mathbf{x})$ is then linearly combined with the existing model $f_{k-1}(\mathbf{x})$ to create a new model $f_k(\mathbf{x})$ as follows

$$f_k(\mathbf{x}) = \frac{kf_{k-1}(\mathbf{x}) + \beta_k g_k(\mathbf{x})}{k+1},$$

with β_k as a shrinkage factor (Zheng et al., 2007).

5. Trip feature extraction

We now describe each real trip $\mathcal{J}$ by a set of relevant and dynamic features $\mathbf{x}(\mathcal{J})$. There may exist explicit and implicit factors which influence the passenger choice. Passengers make their choices in the function of location and time.

We mention two groups of trip features. First, **global features** describe the whole trip; they are the travel time, the number of changes, the usage of specific types of transport (bus, train, tram, etc.), multi-modality, etc. Second, much more relevant and specific are **local features** that describe each service leg and change that compose a given trip. For each PT service, we may extract the estimated means and variance of the speed when using this line at this time period, the average delay with respect to the schedule. For each change point, we can estimate the walking distance if any, the closeness to a commercial zone or transportation hub, etc.

Unfortunately, raw features of services and change counts are generally sparse, noisy and prone to many errors. Main reasons for errors are due to incorrect setup of ticket validation machines, lack of alignment between ticket validation machines and GPS localization, and card misuse by travelers.

So we intend to extract such latent features from sparse and noisy counts that be able to represent user preferences and their dynamic character.

We split all trips $\mathcal{J} \in \mathcal{T}$ in two collections of service and change observations, $\mathcal{A}^s = \{l_i | l_i \in S_\mathcal{J}, \mathcal{J} \in \mathcal{T}\}$ and $\mathcal{A}^c = \{c_i | c_i \in C_\mathcal{J}, \mathcal{J} \in \mathcal{T}\}$. In the following we assume for brevity working with a set of observations $\mathcal{A}$; it may indicate service or change observations, or their sum.

If we split all observations in $\mathcal{A}$ in T time periods, so we obtain a sequence of count matrices $\mathbf{A}_t, t = 1, \ldots, T, \mathbf{A}_t \in R_+^{p\times p}$ at time period t, where a_{ij} is the service or change count during the period t. and p is the number of stops.

The full diagram of latent feature extraction for individual trips and learning the ranking function is given in Figure 6.

5.1. Collapsed matrices

We first consider the static case when T is 1 and all observations from $\mathcal{A}$ are collapsed in one matrix $\mathbf{A}$.

Both service and change data are sparse non-negative counts, and we can use the non-negative matrix factorization (NNMF) as a method giving a great low-rank robust interpretation of data (Lee & Seung, 2001). They can be efficiently computed by formulating the penalized optimization problem and using modern gradient-descent algorithms (Hoyer, 2004).

Matrix A is approximated with a product ot two low-rank matrices that is estimated through the following minimiza-

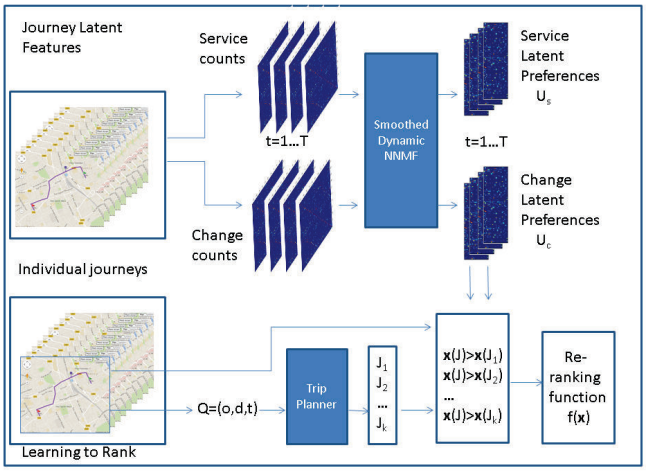


Figure 6. Preference features and re-ranking function learning.

tion

$$\min_{\mathbf{U}\geq 0,\mathbf{V}\geq 0}||\mathbf{A}-\mathbf{U}\mathbf{V}^T||_F^2,$$

where $\mathbf{U}$ and $\mathbf{V}$ are $n \times K$ non-negative matrices. The rank or dimension of the approximation K corresponds to the number of latent factors; it is chosen to obtain a good data fit and interpretability, where $\mathbf{U}$ give latent factors for origin stops and $\mathbf{V}$ does for destination stops.

The factorized matrices are obtained by minimizing an objective function that consists of a goodness of fit term and a roughness penalty

$$\min_{U\geq 0,V\geq 0}||\mathbf{A}-\mathbf{U}\mathbf{V}^T||_F^2+\lambda(||\mathbf{U}||_1+||\mathbf{V}||_1), \quad (1)$$

where the parameter $\lambda \geq 0$ indicates the penalty strength; a larger penalty encourages sparser matrices $\mathbf{U}$ and $\mathbf{V}$. Adding penalties to NMF is a common strategy since they not only improve interpretability, but often improve numerical stability of the estimation.

5.2. Smoothed Dynamic NNMF

In the general case $T > 1$, we have a sequence of matrices $\{\mathbf{A}_t\}_{t=1}^T$ for time periods $t = 1, \ldots, T$. To produce a sequence of low-rank matrix factorizations $\{\mathbf{U}_t, \mathbf{V}_t\}_{t=1}^T$, we can extend the factorization in (1) to the case $T > 1$ by independent factorization of T matrices $\{\mathbf{A}_t\}$. However, we additionally impose a smoothness constraint on both $\mathbf{U}_t$ and $\mathbf{V}_t$, in order to force the latent factors to be similar to the previous time periods, in both boardings and alightings. The objective function then becomes

$$\begin{aligned}&\min_{\mathbf{U}_t\geq 0,\mathbf{V}_t\geq 0}||\mathbf{A}_t-\mathbf{U}_t\mathbf{V}_t^T||_F^2\\&+\mu\textstyle\sum_{t=2}^T(||\mathbf{U}_t-\mathbf{U}_{t-1}||_2^F+||\mathbf{V}_t-\mathbf{V}_{t-1}||_2^F)\\&+\lambda(\textstyle\sum_{t=1}^T||\mathbf{U}_t||_1+||\mathbf{V}_t||_1),\end{aligned} \quad (2)$$

where parameters λ, μ are set by the user. The objective function imposes smoothing $\mathbf{U}_t$ and $\mathbf{V}_t$ on two successive time periods, but it can be generalized to a larger window.

To estimate matrices $\mathbf{U}_t$ and $\mathbf{V}_t$, we use an extended version of the multiplicative updating algorithm for NNMF (Gillis & Glineur, 2012; Lee & Seung, 2001; Mankad & Michailidis, 2013), based an adaptive gradient descent.

Temporal extensions of matrix factorization techniques have been studied in (Elsas & Dumais, 2010; Mankad & Michailidis, 2013; Saha & Sindhwani, 2012; Sun et al., 2014). (Elsas & Dumais, 2010) analyzed the temporal dynamics of Web document content. To improve the relevance ranking, it developed a probabilistic document ranking algorithm that allows differential weighting of terms based on their temporal characteristics. (Sun et al., 2014) addressed recommendation systems with significant temporal dynamics; it developed the collaborative Kalman filter which extends probabilistic matrix factorization in time through a state-space model. Community detection in time-evolving graphs is analyzed in (Mankad & Michailidis, 2013). The latent structure of overlapping communities is discovered through the sequential matrix factorization.

To solve (2), we follow (Mankad & Michailidis, 2013) and consider the Lagrangian as follows

$$\begin{aligned}&L=||\mathbf{A}_t-\mathbf{U}_t\mathbf{V}_t^T||_F^2+\\&+\mu\textstyle\sum_{t=2}^T(||\mathbf{U}_t-\mathbf{U}_{t-1}||_2^F+||\mathbf{V}_t-\mathbf{V}_{t-1}||_2^F)\\&+\textstyle\sum_{t=1}^T(\lambda(||\mathbf{U}_t||_1+||\mathbf{V}_t||_1)+Tr(\Phi\mathbf{U}_t)+Tr(\Psi\mathbf{V}_t)),\end{aligned} \quad (3)$$

where Φ, Ψ are Lagrange multipliers. The method works as an adaptive gradient descent converging to a local minimum. Kuhn-Tucker (KKT) optimality guarantees the necessary conditions for convergence [44]. The KKT optimality conditions are obtained by setting $\frac{\partial L}{\partial \mathbf{U}_t} = 0; \frac{\partial L}{\partial \mathbf{V}_t} = 0, t = 1, \ldots, T$. It can be shown that the KKT optimality conditions are obtained by

$$\begin{aligned}\Phi_t&=-2\mathbf{A}_t\mathbf{V}_t+2\mathbf{U}_t\mathbf{V}_t^T\mathbf{V}_t-2\mu(\mathbf{U}_{t-1}-\mathbf{U}_t)+2\lambda,\\\Psi_t&=-2\mathbf{A}_t^T\mathbf{U}_t+2\mathbf{V}_t\mathbf{U}_t^T\mathbf{U}_t-2\mu(\mathbf{V}_{t-1}-\mathbf{V}_t)+2\lambda,\end{aligned} \quad (4)$$

which after matrix algebra manipulations lead to the multiplicative updating rules presented in Algorithm 2.

The convergence of the multiplicative updating algorithm is often reported slow. In practice we obtain meaningful factorizations after a handful of iterations, which we tend to explain by the sparseness of input matrices A_t. In the future, when working with the dense data, faster methods like active set version of the alternating non-negative least squares (ANLS) algorithm (Kim & Park, 2008) will be more appropriate.

Algorithm 2 Dynamic Smoothing NNMF algorithm.

Require: Matrices $\mathbf{A}_t, t = 1, \ldots, T$, constants λ ,μ
Initialize $\mathbf{U}_t, \mathbf{V}_t$ as dense, positive random matrices
repeat
  for t = 1,. . . ,T **do**
    $\mathbf{U}_t \leftarrow \mathbf{U}_t(\mathbf{U}_t\mathbf{V}_t^T\mathbf{V}_t + \lambda\mathbf{A}\mathbf{U}_t)^{-1}(\mathbf{A}_t\mathbf{V}_t + \mu\mathbf{U}_{t-1})$
    $\mathbf{V}_t \leftarrow \mathbf{V}_t(\mathbf{V}_t\mathbf{U}_t^T\mathbf{U}_t + \lambda\mathbf{A}\mathbf{V}_t)^{-1}(\mathbf{A}_t^T\mathbf{U}_t + \mu\mathbf{V}_{t-1})$
  end for
until Convergence
Ensure: $\mathbf{U}_t, \mathbf{V}_t, t = 1, \ldots, T$

5.3. Dynamic trip features

Algorithm 2 finds sparse factorized matrices for a sequence of input matrices $\mathbf{A}_t, t = 1, \ldots, T$. We first apply the algorithm to sequences of service matrices $\mathbf{A}_t^s$ and change matrices $\mathbf{A}_t^c$, extracted from the full trip collection. We thus obtain smoothed factorized matrices $\mathbf{U}_t^s, \mathbf{V}_t^s$, and $\mathbf{U}_t^c, \mathbf{V}_t^c, t = 1, \ldots, T$ for services and changes, respectively. At time period t, a boarding stop b has latent factors given by a corresponding row in $\mathbf{U}_t^s$ this row is denoted $\mathbf{U}_t^s(b)$. For an alighting stop a, row $\mathbf{V}_t^s(a)$ gives the latent factors at time t. We then apply the algorithm to the sum matrices, $\mathbf{A}_t^f = \mathbf{A}_t^c + \mathbf{A}_t^s, t = 1, \ldots, T$. The smoothed factorized matrices for $\mathbf{A}_t^f$ are denoted $\mathbf{U}_t^f, \mathbf{V}_t^f$.

To generate a feature vector $\mathbf{x}$ for a trip $\mathcal{J}$, we may use its decomposition into service legs and changes, $\mathcal{J} = (\mathcal{S}, \mathcal{C})$. The vector $\mathbf{x}(\mathcal{J})$ is then composed of a general feature vector $\mathbf{x}_g$ and four latent components, $\mathbf{x}(\mathcal{J}) = \{\mathbf{x}_g, \mathbf{x}_b^s, \mathbf{x}_a^s, \mathbf{x}_b^c, \mathbf{x}_b^c\}$, where

- $\mathbf{x}_b^s$, $\mathbf{x}_a^s$ are latent feature vectors averaged over the trip boarding and alighting places, respectively,

$$\mathbf{x}_b^s = \frac{1}{n}\sum_{i=1}^{n} \mathbf{U}_{t_i^b}^s(b_i); \mathbf{x}_a^s = \frac{1}{n}\sum_{i=1}^{n} \mathbf{V}_{t_i^a}^s(a_i);$$

- $\mathbf{x}_b^c$, $\mathbf{x}_a^c$ are latent feature vectors averaged over the change places (alighting and boarding), respectively,

$$\mathbf{x}_b^c = \frac{1}{n-1}\sum_{i=1}^{n-1} \mathbf{U}_{t_i^b}^c(b_i); \mathbf{x}_a^c = \frac{1}{n-1}\sum_{i=1}^{n-1} \mathbf{V}_{t_i^a}^c(a_i).$$

In the case of sum latent matrices $\mathbf{U}_t^f, \mathbf{V}_t^f$, $\mathbf{x}(\mathcal{J})$ is composed of a general feature vector $\mathbf{x}_g$ and two latent components, $\mathbf{x}(\mathcal{J}) = \{\mathbf{x}_g, \mathbf{x}_b^f, \mathbf{x}_a^f\}$ obtained from $\mathbf{U}_t^f$ and $\mathbf{V}_t^f$.

6. Evaluation

To test our method for learning a ranking function from individual trips, we processed 5.2M individual trips collected in Nancy, France during 3 months in 2012. Nancy PT network includes 1129 nodes/stops and offers 107 bus and tram services to travelers. We also processed 12.5M trips from Adelaide, Australia collected during 2.5 months in 2013. Adelaide network offers 312 bus and tram service variations, and accounts for 3524 stops.

To evaluate the impact of modeling user preferences from actual trips, we selected 240 most frequent origin-destination pairs in Nancy (see Figure 4) and 160 most frequent pairs in Adelaide.

When generating temporal sequences of count matrices, we test two cases of $T = 24$ and T=48, when any matrix includes all passenger counts during one hour or 30 minutes. Once a matrix sequence is generated, any matrix is randomly split into 70% for training data and the remaining 30% for testing. All results below are means and variances over 10 independent runs.

We retrieved the trip planner recommendations for Nancy[1] and Adelaide[2]. We learn the ranking function and use it to re-rank the trip recommendations, using different options described in previous sections. To understand the effect of raw count factorization, we consider several options. First, we collapse matrices so disregarding the temporal aspect. Second, we consider either the service $\mathbf{A}_t^s$ and change matrices $\mathbf{A}_t^c$ separately, or sum them up $\mathbf{A}_t^f = \mathbf{A}_t^s + \mathbf{A}_t^c$ before the factorization. Third, we study the effect of temporal smoothing, when factorization is done either independent or by smoothing over successive time periods. Finally, we test different values K for the factorization.

In all experiments with GBRank (see Section 4.1), parameter τ was set to $\tau = 0.3$ and shrinkage factors β_k to 0.8. For smoothed dynamic NNMF, optimal values of μ and λ have been determined by cross-validation. For evaluating the results of ranking methods, we use a measure commonly used in information retrieval, Normalized Discounted Cumulative Gain (NDCG). We choose the perfect ranking's NDCG score 1 which is the error rate of the 1-top recommendation.

Table 6 reports the evaluation results for 12 different methods and compares them to the trip planner baseline for both cities. The analysis of these results provide some interesting insights. First, results are globally better for smaller Nancy than for bigger Adelaide, for both $T = 24$ and $T = 48$ cases. Second, collapsed matrices improve the baseline somewhat, but only taking into account temporal user preferences does really boost the performance. Moreover, smoothed matrix factorization improves considerably over the independent one. Third, the change latent variables appear to be more relevant than services ones. In-

[1] http://www.reseau-stan.com/
[2] https://www.adelaidemetro.com.au/

City	Nancy		Adelaide	
Method	$T = 24$	$T = 48$	$T = 24$	$T = 48$
Baseline: Trip Planner	24.91 ± 1.20	24.91 ± 1.28	38.17 ± 2.28	38.17 ± 2.28
Collapsed:Services	24.73 ± 1.17	24.73 ± 1.22	29.97 ± 2.11	29.97 ± 2.11
Collapsed:Changes	19.69 ± 1.01	19.69 ±1.09	28.63 ± 2.29	28.63 ± 2.29
Collapsed:Services+Changes	19.30 ± 1.14	19.30 ± 1.03	28.05 ± 2.32	28.05 ± 2.32
Collapsed:Sum	19.59 ± 1.13	19.59 ± 1.10	28.17 ± 2.18	28.17 ± 2.18
Indep: Services	14.08 ± 0.92	15.33 ± 0.97	25.33 ± 2.07	24.87 ± 1.87
Indep: Changes	9.55 ± 0.90	9.89 ± 0.86	23.89 ± 1.67	23.93 ± 1.75
Indep: Services+Changes	9.52 ± 0.89	9.41 ± 0.87	22.41 ± 1.72	22.15 ± 1.55
Indep: Sum	10.42 ± 0.89	9.37 ± 0.86	22.37 ± 1.56	23.55 ± 1.59
Smooth: Services	9.22 ±**0.77**	9.37 ± 0.78	15.37 ± 1.38	14.71 ± 1.24
Smooth: Changes	6.71 ± 0.82	6.69 ± **0.74**	16.69 ± **1.24**	16.69 ±1.15
Smooth: Services+Changes	**5.83** ±0.81	**6.12**± 0.79	**14.12**±1.29	**13.63**±**1.14**
Smooth: Sum	7.63 ± 0.79	7.05 ± 0.81	15.05 ± 1.41	14.43 ± 1.32

Table 1. NDCG@1 values for 12 methods and two cities.

stead, using sum counts performs worse than keeping service and change variables separately. We tend to explain this by heterogeneity of service and change preferences.

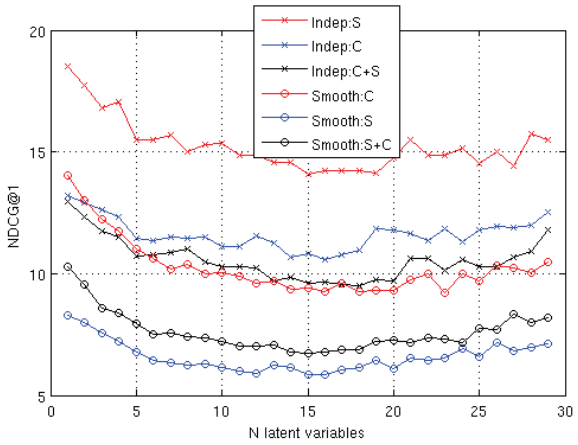


Figure 7. Independent an smoothed predictions vs Number of latent variables.

Figure 7 shows the performance of 3 independent and 3 smoothed methods for $T = 24$ for Nancy, with the number of latent variables K varying between 2 and 30. Surprisingly, already K=2 performs well enough, thus indicating the sparsity of the count matrices.

Figure 8 reports the hour-per-hour performance for the same six methods for Nancy case. Rush hours and lunch time appear to be hard for all methods; the error is the smallest for the periods 10am-12am and 2pm-4pm that points to the correlation between the traffic and trip variability. The traffic growth pushes travelers away from the conventional traveling choices.

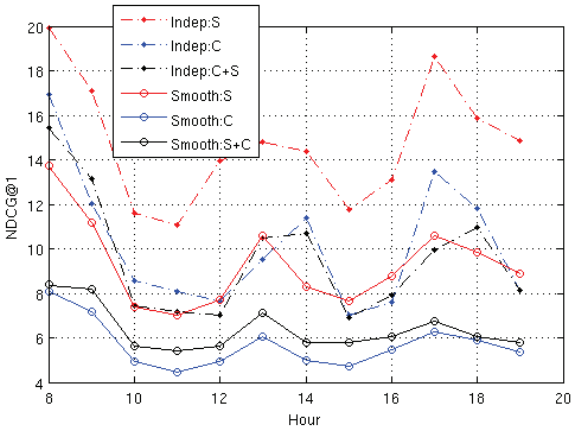


Figure 8. NDCG@1: Independent and smoothed predictions during the day.

7. Conclusion

We address the problem of relevance of trips recommended by urban trip planners. We analyzed passengers' trips extracted from two public transportation systems. We propose a method for improving the recommendation relevance by learning from choices made by travelers who use the transportation system daily. We convert the actual trips into a set of pairwise preferences and learn a ranking function using the Gradient Boosting Rank method. We describe actual trips with a number of time-dependent latent features, and develop a smoothed non-negative matrix factorization to estimate the latent variables of user preferences while choosing PT services and change points. Experiments with real trip data demonstrate that the re-ranked trips are measurably closer to those actually chosen by passengers than are the trips produced by planners with static

heuristics.

References

C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *ICML'05*, pages 89–96, 2005.

C. Burges, R. Ragno, and Q. V. Le. Learning to rank with nonsmooth cost functions. In *Proc. NIPS'06*, pages 193–200, 2006.

Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon. Adapting ranking svm to document retrieval. In *Proc. SIGIR '06*, pages 186–193, New York, NY, USA, 2006. ACM.

B. Casey, A. Bhaskar, H. Guo, and E. Chung. Critical review of time-dependent shortest path algorithms: A multimodal trip planner perspective. *Transport Reviews*, 34:522–539, 2014.

J. L. Elsas and S. T. Dumais. Leveraging temporal dynamics of document content in relevance ranking. In *Proc. WSDM '10*, pages 1–10, New York, NY, USA, 2010. ACM.

Y. Freund, R. D. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *J. Machine Learning Res.*, 4:933–969, 2003.

J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

N. Gillis and F. Glineur. Accelerated multiplicative updates and hierarchical als algorithms for nonnegative matrix factorization. *Neural Comput.*, 24(4):1085–1105, April 2012.

R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. In *Advances in Large Margin Classifiers*, pages 115–132, 2000.

P. O. Hoyer. Non-negative matrix factorization with sparseness constraints. *J. Mach. Learn. Res.*, 5:1457–1469, December 2004.

H. Kim and H. Park. Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. *SIAM J. Matrix Anal. Appl.*, 30(2):713–730, July 2008.

N. Lathia and L. Capra. Mining mobility data to minimise travellers' spending on public transport. In *Proc. ACM KDD'11*, pages 1181–1189, 2011.

D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Proc. NIPS'01*, pages 556–562, 2001.

J. Letchner, J. Krumm, and E. Horvitz. Trip router with individualized preferences: Incorporating personalization into route planning. In *Proc. IAAI'06 - Vol 2*, pages 1795–1800. AAAI Press, 2006.

T. Liebig, N. Piatkowski, C. Bockermann, and K. Morik. Predictive trip planning-smart routing in smart cities. In *EDBT/ICDT Workshops*, pages 331–338, 2014.

T.-Y. Liu. *Learning to Rank for Information Retrieval*. Springer, 2011.

S. Mankad and G. Michailidis. Structural and functional discovery in dynamic networks with non-negative matrix factorization. *Phys. Rev. E*, 88:042812, Oct 2013.

L. McGinty and B. Smyth. Turas: A personalised route planning system. In *Proc. PRICAI'00*, pages 791–791, Berlin, Heidelberg, 2000. Springer-Verlag.

M. Mezghani. Study on electronic ticketing in public transport. *European Metropolitan Transport Authorities (EMTA)*, 38:1–56, 2008.

A. Mokhtari, O. Pivert, and A. HadjAli. Integrating complex user preferences into a route planner: A fuzzy-set-based approach. In *IFSA/EUSFLAT Conf.*, pages 501–506, 2009.

M.-P. Pelletier, M. Trepanier, and C. Morency. Smart card data in public transit planning: A review. *CIRRELT Rapport 2009-46*, November 2009.

A. Saha and V. Sindhwani. Learning evolving and emerging topics in social media: a dynamic nmf approach with temporal regularization. In *Proc. WSDM'12*, pages 693–702, 2012.

C. Seaborn, J. Attanucci, and N. H. M. Wilson. Analyzing multimodal public transport journeys in london with smart card fare payment data. *Transportation Research Record: J. Transp. Research Board*, 2121:55–62, 2009.

J. Z. Sun, D. Parthasarathy, and K.R. Varshney. Collaborative kalman filtering for dynamic matrix factorization. *IEEE Trans. on Signal Processing*, 62(14):3499–3509, July 2014.

M. Trepanier, R. Chapleau, and B. Allard. Can trip planner log files analysis help in transit service planning? *Journal of Public Transportation*, 8(2):79–103, 2005.

M.-F. Tsai, Tie-Yan Liu, T. Qin, H.-H Chen, and W.-Y. Ma. Frank: a ranking method with fidelity loss. In *Proc. SIGIR'07*, pages 383–390, 2007.

F. Xia, T.Y. Liu, J. Wang, W. Zhang, and H. Li. Listwise approach to learning to rank: theory and algorithm. In *Proc. ICML'08*, pages 1192–1199, 2008.

J. Yuan, Yu Zheng, X. Xie, and G. Sun. Driving with knowledge from the physical world. In *KDD '11*, pages 316–324, New York, NY, USA, 2011. ACM.

Z. Zheng, K. Chen, G. Sun, and H. Zha. A regression framework for learning ranking functions using relative relevance judgments. In *Proc. SIGIR '07*, pages 287–294, New York, NY, USA, 2007. ACM.

Automatic Extrapolation of Missing Road Network Data in OpenStreetMap

Stefan Funke FUNKE@FMI.UNI-STUTTGART.DE
University of Stuttgart, 70569 Stuttgart, Germany

Robin Schirrmeister SCHIRRMR@INFORMATIK.UNI-FREIBURG.DE
University of Freiburg, 79110 Freiburg, Germany

Sabine Storandt STORANDT@INFORMATIK.UNI-FREIBURG.DE
University of Freiburg, 79110 Freiburg, Germany

Abstract

Road network data from OpenStreetMap (OSM) is the basis of various real-world applications such as fleet management or traffic flow estimation, and has become a standard dataset for research on route planning and related subjects. The quality of such applications and conclusiveness of research crucially relies on correctness and completeness of the underlying road network data. We introduce methods for automatic detection of gaps in the road network and extrapolation of missing street names by learning topological and semantic characteristics of road networks. Our experiments show that with the help of the learned data, the quality of the OSM road network data can indeed be improved.

1. Introduction

OpenStreetMap (OSM) is a huge collection of crowd-sourced spatial information. The goal of the OSM project is to map the whole world with all its road networks, buildings, regions and other kinds of natural and man-made entities. The OSM data set size increases significantly every year as more and more parts of the world are covered, and information becomes more detailed. For example, the world-wide road network in OSM contained at the beginning of 2007 less than 30 million data points whereas in 2013 this number has grown to more than two billions. Nowadays, the quality of OSM data often even exceeds the quality of proprietary data.

OSM is the basis for numerous applications and research projects, concerned e.g. with pedestrian and vehicle navigation (Holone et al., 2007; Vetter, 2010), location-based-services (Mooney & Corcoran, 2012), disaster warning (Rahman et al., 2012), fleet management (Efentakis et al., 2014), traffic estimation (Tao et al., 2012) and many other related topics[1]. Completeness of the road network data is mandatory for these applications to guarantee usability in practice. Road networks extracted from OSM (mainly Japan, Germany, North and South America and Australia) have also become standard benchmarks in route planning papers, see (Delling & Werneck, 2013; Baum et al., 2013; Funke et al., 2014). For research on route planning the completeness of the data plays an important role, as the developed algorithms are designed to take typical connectivity characteristics of road networks into account.

But while the OSM data is of very high quality already, there is still structural information missing, as e.g. road or path sections (see Figure 1). Moreover street names are far from being complete. The correct street name is necessary for specifying start and destination in a route planning query, for location-based services ('all shops on Norris Street') and for answering complex route planning queries like 'from A to B avoiding Park Street' accurately.

In this paper we design a classifier based on learning topological and semantic characteristics of road networks which can then be used to identify pairs of candidate locations where road segments are likely to be missing inbetween. We refer to missing structural data as *holes* in the following. Furthermore we show how to instrument machine learning techniques to identify road segments where the name tag can be extrapolated with high confidence. Our experimental results prove the ability of our methods to enhance the quality of OSM road network data considerably.

[1] http://wiki.openstreetmap.org/wiki/List_of_OSM-based_services

Proceedings of the 2^{nd} International Workshop on Mining Urban Data, Lille, France, 2015.

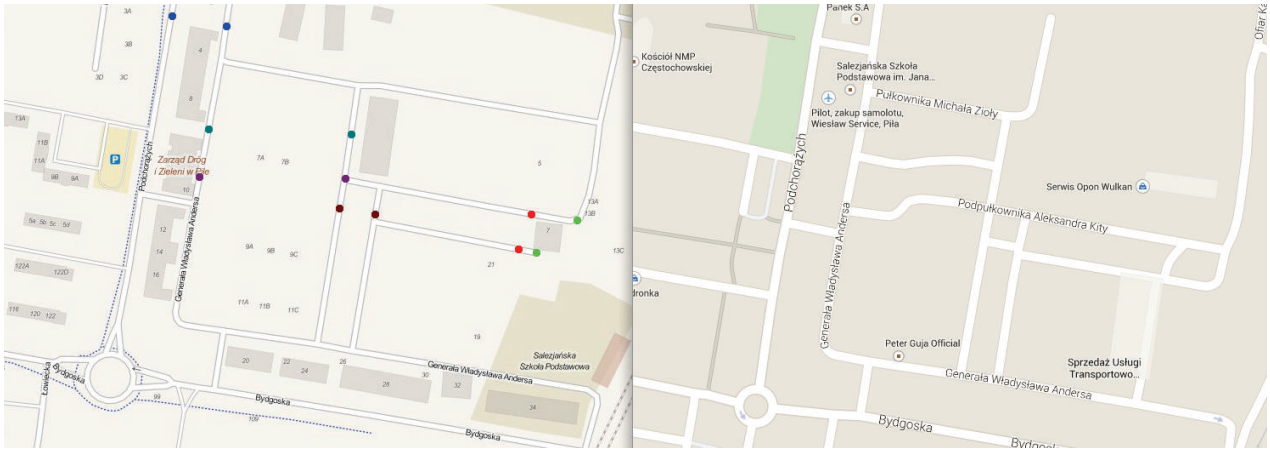


Figure 1. OSM based map (left) and GoogleMaps on a small cut-out of Poland. In the OSM map, the roundabout in the lower left corner is much more detailed and more building footprints and house numbers are available. For the coloured points in the left image, though, there are road segments missing as observable on the right.

2. Related Work

Studies on completeness and correctness of the OSM data are numerous, see e.g. (Girres & Touya, 2010) or (Haklay, 2010). The problem is that there either needs to be a ground truth one can compare to in an automated way (as investigated in (Fan et al., 2014) for building footprints in Munich), or data has to be manually compared to proprietary data. In both cases, the ground truth sample sizes are typically limited. Machine learning was applied to OSM data in order to automatically assess the quality of the road network data (Jilani et al., 2013b). Here, characteristics of certain street types (as e.g. motorways) are learned, including features such as total street length, number of dead-ends, number of intersection points and connectivity to other street types. In the quality analysis, feature vectors of streets are compared to the learned feature vector for the respective street type. If they do not resemble each other it is assumed that the data quality of the considered street is poor. The authors state that these learned features are also useful to classify streets with unknown type.

Preliminary results on automated quality improvement of OSM data were also reported in (Jilani et al., 2013a). Here, Artificial Neural Networks (ANN) are applied to distinguish residential and pedestrian streets; features include node count within a bounding box and betweenness centrality. In (Jilani et al., 2014), the automated street type classification for OSM was considered in more detail. In addition to the above mentioned features, the shape of the street is considered. About 20 different OSM street types were used in the experimental evaluation. The classification accuracy varies widely but for some types even an accuracy of 100% is achieved.

Further research on enhancing or correcting OSM data automatically (not necessarily using machine learning techniques) include the deduction of turn restrictions from GPS tracks (Efentakis et al., 2014), the detection of vandalism using a rule-based approach (Neis et al., 2012), or the identification of basic spatial units (so called parcels) for fine-scale urban modeling (Long & Liu, 2013).

To the best of our knowledge, tools for detecting missing parts of the OSM road network automatically were not investigated before, the quality assurance tools in the OSM project[2] mostly focus on detecting syntactic errors in the map specification. Hole detection in other kind of networks, as e.g. sensor networks, is an important and well established problem, though. Here also topological characteristics of the networks were taken into account (Funke, 2005). But as such networks differ significantly from road networks in many aspects results are hardly transferable.

3. Basics

OSM data comes in form of nodes, ways and relations. Nodes are single locations with latitude, longitude and additional tags (like the name of the location). Ways are ordered sets of nodes, describing e.g. a road or a building footprint. Relations are compositions of multiple nodes or ways, e.g. to aggregate all buildings and roads within an industrial area. Ways and relations are typically also augmented with tags that provide various information (e.g. the street or region name).

To be able to identify meaningful features for hole classification and street name extrapolation later on, we extracted all nodes and ways that describe roads in OSM and mod-

[2] http://wiki.openstreetmap.org/wiki/Quality_assurance

eled them into a directed graph $G(V, E)$ where V is the set of vertices and $E \subseteq \binom{V}{2}$ is the set of edges. Additionally, we define a weight function $w : E \to \mathbb{R}^+$ which provides the Euclidean length (computed on the sphere) of each edge. For given vertices $s, t \in V$ we also define the shortest path $\pi(s, t)$ as the path from s to t minimizing the summed weight of the edges $\sum_{e \in \pi} w(e)$.

Moreover we break down name tags associated with ways by assigning the respective name $n(e)$ to every edge e making up the way. Edges without names are labeled $n(e) =$ null. Similarly we associate with every edge e a type $t(e)$ as inherited from the respective way it is part of. Here, $t(e) \in \{0, 15\}$ and reflects the hierarchy of the network (small numbers indicate important streets as motorways, while high numbers refer to living streets).

4. Hole Detection in Road Networks

The basic question is how to identify pairs of locations in the network where road segments are very likely to be missing inbetween. To be able to deal with the enormous amount of OSM data, we aim for methods which work without the need for manually checking large portions of the data set. Therefore, we will apply machine learning to design a hole classifier which can be used to automatically check location pair candidates.

In the following we discuss several features that are relevant for hole detection. Obviously, connectivity characteristics of the network play an important role. Therefore, we will describe thoroughly how to measure connectivity between two nodes in a road network reasonably. Finally, we sketch the complete pipeline for hole detection, including the generation of suitable training data for the machine learning approach as well as a redundancy filter.

4.1. Road Network Characteristics

To identify holes, we make use of several characteristics of road networks. To be specific, we are interested in the following features:

- *Connectivity.* The most important feature is how well two locations are connected via the street network. Measuring connectivity is non-trivial and therefore discussed in detail in the next section.
- *Street type difference.* Missing links between locations exhibit most likely the same street type as the mapped streets adjacent to those locations. So a hole/missing link between an interstate and a dirt road is very unlikely. Therefore we compute the minimal street type difference for two locations v, w by iterating over all their adjacent edges $E(v), E(w)$ and calculating $\min_{e_1 \in E(v), e_2 \in E(w)} |t(e_1) - t(e_2)|$. If the street type is not available for one or more of these edge, we set the feature value to 0.
- *Node degree.* In typical (OSM) road networks, the average node out-degree and in-degree (i.e. number of outgoing/incoming adjacent edges) is about 2, the maximum rarely exceeds 9. Nodes with a high degree are typically important intersections and therefore have a better chance to be in a well mapped area in OSM. On the other hand, nodes with a low degree and especially dead-ends might be indicators for poor data coverage.

While street type difference and node degree can be computed quite easily, coming up with a reasonable measure for connectivity is more difficult. In the following, we design a measure based on the notion of *local stretch* that fits our purpose of hole detection well.

4.2. Measuring Connectivity

Intuitively, two locations in a road network that are in close proximity of each other should also have a short path within the street network. If the shortest path in the street network is much longer than the straight line distance, it might well be that parts of a street are missing.

We will first formalize this condition and provide empirical evidence for the assumption that the shortest path distance and the straight line distance are highly correlated in road networks. Subsequently, we describe common exceptions from this observation and introduce methods to deal with those.

4.2.1. Local Stretch Computation

Our goal is to automatically identify pairs of vertices $s, t \in V$ for which we assume that there exists a shorter path in reality than the one derived from the OSM network. We already outlined that the ratio of the shortest path distance between s and t and the straight line distance might be a good indicator. This ratio is called *local stretch*. In the following we refer to the length of a shortest path by $l(s, t) = |\pi(s, t)|$, and to the straight line or Euclidean distance by $d(s, t)$. Then the local stretch can be formally defined as $LS(s, t) = \frac{l(s,t)}{d(s,t)}$. As $d(s, t)$ is a lower bound for the shortest path length, the local stretch is always greater or equal to 1. The closer it is to 1 the better the connectivity between s and t in the road network. To compute $\pi(s, t)$ a Dijkstra run from s to t is the method of choice (or some accelerated variant).

In Figure 2, the correlation of straight line and shortest path distance is visualized via the local stretch value. We observe that for large shortest path distances the local stretch is remarkable small, in fact it converges to about 1.25 (so

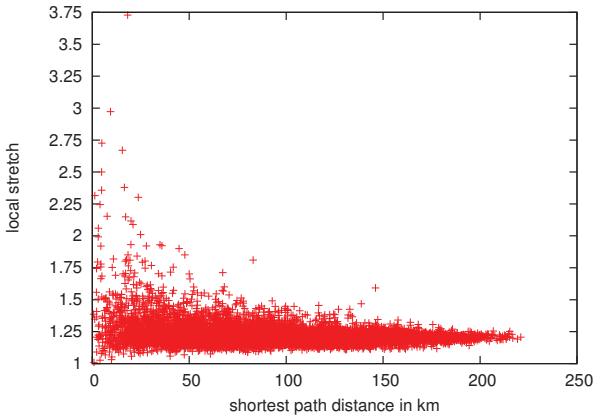


Figure 2. Local stretch in dependency of the shortest path length for 8,000 random point-to-point queries in Southern Germany.

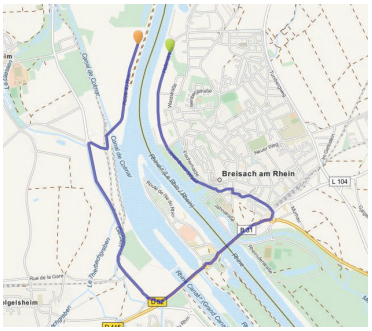


Figure 3. The straight line distance between the orange and the green marker is about 500 meters, but the shortest path distance is almost 8 kilometers, as the next bridge over the river is not close-by. This results in a local stretch value of 16.

the shortest path is only 25% longer than the straight line distance). For smaller shortest path distances (< 50 km), the local stretch values vary more and exceed 2 for some of the queries. Such point pairs with a higher local stretch than the average are good candidates for hole indication as they imply poor connectivity of the road network. Also it makes only sense to search for holes on a very local level anyhow. Holes between two far away locations are likely to be caused by (several) local holes, i.e. many missing road segments. Furthermore holes between two locations that are many kilometers apart are at least equally likely to result from poor infrastructure in the area than from missing road network data.

4.2.2. Incorporating Obstacles

Unfortunately, local stretch alone is not a sufficient measure for connectivity in road networks. Consider e.g. a river which can only be crossed via few bridges, then the local stretch for two points on opposite sides of the river is high as well (see Figure 3 for an illustration). Other kinds of natural or artificial obstacles have the same effect, as e.g. lakes or interstates.

One way to overcome this problem would be to add a post-processing phase in which for each identified pair of nodes with high LS it is automatically checked whether there is some obstacle between them. But this approach imposes several problems:

- *How to decide if an obstacle blocks the hole enough.* Consider e.g. the two green points in Figure 1 (left): There is a building on the straight line between them. Nevertheless, these two points indicate a real hole.
- *Increased Runtime.* If the number of pairs is large (e.g. along every river, we expect a multitude of candidates), then to check for every single one if there is a blockage inbetween is very time-consuming even if a suitable spatial data structure for managing the obstacles is used.
- *Distorted Learning.* In the end, we want to use connectivity as a feature in our machine learning approach for hole detection. If we consider these obstacle induced high LS values in the learning process, it might affect the ability to identify real holes later on.

To overcome these problems, we introduce an approach that avoids reporting such obstacle induced high LS values in the first place. The basic idea is to incorporate obstacles already in the local stretch computation phase. Comparing the shortest path distance to the straight line distance is somewhat unfair if the straight line is blocked with obstacles. Therefore we should rather compare the shortest path between two locations in the street network to the shortest path in the plane with movement-blocking obstacles, see Figure 4 (left) for an illustration. The exact computation of the shortest path with obstacles is rather complicated and expensive (see e.g. (Mitchell, 1996)), therefore we suggest an easy way to get the approximative distance: We construct a two-dimensional grid graph covering the whole area with a cell width of e.g. 10 meter. For every obstacle, we determine all grid points that are blocked by this obstacle and remove them and all adjacent edges from the grid graph. Then a conventional Dijkstra computation in the resulting grid graph provides a feasible path, see again Figure 4 (right). We refer to the length of this path as $g(s,t)$ in the following.

On this basis, we redefine local stretch as the ratio of $l(s,t)$ and $g(s,t)$ – abbreviated by $LS'(s,t)$. Note that due to the approximative nature of our shortest path length in the plane and the fact that bridges etc. are not incorporated in this calculation, LS' might be smaller than 1 (while LS always is ≥ 1). Still, the smaller LS', the better the connectivity between two locations in the road network.

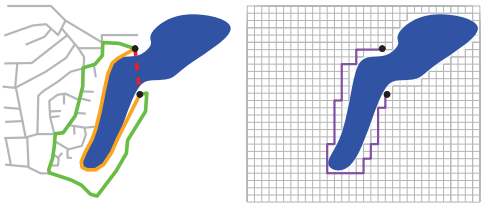


Figure 4. Left image: The shortest path (green) between the two black locations is much longer than the straight line distance (red). But it is not much longer than the shortest path in the plane with the lake considered as obstacle (orange). Right image: Approximative shortest path (purple) in the plane with obstacles using a grid approach.

4.3. Learning a Hole Classifier

With our newly designed connectivity measure LS', we are now able to compute all described road network features for hole detection. As already outlined above, we are going to search for holes only between locations with a small straight line distance as otherwise we cannot hope for good accuracy – furthermore, considering every pair of locations in a large road network is computationally infeasible.

4.3.1. Generating Training Data

Manual creation of a ground truth data set large enough for training the classifier is very time-consuming. Moreover, one needs to rely on the correctness/completeness of other data (e.g. GoogleMaps) for this purpose. Hence to construct a large ground truth set of classified node pairs, we used the following method: Nodes in the network that are directly connected with an edge ($LS = 1$) are no holes for sure. Also node pairs with a small local stretch do not indicate a hole with high confidence (we used 2 as a threshold in the experiments). We repeatedly selected a node in the network randomly and then searched for other nodes in close proximity with small LS value. Among those we randomly picked one to form a respective pair. For each such pair we computed the feature vector and added it to the training data set. To generate training data for actual holes, we used a similar approach but removed all edges on the shortest path between the two selected nodes before computing the feature vector. In this way, we created artificial holes. For the final evaluation of the accuracy of our method, real holes will be used.

4.3.2. Classifier Choice

Using the described feature vectors, the goal is to learn a good classifier which distinguishes between holes and non-holes. We expect the relationships between our features and the existence of a hole to be rather simple; for example, we expect the higher the local stretch the more likely there is a hole between two nodes. Due to these expected feature-target correlations, one suitable method for learning is *Logistic Regression*. Nevertheless, we also want to check whether there might be more complex relationships (e.g. considering node degrees). Therefore, we also used *Random Forest* as it might be able to exploit these more complex relationships.

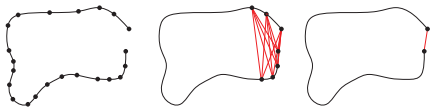


Figure 5. The left image shows a small cutout of a road network. In the middle image, hole candidates are indicated by red lines. The right image shows the single remaining hole after applying the extremeness check.

Both of these classifiers often work well with default parameters[3] and their learned models are fairly easily interpretable. This makes them more suitable for our task than e.g. Artificial Neural Networks.

4.3.3. Extremeness Check

Feeding all reasonable node pairs into the learned classifier provides us with the set of potential holes in the network. Unfortunately, it is very likely that a single missing road segment leads to a multitude of reported candidate node pairs. If between two vertices $s, t \in V$ a segment is missing, the classifier might not only declare s, t a hole but also s', t' with s' in close proximity of s and t' in close proximity of t. In the example in Figure 5 the problem is illustrated. This unnecessarily decreases the accuracy of our method and leads to more candidate locations that have to be manually checked in the end.

To avoid this overhead, we introduce a filter in form of an extremeness check: For every pair s, t classified as hole, we inspect LS' for all vertex pairs s', t' with $(s, s') \in E$ and $(t', t) \in E$, i.e. all neighbors of s and t in G. If for one of those pairs $LS'(s', t')$ is larger than $LS'(s, t)$, we prune s, t from the candidate list. The image in Figure 5 on the right shows the result of applying the extremeness procedure for the considered example.

The remaining candidate location pairs are then reported as the result of the automatic hole detection procedure.

5. Extrapolation of Missing Street Names

Another important part of the road network data in OSM are the street name tags. If a user issues a query to a route planning service, start and destination are often specified by their respective street names. This only works well if street names are complete. In OSM, though, unlabeled or only partially labeled streets are quite frequent. Often, there are multiple ways in OSM with the same street name tag but these ways are not connected (as the ways

[3] using e.g. scikit-learn (Pedregosa et al., 2011)

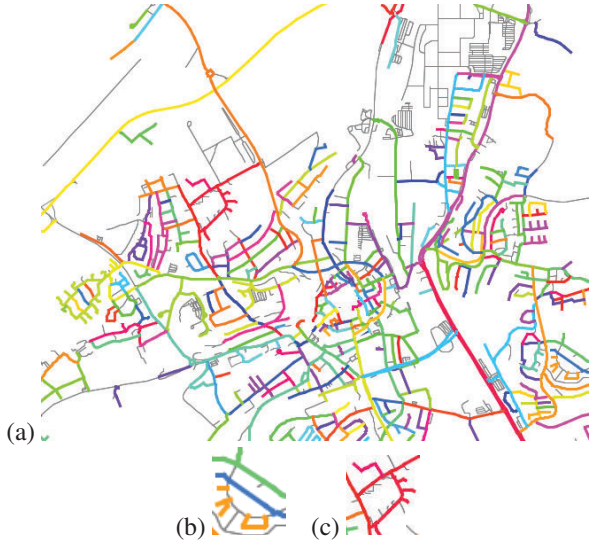


Figure 6. (a) Small map section based on OSM data. For every street name a random color was chosen and all segments with the same name share the same colour. Thin gray road segments are not tagged in OSM. In the second row, close-ups of (a) are shown: (b) illustrates a set of disconnected road segments with the same name (orange), and (c) shows small untagged side roads which are likely to have the same name as the red street.

might be contributed by different volunteers, but none of them mapped the complete course). If a user searches for a specific street (e.g. to see which shops are close-by), he expects a single entity to be returned and not multiple ways. Also if in a route planning query a user prefers certain streets or wants to avoid them, their names have to be fully contained in the data to account for that.

In the following we try to extrapolate missing street name tags from given data. We want to connect multiple ways with the same street name and extend partially tagged roads to completely tagged roads where possible. We describe semantic and topological characteristics of the road network that used as features in machine learning help to decide whether an untagged road segment can be labeled with high confidence.

5.1. Feature Extraction

We primarily rely on the assumption that all the road segments that belong to a street with one name are connected. According to our study in completely tagged areas this assumption is true for almost 99% of all streets. The visualization in Figure 6 (a) also shows typical connectivity characteristics of road segments with the same name. We refer to a connected set of edges with the same name as name component. A first feature we consider is whether an edge *is on a shortest path between two disconnected name components* with the same name (see Figure 6(b) for an example). We initially set this feature to 0 for all edges. Then we extract all name components in the network and identify street names which exhibit multiple name components in close proximity of each other (as the same street name might also occur in many villages/cities, as e.g. 'Main Street'). For every such street name we run Dijkstra computations between all pairs of nodes in different components. For all untagged edges on one of the resulting shortest paths we set the feature value to 1.

As a second feature we consider *the number of close-by name components*. So we run a Dijkstra computation from each of the two endpoints of an untagged edge until all nodes in the Dijkstra search tree are either dead-ends or are only adjacent to unrelaxed edges with $n(e) \neq \text{null}$. For all nodes in the Dijkstra search trees we compute the set of name tags of adjacent edges. Figure 6(c) shows a small example where the feature vector entry equals 1. Being connected to a single name component might be a strong indicator for the segment to belong to this component.

But as connectivity to a single name component could also mean that only one street in the area is tagged, we also consider the *shortest path distance to the closest name component* (retrievable from the two Dijkstra runs described above) and the *number of intersections* on the shortest path from the edge to the nearest name component. The higher those two values the less likely it is that the edge belongs to that name component. Finally, we again consider the *street type*. Typically, a name component consists only of edges of the same street type. Hence the feature value is computed as the absolute street type difference of the edge and the most frequent street type in the closest name component.

5.2. Training Data and Machine Learning

Again, we generated a large training data set automatically. For that purpose we first extracted completely tagged streets, i.e. we searched for name components with all nodes in that component only being adjacent to tagged streets, so no surrounding street name data is missing. Then we randomly deleted less than half of the name tags from edges on this street and also from edges inside a certain radius around the street. Afterwards, we computed the feature vector for each now untagged edge on the selected street and added the result to the training data set. Furthermore we selected completely tagged streets in the same way, but removed all of its tags and some tags on edges in the neighborhood. These are examples where extrapolation is not possible. Again, we computed the feature vectors and added them to the training data.

Like for hole classification, we deem Logistic Regression and Random Forest as suitable learning methods to infer which street segments can be extrapolated.

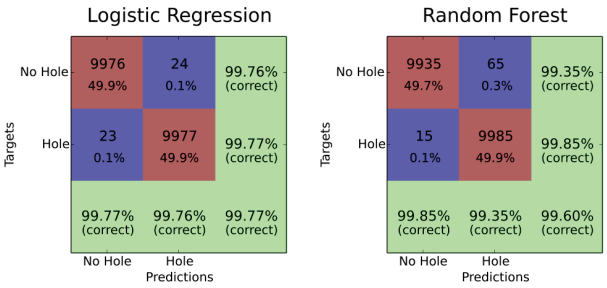


Figure 7. Accuracy of learned classifier on generated data using Logistic Regression or Random Forest.

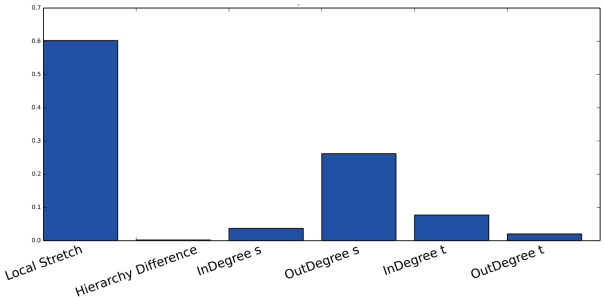


Figure 8. Feature importance when using Random Forest for classification.

6. Experimental Evaluation

We implemented the described feature extraction methods in C++. For machine learning, we used the scikit-learn package for Python (Pedregosa et al., 2011). Experiments were conducted on a single core of an Intel i5-4300U CPU with 1.90GHz and 12GB RAM. We used the OSM road network data of Germany (22.3 million nodes) and Poland (6.3 million nodes) for learning and evaluation.

6.1. Hole Classification

We created a data set containing 10,000 feature vectors of holes and 10,000 feature vectors of non-holes with the procedure described in Section 4.3.1 on the Germany data set.

For evaluating our machine learning pipeline on this data, we used 10-fold stratified cross-validation applying Logistic Regression and Random Forest to our training data. The outcomes are summarized in Figure 7. We observe that both Logistic Regression and Random Forest work remarkably well; both predict correctly in over 99% of the cases. While Logistic Regression achieves a better overall accuracy, Random Forest misclassifies slightly less holes as non-holes.

Having a closer look at the importance of the considered features (see Figure 8) for Random Forest, we see that local stretch is most important followed by the out-degree of s and the in-degree of t. We also evaluated the AuC score of the features. Local stretch achieved a score of

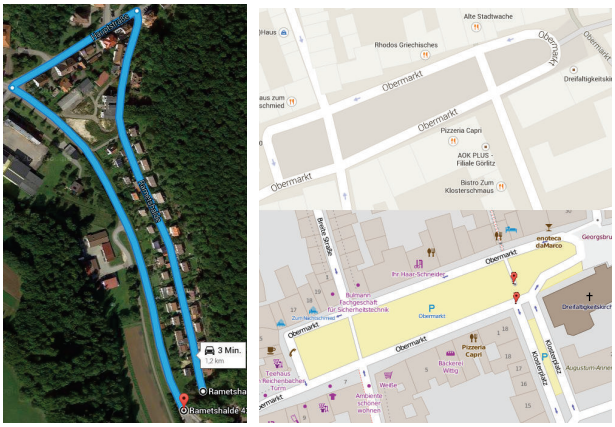


Figure 9. Left: Location pair falsely identified as hole by our classifier as observable when marked as start and endpoint in GoogleMaps. Right: Correctly identified hole indicated in the lower image by the two red markers on the OSM based map. The upper image shows the same cut-out on GoogleMaps with the two points being directly connected.

0.97 which underpins its importance for classification. The other features achieved AuC scores below 0.6. Nevertheless the combination of all features led to a 1-2$\%$ higher accuracy than considering only local stretch. The hierarchy difference resulting from the street types adjacent to s, t does not really contribute to the classification process. One reason might be the way we generated the training data. Non-holes between e.g. parallel running motorways and federal streets which exhibit rather high local stretch but also high hierarchy difference are not likely to be included in our data set. Manually selecting such examples and adding them to the training data might increase the importance of the hierarchy feature. Another problem is that there are road segments without a type which distorts the learning process.

Finally, we used our complete pipeline for real hole detection on Germany and Poland. For evaluation, we firstly selected 2000 nodes randomly in Germany and Poland. For each such node s, we extracted all nodes t within a radius of 500m (straight line distance) to form candidate pairs (s, t). For each candidate pair we computed the respective feature vector. We extracted rivers and lakes from OSM and treated them as obstacles for the LS' computation. Due to our efficient implementation of the local stretch computation, it took less than 5 minutes to process all nodes. Then we applied our classifier (Random Forest) to decide which of the candidate pairs are likely to be holes. Afterwards, we used the extremeness check to filter superfluous candidates by classifying also node pairs close to the identified holes and selecting those with the highest LS' value (therefore final node pairs not necessarily include one of the randomly selected nodes in the beginning). Table 1 shows an overview of the number of pairs resulting from each step.

	Germany	Poland
$d(s,t) < 500m$	64,194	44,332
classified	18,970	11,432
extreme	216	128
real holes	7	19

Table 1. Number of hole candidates after each step of our detection pipeline and number of correctly recognized holes in the end.

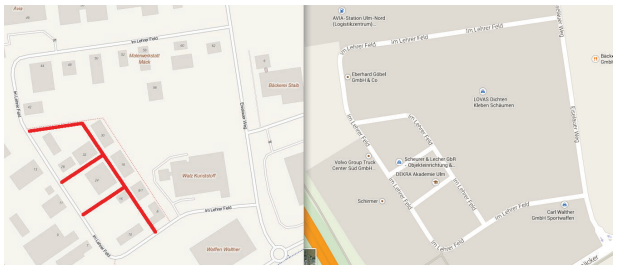


Figure 10. OSM based map (left) and GoogleMaps (right) on a cutout of Ulm, Germany. The red segments on the left indicate missing street names in the OSM. In GoogleMaps the correct name for all those segments is 'Im Lehrer Feld'. As the surrounding streets are tagged with this name in OSM and our approach classified the road segments as extrapolatable, the OSM data coverage can be increased here.

For the remaining extreme candidates we checked one-by-one if they are correctly classified by comparing to satellite images and map data from GoogleMaps. Figure 9 shows examples for a falsely identified hole and a real hole. The falsely identified hole shows that it is nearly impossible to design a perfect classifier as the very same configuration of streets might very well indicate a real hole at some other location. Main sources of misclassifying non-holes as holes were clusters of one-way streets in the middle of cities, villages close to federal streets that are not directly connected and tree-like network structures in rural areas with many dead-ends. In future work, training data could be created in a way that the classifier can better deal with such scenarios.

Nevertheless, the number of pairs that have to be manually checked is significantly smaller than the number of initially created candidate pairs. The percentage of real holes among the extreme pairs is 3% for Germany and about 15% for Poland. The difference might result from the much better overall OSM data quality in Germany or could also be seen as an indicator for already high data coverage.

6.2. Street Name Extrapolation

We extracted 10,000 feature vectors for edge segments where we assume extrapolation is possible and 10,000 for edge segments where we are sure extrapolation is impossible. Then we used Logistic Regression and Random Forest to learn feature importance/weights. In our cross-validation both approaches achieved an accuracy of 99.95%. Also in both approaches the feature expressing whether the segment connects two name components with the same name had most influence. Despite a high AuC score, the number of close-by name components made only little difference in the learned classifiers. We assume the feature indicating the shortest path distance to the closest name component shadows the number of name components, as a very small shortest path distance and a small number of close-by components are highly correlated.

For real-world validation of our learned classifier, we selected 2000 unnamed road segments in each Germany and Poland and computed the feature vectors. Our classifier declared 235 road segments in Germany extrapolatable and 164 in Poland. A visual analysis showed that most of the segments not declared extrapolatable lied in larger untagged areas. For the road segments where the classifier indicated extrapolation might be possible, we selected the closest name component for name suggestion or, if the segment connects two components with the same name, this name is the obvious choice. We relied on a comparison to GoogleMaps and BingMaps data for evaluation. Unfortunately, in surprisingly many cases (about 10%) the street segments in question were unlabeled or unclear or not even present in GoogleMaps or BingMaps. We excluded these cases from the evaluation. For the remaining cases, we achieved an accuracy of 96% in Germany and 91% in Poland (see Figure 10 for a positive example).

7. Conclusions and Future Work

We showed that machine learning is a useful tool to detect missing and possibly extrapolatable road network data in OSM. Making classified holes and nameless street segments with a good name suggestion available to the OSM community might raise attention to such locations and finally lead to a faster improvement of the OSM data quality.

There are various directions for future research. Our current methods do not work in regions where road data is completely missing. But e.g. mapped building footprints could be a strong indicator for the existence of infrastructure in an area. Considering buildings could also improve our classifiers. Including (large) buildings as obstacles for hole detection could lead to more realistic local stretch values in cities. Moreover, considering house numbers could significantly help to decide if a street segment should be tagged with a certain name. If the house numbers of tagged and untagged segments complement each other there is a good chance that they share the same name.

Finally, many other aspects of the OSM data might be suitable for extrapolation or classification using machine learning, e.g. distinguishing living and industrial areas or extrapolating missing house numbers.

References

Baum, Moritz, Dibbelt, Julian, Pajor, Thomas, and Wagner, Dorothea. Energy-optimal routes for electric vehicles. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 54–63. ACM, 2013.

Delling, Daniel and Werneck, Renato F. Faster customization of road networks. In *Experimental Algorithms, 12th International Symposium, SEA 2013, Rome, Italy, June 5-7, 2013. Proceedings*, pp. 30–42, 2013.

Efentakis, Alexandros, Brakatsoulas, Sotiris, Grivas, Nikos, and Pfoser, Dieter. Crowdsourcing turning restrictions for openstreetmap. In *EDBT/ICDT Workshops*, pp. 355–362, 2014.

Fan, Hongchao, Zipf, Alexander, Fu, Qing, and Neis, Pascal. Quality assessment for building footprints data on openstreetmap. *International Journal of Geographical Information Science*, 28(4):700–719, 2014.

Funke, Stefan. Topological hole detection in wireless sensor networks and its applications. In *Proceedings of the 2005 joint workshop on Foundations of mobile computing*, pp. 44–53. ACM, 2005.

Funke, Stefan, Nusser, André, and Storandt, Sabine. On k-path covers and their applications. In *International Conference on Very Large Databases (VLDB)*, 2014.

Girres, Jean-François and Touya, Guillaume. Quality assessment of the french openstreetmap dataset. *Transactions in GIS*, 14(4):435–459, 2010.

Haklay, Mordechai. How good is volunteered geographical information? a comparative study of openstreetmap and ordnance survey datasets. *Environment and Planning B Planning and Design*, (37):682–703, 2010.

Holone, Harald, Misund, Gunnar, and Holmstedt, Hakon. Users are doing it for themselves: Pedestrian navigation with user generated content. In *Next Generation Mobile Applications, Services and Technologies, 2007. NGMAST'07. The 2007 International Conference on*, pp. 91–99. IEEE, 2007.

Jilani, Musfira, Corcoran, Padraig, and Bertolotto, Michela. Automated quality improvement of road network in openstreetmap. In *Agile Workshop (Action and Interaction in Volunteered Geographic Information)*, 2013a.

Jilani, Musfira, Corcoran, Padraig, and Bertolotto, Michela. Multi-granular street network representation towards quality assessment of openstreetmap data. In *Proceedings of the Sixth ACM SIGSPATIAL International Workshop on Computational Transportation Science*, IWCTS '13, pp. 19:19–19:24. ACM, 2013b.

Jilani, Musfira, Corcoran, Padraig, and Bertolotto, Michela. Automated highway tag assessment of openstreetmap road networks. 2014.

Long, Ying and Liu, Xingjian. Automated identification and characterization of parcels (AICP) with openstreetmap and points of interest. *CoRR*, abs/1311.6165, 2013.

Mitchell, Joseph SB. Shortest paths among obstacles in the plane. *International Journal of Computational Geometry & Applications*, 6(03):309–332, 1996.

Mooney, Peter and Corcoran, Padraig. *Using OSM for LBS–an analysis of changes to attributes of spatial objects*. Springer, 2012.

Neis, Pascal, Goetz, Marcus, and Zipf, Alexander. Towards automatic vandalism detection in openstreetmap. *ISPRS International Journal of Geo-Information*, 1(3): 315–332, 2012.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Rahman, Kazi Mujibur, Alam, Tauhidul, and Chowdhury, Mashrur. Location based early disaster warning and evacuation system on mobile phones using openstreetmap. In *Open Systems (ICOS), 2012 IEEE Conference on*, pp. 1–6. IEEE, 2012.

Tao, Sha, Manolopoulos, Vasileios, Rodriguez, Saul, Rusu, Ana, et al. Real-time urban traffic state estimation with a-gps mobile phones as probes. *Journal of Transportation Technologies*, 2(01):22, 2012.

Vetter, Christian. Fast and exact mobile navigation with openstreetmap data. *Master's thesis, Karlsruhe Institute of Technology*, 2010.

Distributed Traffic Flow Prediction with Label Proportions: From in-Network towards High Performance Computation with MPI

Thomas Liebig THOMAS.LIEBIG@TU-DORTMUND.DE
University of Dortmund, 44221 Dortmund, Germany

Marco Stolpe MARCO.STOLPE@TU-DORTMUND.DE
University of Dortmund, 44221 Dortmund, Germany

Katharina Morik KATHARINA.MORIK@TU-DORTMUND.DE
University of Dortmund, 44221 Dortmund, Germany

Abstract

Modern traffic management should benefit from the diverse sensors, smart phones, and social networks data that offer the potential of enhanced services. In disaster scenarios, it is no longer guaranteed that a central server and reliable communication is always available. This motivates a distributed computing setting with restricted communication. Also in distributed High Performance Computing communication costs have to be reduced to the minimum and costly broadcast to all compute nodes hould be avoided. We want to learn local models with high communication efficiency. They still require the exchange of label information in a setting of supervised learning. The transmission of all labels among the nodes can be as costly as communicating all observations. Sophisticated methods are required to trade-off prediction performance against communication costs.

We hereby present an in-network algorithm based on local models that only sends label counts to neighboring nodes. Therefore the method is a novel approach that transfers no data about individual observations, but just aggregated label information. We outline its MPI implementation. And evaluate our approach on real world data in a traffic monitoring scenario. Tests reveal that in comparison to sending all labels, the algorithm is scalable.

Proceedings of the 2^{nd} International Workshop on Mining Urban Data, Lille, France, 2015.

1. Introduction

Traffic flow prediction is an important task for traffic managers. It allows performance assessment of major traffic infrastructure, like roads and junctions. Individual mobility benefits from predictions, as they provide necessary data for proactive, smart decisions on individual travel plans, e.g. predictive situation-aware trip planning by avoidance of likely traffic hazards (Niu et al., 2015; Liebig et al., 2014). Traffic flow models are based on sensor observations of current traffic gained by a mesh of (mostly static) presence sensors. While existing learning methods centralize and process measurements on a dedicated traffic management server, they have some major drawbacks, particularly in cases of disaster: The need for a reliable communication infrastructure reduces sustainability in case of natural hazards. First, the server-side collection causes high communication costs, decreasing the system's ability to process all sensor data, in time. Second, the area of traffic prediction systems is limited by the political area of homogeneous regulations for sending the data through the network. Third, increasing the network's density bares the risk of re-identification of individual persons and tracking them throughout the network. Existing systems are therefore limited by communication bandwidths, processing capabilities and political regulations.

We tackle these limitations by a distributed spatio-temporal in-network learning algorithm, where sensors compute local models and efficiently communicate label counts with their topological neighbors. Our approach sends space-time aggregated values that, by design, provide k-anonymity. Hence, our method is privacy preserving and can be applied for large-scale traffic management scenarios. Our particular focus is on the prediction of future traffic flow at junctions throughout the region of interest (e.g. a city, a state or even areas at European scale). Possible

applications comprise, for instance,

- distributed car-to-car scenarios where cars or trucks communicate at junctions the number of observed vehicles at the road to estimate traffic flow and alter their individual transportation plans based on predicted traffic conditions, or,
- large scale traffic flow prediction that processes massive local observations on a high performance computer.

Scalable in-network algorithms belong to the field of distributed data mining. Existing work mostly focuses on horizontally partitioned data. There, full observations, i.e. all features and labels, are stored on different nodes in a network. However, network states representing the current traffic flow are *vertically partitioned*. Here, only partial information about observations is stored on different nodes. Learning and prediction therefore either require the transmission of observations or labels to other nodes. Previous work (Das et al., 2011; Lee et al., 2012; Stolpe et al., 2013) has focused on sending less information about observations to a central coordinator. Here, we deal with reducing the amount of labels sent to neighboring peer nodes. Communication-efficient algorithms for vertical distributed learning are not just relevant for traffic flow prediction, but for applications as diverse as intrusion detection, monitoring production processes or smart grid management. The main contributions of our work are the following:

1. We introduce a privacy-preserving approach for the distributed learning of spatio-temporal prediction models which transfers only aggregated label information, but *no* data about individual observations.
2. A connection is drawn between the task of learning from label proportions and reducing communication costs in distributed environments, and it is evaluated on real-world data.
3. We introduce a fast search strategy for the LLP algorithm (Stolpe & Morik, 2011) and demonstrate its prediction performance in the context of traffic flow prediction.

The next section reviews related work. Section 3 details our problem setting and introduces a novel approach for the in-network training of local models. Section 4 discusses learning from aggregated label information, discusses its implementation in using message passing interface (MPI), analyses its communication cost and aspects of privacy. Evaluations of our approach can be found in Sect. 5. We finish with conclusions and outlook on future work.

2. Related Work

Many distributed data mining algorithm learn from horizontally partitioned data, whereas our data is vertically partitioned. In this context, privacy-preserving SVMs like (Yunhong et al., 2009) are not scalable, since they send quadratic kernel matrices to a central server. Distributed optimization algorithms (Bellet et al., 2014) exchange predictions for each observation per iteration, potentially sending more than the entire dataset. So does a co-regularized least squares regression in (Brefeld et al., 2006). Communication-efficient anomaly detection algorithms (Das et al., 2011; Stolpe et al., 2013) combine local and global models, but are 1-class algorithms reducing data sent about observations, not labels. In (Lee et al., 2012), local support vector machine (SVM) models are trained, but all labels are sent by a central server.

Also in traffic flow prediction, most literature describes processes on central servers. There are two major ways to model traffic: using a simulation (Raney & Nagel, 2006) or applying an imputation model, trained on previous sensor measurements. Models are required for the estimation of traffic flow at locations not being observed at all. Such imputation is not the focus of our study, but the prediction of traffic flow at sensor locations. We point the interested reader to methods of simulation (e.g. cellular automaton (Raney & Nagel, 2006)) and model-based imputation (e.g. (Liebig et al., 2012)). Most learning-based traffic flow prediction methods analyse time series, where a popular model is based on auto-regressive integrated moving average (ARIMA) (Ahmed et al., 1979). Recently, an application of a Gaussian Markov Model was proposed in (Schnitzler et al., 2014), and more advanced graphical models, namely Spatio-Temporal-Random-Fields (STRFs), were applied to traffic modeling in (Piatkowski et al., 2013).

Distributed approaches comprise an approach that applies kNN and Gaussian Process Regression (Chen et al., 2014), on-line distributed prediction of traffic flow in a large-scale road network (Wang et al., 2014), distributed traffic modeling in a MapReduce framework (Chen et al., 2013), Mapreduce parallel multivariate regression (Dai et al., 2014) and MPI (Message Passing Forum, 1994) based high performance computation based on SVM (Yang et al., 2014). Few distributed approaches combine sketches of neighbouring sensors to get probabilistic estimates of the number of vehicles co-occurring at different locations. Instead of counting and re-identifying individual vehicles, we use aggregated quantities.

The task of learning from aggregated label information was first introduced in (Kück & de Freitas, 2005). Theoretical bounds have only recently been proven in (Yu et al., 2014). (Musicant et al., 2007) propose variants of existing algorithms. The SVM optimization problem has been

adapted to the setting (Rüping, 2010; Yu et al., 2013). Mean Map (Quadrianto et al., 2009) estimates the mean operator solving a system of linear equations, while (Patrini et al., 2014) extend it with a manifold regularization, outperforming both SVMs and Mean Map on standard datasets. A modified Kernel k-Means algorithm (Chen et al., 2009) minimizes the distance to the given label proportions by matrix factorization. Recent work learns Bayesian network (Hernndez-Gonzlez et al., 2013) and generative (Fan et al., 2014) classifiers. The LLP algorithm proposed in (Stolpe & Morik, 2011) first determines clusters and then tries to label them. LLP only has linear running time, while its prediction performance competes with the approaches in (Quadrianto et al., 2009; Rüping, 2010) and(Chen et al., 2009).

3. Distributed Learning of Spatio-Temporal Local Models

Given are m distributed sensor nodes $P_1, \ldots, P_m$. Each sensor node P_i delivers an infinite series of real-valued measurements $\ldots, v_{t-1}^{(i)}, v_t^{(i)}, v_{t+1}^{(i)}, \ldots$ for different time points $\ldots, t-1, t, t+1, \ldots$. Time spans between two measurements are equidistant, given a constant sample rate. Let t denote the current time of measurement, while $t-a$ and $t+a$ are time points a steps in the past and future. Each sensor node also has a spatial location.

Many traffic flow management tasks require the prediction of traffic flow categories, that are achieved by a discretization of raw values into distinct intervals (e.g. risk level assignment or decision for emergency traffic signals).The task, given the current time point t, is therefore to predict a label y from a set $Y = \{Y_1, \ldots, Y_l\}$ of distinct categories at some arbitrary node P_i at future time point $t+r$, based on the current and previous (raw) sensor readings at all or a subset of nodes $P_1, \ldots, P_m$.

We assume that for learning, measurements and labels are somehow recorded (see below) over a fixed-length time period. For the supervised training of prediction models, each node P_i thus provides a sequence $V_i = \langle v_1^{(i)}, \ldots, v_n^{(i)} \rangle$ of measurements, $v_j^{(i)} \in \mathbb{R}$, and a sequence $L_i = \langle y_1^{(i)}, \ldots, y_n^{(i)} \rangle$ of labels $y_j^{(i)} \in Y$.

Distributed Learning of Local Models

Instead of centralizing all data, we propose that each P_i records and stores its own measurements and labels. For predicting future traffic flow categories at node P_i, we restrict learning to P_i itself and c topological neighboring nodes around P_i. For instance, to learn and predict the future type of traffic flow at some street junction, considered are only measurements and labels recorded at the junction itself and at c junctions closest to it.

Before training, each P_i preprocesses measurements V_i as follows. A window of size p is slided over the series V_i with step size 1, storing all thereby created windows $x_t^{(i)} = \{v_{t-p+1}^{(i)}, \ldots, v_t^{(i)}\}$, $t = p, \ldots, n$ as rows in a dataset D_i. Let $N^{(i)} = \{n_1^{(i)}, \ldots, n_c^{(i)}\}$ be the set of indices for the c neighboring nodes around P_i. Based on the datasets $D_i, D_{n_1^{(i)}}, \ldots, D_{n_c^{(i)}}$ and labels L_i, we want to learn a *local* function (model) $f^{(i)}$ that, given windows $x_t^{(i)}, x_t^{n_1^{(i)}}, \ldots, x_t^{n_c^{(i)}}$ of sensor readings from node P_i and its neighbors, predicts the label $y_{t+r}^{(i)}$ at node P_i with horizon r correctly.

Interpreting windows $x_t^{(i)}, x_t^{n_1^{(i)}}, \ldots, x_t^{n_c^{(i)}}$ as features of a single observation x that should be classified, the data is *vertically partitioned*, since each neighboring node of P_i only stores partial information about x, i.e. a subset of features.

An obvious choice for the training of $f^{(i)}$ at P_i is to ask for the recorded measurements at each neighboring node, concatenate their columns at P_i and join the labels stored at P_i to the new dataset. The approach is more scalable than centralizing all data, since the number c of neighbors is fixed, avoiding the bottleneck problem of limited bandwidth. However, each node still needs to transmit *all* measurements to each of its neighbors, consuming at least as much energy per node as sending all data to a single server.

Therefore, we propose to send only label information from node P_i to its neighbors and to train models $f_0^{(i)}$ at node P_i and $f_{n_1^{(i)}}^{(i)}, \ldots, f_{n_c^{(i)}}^{(i)}$ at its neighbors. As model $f^{(i)}$ at node P_i, we propose a majority vote over predictions from itself and its neighboring nodes. All models are *local*, since they only consider measurements and labels of a fixed number of close topological neighboring nodes around P_i. Moreover, the approach works fully *in-network* without a central coordinator, since each node only communicates with its neighboring peer nodes. As learners at each node, one may consider supervised learners, like kNN, Decision Trees or SVMs. Considering the limited computational resources of sensor nodes, however, our evaluation in Sect. 5 is solely based on kNN.

Since the number of bits to encode all labels is often less than an encoding of all measurements, communication is saved by sending labels *from* P_i instead of measurements *to* P_i. However, supervised learning still requires individual labels for all observations. The question is if communication can be reduced even further, by sending fewer labels or aggregated label information to each neighboring node.

Semi-supervised (Chapelle et al., 2006) and active learning (Balcan et al., 2010) show that training on fewer labels may achieve a similar performance as training on all labels.

However, such methods do not preserve the privacy of the data, since they need individual labels of observations (see Sect. 4). Instead, we propose to send only *aggregated* label information, i.e. label counts, to neighboring nodes for learning.

4. Aggregation of Label Information

Before sending label information to each of its neighboring nodes, P_i divides its time-related sequence L_i of labels into consecutive batches $C_1^{(i)}, \ldots, C_h^{(i)}$ of a fixed size b (see Fig. **??**). It respects the prediction horizon r, such that each $C_j^{(i)}$ consists of labels from time point $t+(j-1)b+r$ to $t+jb+r$ and align correctly with time points of observations (i.e. windows of measurements) at other nodes. Let n be from here on the size of datasets D_i, i.e. the number of windows stored. Then, h is $\lceil n/bs \rceil$. For each batch j, labels $y \in Y$ are aggregated by counting how often they occur, and stored in a $h \times l$ matrix of label counts $Q^{(i)} = (q_{jd}^{(i)})$, where $q_{jd}^{(i)} = |\{y \in C_j | y = Y_d\}|$.

Let $P_{n_e^{(i)}}$ be a neighboring node receiving label counts from P_i. $P_{n_e^{(i)}}$ transforms $Q^{(i)}$ into a label proportion matrix $\Pi^{(i)} = (\pi_{jd}^{(i)}) = q_{jd}^{(i)}/b$, i.e. the counts of labels are divided by batch size b. Since every node knows b and r, $P_{n_e^{(i)}}$ can partition its own windows $x_1^{n_e^{(i)}}, \ldots, x_n^{n_e^{(i)}}$ of measurements into batches $B_1^{n_e^{(i)}}, \ldots, B_h^{n_e^{(i)}}$. Since the sender respects r, the time spans used for aggregating the labels align correctly with the windows of measurements stored at $P_{n_e^{(i)}}$.

The learning task at node $P_{n_e^{(i)}}$ now consists of learning a model $f_{n_e^{(i)}}^{(i)}$, only based on its batches of (unlabeled) measurements and the label information from node P_i, stored in the label proportion matrix $\Pi^{(i)}$, such that the expected prediction error over individual observations is minimized. This task is also known as *learning from label proportions*.

Several methods have been developed to solve the task (see Sect. 2). Considering the limited computational resources of sensor nodes, the LLP algorithm (Stolpe & Morik, 2011) looked most promising for our evaluation in Sect. 5, since LLP has a linear running time and its centroid model a small memory footprint. Moreover, it can handle multiclass classification problems as they arise in traffic monitoring. However, we found that it still needs to be improved for scalability issues and performance. The next section describes LLP shortly, while Sect. 4 introduces a new local search method.

The LLP Algorithm

LLP learns from label proportions by first clustering all observations and then assigning labels to each cluster. The task of cluster analysis consists of partitioning a set of observations into a set $\mathcal{C}$ of k disjunct groups (clusters) $C_1, \ldots, C_k$, such that the similarity of observations in each cluster is minimized. LLP relies on the idea that observations having the same class also share similar features, i.e. that clusters somehow correspond to classes. LLP allows for several clusters per class and assumes that the majority of elements of a cluster belongs to the same class. Once given a clustering the only remaining problem is to assign correct labels to each cluster.

More formally, let $\mu : X \to \mathcal{C}$ be a mapping that assigns an arbitrary observation $x \in X$ to a cluster $C \in \mathcal{C}$. For centroids $c_1, \ldots, c_k$ found with k-Means, $\mu(x)$ would be defined as $\mu(x) = \text{argmin}_{C_k \in \mathcal{C}} \, ||x - c_k||^2$.

Further, let $\ell : \mathcal{C} \to Y$ be a mapping which assigns a label $\lambda \in Y$ to each cluster $C \in \mathcal{C}$. For ease of notation, let f denote model $f_{n_e^{(i)}}^{(i)}$ to be learned at node $P_{n_e^{(i)}}$, B_i denote the batch $B_i^{n_e^{(i)}}$ and Π denote matrix $\Pi^{(i)}$. f is the composition of mappings ℓ and μ, i.e. $f = \ell \circ \mu$.

With prediction model f, entries γ_{jd} of a model-based proportion matrix $\Gamma_f = (\gamma_{jd})$ can be calculated as

$$\gamma_{jd} = \frac{1}{|B_j|} \sum_{x \in B_j} I(f(x), Y_d), \quad I = \begin{cases} 1 & : \quad f(x) = Y_d \\ 0 & : \quad f(x) \neq Y_d \end{cases} . \tag{1}$$

The LLP algorithm now minimizes the mean squared error

$$\text{MSE}(\Pi, \Gamma_f) = \frac{1}{hl} \sum_{j=1}^{h} \sum_{d=1}^{l} (\pi_{jd} - \gamma_{jd})^2 , \tag{2}$$

between the given label proportion matrix Π and the model-based proportion matrix Γ_f by trying different label mappings ℓ.

A Local Search Strategy with Multistarts

The LLP algorithm as introduced in (Stolpe & Morik, 2011) can work with different cluster algorithms and labeling strategies. LLP with an exhaustive labeling strategy, called LLP_{exh} in the following, tries all possible labelings of the clusters. We found it too time-consuming for the evaluations done in Sect. 5. The greedy strategy proposed in (Stolpe & Morik, 2011) didn't achieve sufficient accuracies for traffic prediction. Hence, a better search strategy is demanded.

We propose a local search that is started multiple times with different random combinations of labels. LLP with this search strategy will be called LLP_{lsm} in the following. The local search greedily improves on the current labeling of clusters by trying all possible labels at each component of a labeling vector $\boldsymbol{\lambda}$. Fitness measures how well the model-

based label proportion matrix Γ_f, as calculated from the current labeling, matches the given label proportions. If the fitness improves, the search starts from the first component of the labeling vector $\boldsymbol{\lambda}$, again. Otherwise, it resets the label at the current position $kpos$ to the label of the best (local) solution found so far. Returned is the best labeling found over all starts of the different greedy searches.

In each iteration, the greedy search runs until no further improvement is possible. Moreover, at each step of the algorithm, the fitness either improves or is staying the same (which is a stopping criterion). Therefore, each search finds a local minimum. Since the number of searches is finite, the returned labeling vector is also locally minimal. In comparison to LLP_{exh}, it cannot be guaranteed that a globally optimal solution is found. However, with regard to the prediction results presented in Sect. 5, we found that a local search performed sufficient enough, despite a much lower running time.

LLP as introduced in (Stolpe & Morik, 2011) combines the MSE with two other error measures. However, we found that the use of these additional measures decreases the accuracy in the traffic monitoring scenario. Hence, all experiments in Sect. 5 are based on the MSE, only. Similarly, we abstain from the evolutionary feature weighting presented in (Stolpe & Morik, 2011), since it would heavily increase the algorithm's running time.

MPI IMPLEMENTATION

We explicitly focus on the implementation with the Message Passing Interface (MPI) (Message Passing Forum, 1994) as message passing in the top super computers in the top500 list[1] base on the MPI standard. MPI implements the single program multiple data paradigm (Darema, 2001) whereas every node of a distributed system executes the same program but uses different data. To coordinate this architecture a MPI program consists of 1 master node and multiple slave nodes that perform computations, the results are collected at the master. The art of MPI programming is to divide the problem into multiple tasks that are transferred to the slaves and processed thereby. Usually, the number of tasks exceeds the number of slave nodes and the distribution of the tasks has to be organized by the master.

The work in (Nupairoj & Ni, 1994) analyses the performance of such MPI systems and reveals that communication is major bottleneck in MPI programs. A succeeding publication (Piernas et al., 1997) provides empirical estimates for computation of communication costs depending on message lengths. Based on this publications two major conclusions ban be made: (1) The shorter the messages the lower the communication cost, and(2) broadcast messages should be avoided. Our algorithm respects both findings and therefore seems suitable for an MPI implementation. For ease of development we decided for the Cran-R package Rmpi (Yu, 2002) which provides basic MPI functionalities in Cran-R, thus matrix operators can be applied to the data. Our program comprises the following generic steps:

1. Load Rmpi, and spawn slaves
2. Definition of the functions for the master
3. Definition of necessary functions for the LLP algorithm at the slave
4. Initialization of the data
5. Send required data and functions to the slaves
6. Tell slaves to execute their function
7. Communicate with the slaves to perform computation
8. Collect the results
9. Close slaves and quit

For the learning from label proportions, our implementation presumes a shared network file system and initially processes the data at the master such that the sliding windows of the measurements are stored as Robjects on the file system. Every task gets its pointer to the corresponding slice of data and the label proportions of neighbouring nodes. The LLP algorithm is executed in every task and the trained models (cluster centers and their labels) are again stored physically for later re-use. This also allows the deployment of the parallel learned models in embedded devices or the future application of the label proportion models in high performance computation settings. Next subsection analyses the communication cost of LLP in comparison to kNN algorithm.

ANALYSIS OF COMMUNICATION COSTS

Each node P_i transmits a matrix Q to each of its neighboring nodes, consisting of counts for each label $Y_d \in Y$ and batch. Such counts may be assumed to be integers. The maximum value of each integer is b, which means we need to reserve at most $\lceil \log_2 b \rceil$ bits for each label. The number of batches, given n observations, is $\lceil n/b \rceil$. The total number of bits z_{AGG} for encoding matrix Q is therefore

$$z_{\text{AGG}} = \left\lceil \frac{n}{b} \right\rceil \lceil \log_2 b \rceil |Y| \quad . \tag{3}$$

In comparison, the number of bits z_{ALL} required to encode *all* labels of n observations, for $|Y|$ different labels, is at most

$$z_{\text{ALL}} = n \lceil \log_2 |Y| \rceil \quad . \tag{4}$$

[1] http://www.top500.org/lists/2014/11/, last accessed May, 1st

The total costs are then either z_{AGG} or z_{ALL}, multiplied by the number of nodes m. Here we assume that label information is broadcast to each neighboring node, which is not unrealistic for sensors in topologically close regions. All payloads reported in Sect. 5 base on this assumption.

Analysis of Privacy

The *vulnerable data* are the original sensor readings. These traffic flow measurements bare the risk of re-identification of individual vehicles. For example in a dense sensor network with sparse observations of vehicles, their occurrence may be tracked throughout the network. As mobility often is a regular behaviour and contains patterns this risk is even higher. In this section we show that our LLP_{lsm}-based algorithm transforms the data such that re-identification risk is at most $1/s$.

In our distributed setting, *adversaries* of a particular sensor node are malicious sensors that could use received measurements of neighboring sensors for deduction of individual mobility traces. The following *attack model* is possible: The adversary analyses differences among neighboring sensor readings and deduces individual movement. If the difference among two neighboring sensor readings is zero and both traffic flow counts are w, it is (depending on network topology) likely that w vehicles moved between the two sensors. In case of three neighboring sensors P_a, P_b, P_c their measurements v_a, v_b, v_c can be combined as follows: If $v_a - v_b = w = v_c$ it may be deduced that on the way from P_a to P_b w vehicles turned to P_c, in case $v_a - v_b = -w = -v_c$ w vehicles originated from the location P_a.

With our new LLP_{lsm}-based approach we process discretized traffic flow values and just communicate counts of these value ranges. We denote the minimal (nonzero) interval width by s. Thus, measurements may not be distinguished up to a granularity of s vehicles and w is bounded by s, $w \geq s$. In turn, the risk of re-identification with the hereby described attack model is at most $1/s$. Our approach therefore provides s-anonymity by design. The aggregation of label information reduces the remaining risk for disclosure of neighboring labels at a malicious sensor node. The solely transmission of label counts prevents doubtless reconstruction of the labels (Yu et al., 2014).

5. Experiments

We perform tests of the method on data of the city of Dublin. The Sydney Coordinated Adaptive Traffic System (SCATS) provides information on vehicular traffic at over 750 fixed sensor locations as spatio-temporal time series (McCann, 2014). The data we use[2] is a snapshot from 01/01/2013 till 14/05/2013, consisting of tuples (t, u, w), where u is the location of the observation and consists of an index for the junction, the arm and the lane number at which the sensor is located at. The metric w contains the aggregated vehicle count at sensor location since last measurement. The time stamp t denotes the recording time.

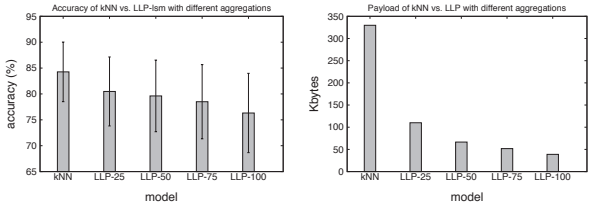


Figure 1. Trade-off between accuracy and payload sent for kNN and LLP_{lsm}

Local models are trained for each of the 296 sensor nodes and their nearest topological neighbors. As supervised base-line learner that receives all labels, we use kNN with $k = 15$. For learning from aggregated label counts, we cluster the observations at each node with k-Means ($k = 15$, 50 different random starting points, 500 iterations at maximum) and label the clusters with LLP_{lsm} (with 150 starts of the local greedy search) at each node for different batch sizes $b = 25, 50, 75$ and 100. The accuracy of each method is assessed by a 10-fold cross validation, i.e. all models are trained and evaluated for different hold-out sets 10 times. In total $296 \times 7 \times 10 = 20,720$ models for kNN need to be evaluated and $296 \times 7 \times 10 \times 4 = 82,880$ models trained and evaluated for LLP_{lsm}. The evaluation has been done offline in parallel on different machines (about 36 CPU cores).

Figure 1 shows the trade-off between accuracy and payload sent for kNN and LLP_{lsm} trained on differently sized batches of aggregated labels. Besides the average accuracy over all 10-fold cross-validations at each node, the bars in Fig. 1 (left) also depict the standard deviation of accuracy over all nodes.

In general, LLP_{lsm} performs slightly worse than kNN. Nevertheless, there are still many junctions for which the traffic flow is predicted quite well with LLP_{lsm}. Some locations have bad performance with both methods, a comparison to the map reveals that these are locations of parking areas e.g. inner-city parking houses and recreational areas where many vehicles stay for a long period of time.

6. Conclusions

The task of scalable traffic flow prediction involves a trade-off between the accuracy of models and the amount of communication between networked nodes. Especially in high

[2]Data is publicly available at http://dublinked.ie .

performance computation and embedded devices communication is costly.

In this paper we presented a novel approach for local models that trades-off communication costs to prediction accuracy which is suitable for in-network deployment and cluster computations.

Future work will focus on examining more sophisticated aggregation strategies for labels. We will study how to include dynamic distributed traffic flow prediction in state-of-the-art (multi-modal) route planning methods proposed by (Bast et al., 2014).

Acknowledgements

This research has received funding from the European Union's Seventh Framework Programme under grant agreement number FP7-318225, INSIGHT. Additionally, this work has been supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876, project B3. We thank Jan Czogalla for data preprocessing.

References

Ahmed, M.S., Cook, A.R., of Oklahoma. School of Civil Engineering, University, and Science, Environmental. *Analysis of Freeway Traffic Time Series Data Using Box and Jenkins Techniques*. 1979.

Balcan, M.-F., Hanneke, S., and Vaughan, J. W. The true sample complexity of active learning. *Machine Learning*, 80(2–3):111–139, 2010.

Bast, Hannah, Delling, Daniel, Goldberg, Andrew, Müller-Hannemann, Matthias, Pajor, Thomas, Sanders, Peter, Wagner, Dorothea, and Werneck, Renato. Route planning in transportation networks. Technical Report MSR-TR-2014-4, January 2014.

Bellet, A., Liang, Y., Garakani, A. B., Balcan, M.-F., and Sha, F. Distributed Frank-Wolfe algorithm: A unified framework for communication-efficient sparse learning. *CoRR*, abs/1404.2644, 2014.

Brefeld, U., Gärtner, T., Scheffer, T., and Wrobel, S. Efficient co-regularised least squares regression. In *Proc. of the 23rd Int. Conf. on Machine Learning (ICML)*, pp. 137–144, New York, NY, USA, 2006. ACM.

Chapelle, O., Schölkopf, B., and Zien, A. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

Chen, Cheng, Liu, Zhong, Lin, Wei-Hua, Li, Shuangshuang, and Wang, Kai. Distributed modeling in a mapreduce framework for data-driven traffic flow forecasting. *Intelligent Transportation Systems, IEEE Transactions on*, 14(1):22–33, 2013.

Chen, S., Liu, B., Qian, M., and Zhang, C. Kernel k-Means based framework for aggregate outputs classification. In *Proc. of the Int. Conf. on Data Mining Workshops (ICDMW)*, pp. 356–361, 2009.

Chen, Xing-Yu, Pao, Hsing-Kuo, and Lee, Yuh-Jye. Efficient traffic speed forecasting based on massive heterogenous historical data. In *Big Data (Big Data), 2014 IEEE International Conference on*, pp. 10–17, Oct 2014. doi: 10.1109/BigData.2014.7004425.

Dai, Liang, Qin, Wen, Xu, Hongke, Chen, Ting, and Qian, Chao. Urban traffic flow prediction: A mapreduce based parallel multivariate linear regression approach. In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pp. 2823–2827, Oct 2014. doi: 10.1109/ITSC.2014.6958142.

Darema, Frederica. The spmd model: Past, present and future. In *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, pp. 1–1. Springer, 2001.

Das, K., Bhaduri, K., and Votava, P. Distributed anomaly detection using 1-class SVM for vertically partitioned data. *Stat. Anal. Data Min.*, 4(4):393–406, 2011.

Fan, K., Zhang, H., Yan, S., Wang, L., Zhang, W., and Feng, J. Learning a generative classifier from label proportions. *Neurocomput.*, 139:47–55, 9 2014.

Hernndez-Gonzlez, J., Inza, I., and Lozano, J. A. Learning bayesian network classifiers from label proportions. *Pattern Recognition*, 46(12):3425–3440, 2013.

Kück, H. and de Freitas, N. Learning to classify individuals based on group statistics. In *Proc. of the 21th UAI*, pp. 332–339, 2005.

Lee, S., Stolpe, M., and Morik, K. Separable approximate optimization of support vector machines for distributed sensing. In *Machine Learning and Knowledge Discovery in Databases*, volume 7524 of *LNCS*, pp. 387–402, Berlin, Heidelberg, 2012. Springer-Verlag.

Liebig, Thomas, Xu, Zhao, May, Michael, and Wrobel, Stefan. Pedestrian quantity estimation with trajectory patterns. In *Machine Learning and Knowledge Discovery in Databases*, pp. 629–643. Springer Berlin Heidelberg, 2012.

Liebig, Thomas, Piatkowski, Nico, Bockermann, Christian, and Morik, Katharina. Predictive trip planning - smart routing in smart cities. In *Proceedings of the Workshops of the EDBT/ICDT 2014 Joint Conference (EDBT/ICDT 2014), Athens, Greece, March 28, 2014*, volume 1133, pp. 331–338. CEUR-WS.org, 2014.

McCann, Barry. A review of scats operation and deployment in dublin. In *Proceedings of the 19th JCT Traffic Signal Symposium & Exhibition*. JCT Consulting Ltd, 2014.

Message Passing Forum. Mpi: A message-passing interface standard. Technical report, Knoxville, TN, USA, 1994.

Musicant, D. R., Christensen, J. M., and Olson, J. F. Supervised learning by training on aggregate outputs. In *7th Int. Conf. on Data Mining (ICDM)*, pp. 252–261, 10 2007.

Niu, Xiaoguang, Zhu, Ying, Cao, Qingqing, Zhang, Xining, Xie, Wei, and Zheng, Kun. An online-traffic-prediction based route finding mechanism for smart city. *International Journal of Distributed Sensor Networks*, 501:970256, 2015.

Nupairoj, Natawut and Ni, Lionel M. Performance evaluation of some mpi implementations on workstation clusters. In *Scalable Parallel Libraries Conference, 1994., Proceedings of the 1994*, pp. 98–105. IEEE, 1994.

Patrini, G., Nock, R., Caetano, T., and Rivera, P. (almost) no label no cry. In *Advances in Neural Information Processing Systems 27*, pp. 190–198. Curran Associates, Inc., 2014.

Piatkowski, Nico, Lee, Sangkyun, and Morik, Katharina. Spatio-temporal random fields: compressible representation and distributed estimation. *Machine Learning*, 93 (1):115–139, 2013. ISSN 0885-6125.

Piernas, Juan, Flores, A, and García, José M. Analyzing the performance of mpi in a cluster of workstations based on fast ethernet. In *Recent advances in Parallel Virtual Machine and Message Passing Interface*, pp. 17–24. Springer, 1997.

Quadrianto, N., Smola, A. J., Caetano, T. S., and Le, Q. V. Estimating labels from label proportions. *J. Mach. Learn. Res.*, 10:2349–2374, 12 2009.

Raney, B. and Nagel, K. An improved framework for large-scale multi-agent simulations of travel behavior. *Towards better performing European Transportation Systems*, pp. 305–347, 2006.

Rüping, S. SVM classifier estimation from group probabilities. In *Proc. of the 27th Int. Conf. on Machine Learning (ICML)*, pp. 911–918, 2010.

Schnitzler, François, Liebig, Thomas, Mannor, Shie, and Morik, Katharina. Combining a gauss-markov model and gaussian process for traffic prediction in dublin city center. In *Proceedings of the Workshops of the EDBT/ICDT 2014 Joint Conference (EDBT/ICDT 2014), Athens, Greece, March 28, 2014*, volume 1133, pp. 373–374. CEUR-WS.org, 2014.

Stolpe, M., Bhaduri, K., Das, K., and Morik, K. Anomaly detection in vertically partitioned data by distributed core vector machines. In *European Conf. on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD)*, pp. 321–336. Springer, 2013.

Stolpe, Marco and Morik, Katharina. Learning from label proportions by optimizing cluster model selection. In *Proceedings of the 2011 European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part III*, ECML PKDD'11, pp. 349–364, Berlin, Heidelberg, 2011. Springer-Verlag.

Wang, Yubin, van Schuppen, Jan H., and Vrancken, Jos. On-line distributed prediction of traffic flow in a large-scale road network. *Simulation Modelling Practice and Theory*, 47(0):276 – 303, 2014. ISSN 1569-190X. doi: http://dx.doi.org/10.1016/j.simpat.2014.06.011.

Yang, Zhaosheng, Mei, Duo, Yang, Qingfang, Zhou, Huxing, and Li, Xiaowen. Traffic flow prediction model for large-scale road network based on cloud computing. *Mathematical Problems in Engineering*, 2014, 2014.

Yu, F. X., Kumar, S., Jebara, T., and Chang, Shih-Fu. On learning with label proportions. *CoRR*, abs/1402.5902, 2014.

Yu, F. X. Yu, Liu, D., Kumar, S., Jebara, T., and Chang, S. $\propto$SVM for learning with label proportions. In *Proc. of the 30th Int. Conf. on Machine Learning (ICML)*, pp. 504–512, 2013.

Yu, Hao. Rmpi: Parallel statistical computing in r. *R News*, 2(2):10–14, 2002. URL http://cran.r-project.org/doc/Rnews/Rnews_2002-2.pdf.

Yunhong, H., Liang, F., and Guoping, H. Privacy-preserving SVM classification on vertically partitioned data without secure multi-party computation. In *5th Int. Conf. on Natural Computation (ICNC)*, volume 1, pp. 543–546, 8 2009.

Event-Based Clustering for Reducing Labeling Costs of Incident-Related Microposts

Axel Schulz SCHULZ.AXEL@GMX.NET
DB Mobility Logistics AG, Germany and Telecooperation Lab, Technische Universität Darmstadt, Germany

Petar Ristoski PETAR.RISTOSKI@INFORMATIK.UNI-MANNHEIM.DE
Data and Web Science Group, University of Mannheim, Germany

Johannes Fürnkranz JUFFI@KE.INFORMATIK.TU-DARMSTADT.DE
Knowledge Engineering Group, Technische Universität Darmstadt, Germany

Frederik Janssen JANSSEN@KE.TU-DARMSTADT.DE
Knowledge Engineering Group, Technische Universität Darmstadt, Germany

Abstract

Automatically identifying the event type of event-related information in the sheer amount of social media data makes machine learning inevitable. However, this is highly dependent on (1) the number of correctly labeled instances and (2) labeling costs. Active learning has been proposed to reduce the number of instances to label. Though, current approaches focus on the thematic dimension, i.e., the event type, for selecting instances to label; other metadata such as spatial and temporal information that is helpful for achieving a more fine-grained clustering is currently not taken into account. Also, labeling quality is always assumed to be perfect as currently no qualitative information is present for manual event type labeling.

In this paper, we present a novel event-based clustering strategy that makes use of temporal, spatial, and thematic metadata to determine instances to label. Furthermore, we also inspect the quality of the manual labeling in a crowdsourcing study by comparing experts and non-experts. An evaluation on incident-related tweets shows that (i) labels provided by crowdsourcing are of acceptable quality and (ii) our selection strategy for active learning outperforms current state-of-the-art approaches even with few labeled instances.

Proceedings of the 2^{nd} International Workshop on Mining Urban Data, Lille, France, 2015.

1. Introduction

Detecting event-related information in microposts has shown its value for a variety of domains. Especially in emergency management, different situational information is present that could contribute to understand the situation at hand (Schulz, 2014). However, solving the actual problem of classifying the incident type in this domain requires labeled data. One of the main problems with microposts is acquiring ground truth for utilizing supervised learning. Thus, we deal with two major issues: (1) The costs for labeling a single instance, and (2) the number of instances to label.

On the one hand, to actually build a classifier that is able to accurately predict the type of the incident mentioned in a tweet, usually experts are deployed for labeling as they have enough domain knowledge to create ground truth. However, as often several hundreds of examples have to be labeled until the classifier is able to reach sufficient quality, relying on experts for labeling is not always possible and it is costly. In contrast, labels can also be derived from non-experts, i.e., by making use of crowdsourcing. Given that the labels obtained in this way are of sufficient quality, the costs for such a process would be acceptable as crowdsourcing is rather cheap. But up to now there is no information about labeling quality for incident-related tweets. Hence, first we proceeded by comparing the labeling quality of experts and non-experts.

On the other hand, the number of instances to label has to be kept as low as possible. Due to the huge number of tweets, labeling all instances is not possible as even with cheap labeling the costs would explode. Keeping the number of instances to label low while maintaining accurate

classifiers is a typical *active learning* (Settles, 2012) problem. Here, labeling costs are reduced by iteratively (1) selecting small subsets of instances to query for labels and (2) re-training a classifier with the newly labeled data. Thus, in general, but also specifically for classifying microposts, there are two issues to solve, namely selecting a good initial training set and the right instances in each iteration.

For selecting appropriate instances, several selection strategies have been proposed based on the two criteria, *informativeness* and *representativeness* (Huang et al., 2010). *Informativeness* measures the usefulness of an instance to reduce the uncertainty of the model, whereas *representativeness* measures how good an instance represents the overall input of unlabeled data. The latter usually is solved by employing clustering approaches where then from each cluster the representative instances are drawn. Indeed, for event-type classification the number of clusters to build is not known in advance, as it is unknown how often an event occurred. Hence, most often it is set to the number of distinct event types, which obviously is not appropriate. For instance, one event might be a tiny fire in a waste bin whereas another is a huge fire in a factory; though microposts for both events need to be classified with the "fire" event type, state-of-the-art approaches would not distinguish these two events and thus could not yield an optimal selection of instances to label. For better distinguishing events, a straightforward approach is to characterize an event not only by its type, but also by spatial and temporal information. Proceeding this way, the two example events are inherently assigned to different clusters and hence instances to be labeled are drawn from both of them.

Consequently, we contribute an event-based clustering approach that also leverages the temporal and spatial dimension of tweets to allow a more fine-grained clustering. Due to smaller clusters the selection of appropriate instances is easier because one can assume that even with a bad sampling the selected instances will still be of high quality. The evaluation on incident-related tweets shows that this enhanced clustering indeed improves the selection compared to state-of-the-art approaches. It is also shown that our approach has a good performance even when only few examples are labeled.

In summary, the contributions of this paper are: (1) A study comparing the labeling quality of experts and non-experts showing no significant difference of error rates. (2) A novel event-based clustering approach that makes use of spatial, temporal, and thematic information present in microposts. The clustering benefits strongly from these additional dimensions. (3) A comparison of our approach using different number of annotators and different levels of noise. Even with a classifier that was not explicitly build to be robust, noise does not hinder the classifier much.

We begin with summarizing related approaches. Next, we show how the ground truth data was developed. Then we summarize the results of the study on crowdsourced labels (Section 4) followed by a description of the event-based clustering for active learning. After, the results are shown and discussed (Section 6) and the paper is concluded.

2. Related Work

Although active learning has been studied extensively for text classification (Hoi et al., 2006; Tong & Koller, 2002), it was used for tweets only by a few previous works. (Thongsuk et al., 2010) presented a technique for classifying tweets into three business types. They showed that using active learning outperforms simple supervised learning approaches in terms of labeling costs.

(Hu et al., 2013) presented the ActNeT approach, which takes the relations between tweets into account for identifying representative as well as informative instances. Based on a social network, the topology is used to detect representative instances using the PageRank algorithm. Informative instances are chosen using an entropy-based uncertainty sampling. However, as building the social network is time consuming and not always possible due to API restrictions, their approach is not applicable for our problem. Also, they do not use event-related metadata.

Several selection strategies were presented that propose to select informative as well as representative instances. (Tang et al., 2002) used k-means clustering and proposed to select the most uncertain instance for each cluster. Information density was then used to weight instances. (Shen et al., 2004) applied k-means clustering and uncertainty sampling and used the information density calculated within a cluster. (Donmez et al., 2007) combined uncertainty sampling and k-Medoid to identify representative as well as informative instances and showed that this combination is indeed beneficial.

The approach of (Zhu et al., 2008) is the most advanced related approach when it comes to combining representativeness and informativeness, thus, we used it as a foundation for our technique. The authors employed clustering for the initial selection. Uncertainty sampling is combined with estimating a density for each iteration. Unlike their work, we apply our event-based clustering also for the iterations. (Huang et al., 2010) followed a similar approach. Instances are selected based on clustering and on confidence in predicting a class label as informativeness measure. Though their approach is quite promising, the authors stated that it is restricted to binary classification, whereas we are able to classify multiple classes.

Taking labeling quality into account is still open to research. Up to now, there is no study of labeling quality

of event-related tweets, but only studies on structured texts such as the work of (Hsueh et al., 2009). Since 2008, the active learning community also tackled the problem of different reliabilities of oracles (Donmez & Carbonell, 2008; Zhao et al., 2011; Wallace et al., 2011). These approaches have been proposed to take labeling uncertainty into account and show that repeated re-labeling of wrongly labeled tweets could improve label quality and model quality. Nevertheless, most often synthetic error rates have been assumed.

To sum up, some works tried to combine informativeness and representative for selecting instances and showed promising results. Nevertheless, none of these approaches has been evaluated on microposts or has taken event-related metadata into account. Also, no information about real-world error rates is present or was used in active learning.

3. Developing Ground Truth Data

In this section, we present our dataset used for our evaluation. We focus on incident-related tweets as a specific type of event-related data. We differentiate between three incident types in order to classify microposts. These have been chosen because we identified them as the most common incident types in the Seattle Fire Calls dataset[1], which is a frequently updated source for official incident information. We also add one neutral class, thus, our final classes are: *car crash*, *fire*, *shooting*, and *no incident*.

As there are no publicly available labeled datasets for event-related microposts, we needed to create our own high-quality ground truth data. For this, we collected English microposts using the Twitter Search API. For the collection, we used a 15km radius around the city centers of Seattle, WA and Memphis, TN. We focused on only two cities, as for our analysis we were interested in a large stream of tweets for a specific time period of certain areas instead of a world-wide scattered sample. This gave us a set of 7.5M microposts from Nov. 19th, 2012 until Feb. 7th, 2013. Although the datasets have been collected in different time periods, we do not expect any difference in the way people post about incidents.

As this initial set was used for conducting our experiments, we had to further reduce the size of the datasets following our approach as described in (Schulz et al., 2013b). The resulting 2,000 tweets were manually labeled by four domain-experts using an online survey. To assign the final coding, at least three coders had to agree on a label. Instances without an agreement were further examined and relabeled during a group discussion. The final dataset consists of 328 fire, 309 crash, 334 shooting, and 1029 not incident related tweets.[2] For our evaluation, we used 1,200 tweets for training and 800 tweets for testing (temporal split, i.e., the testing instances are later in time than the training instances). Though this selection might seem arbitrary, all compared algorithms rely on the same sampling, thus, allowing for a fair comparison.

[1] http://data.seattle.gov

Table 1. Results for the random error evaluated in a study on quality of crowdsourced labels. Means (μ) and standard deviation (SD) of the error rates are displayed for each user group.

	Random Error	
	Crowd	Expert
μ	0.0338	0.0323
SD	0.0006	0.0002

4. Study on Quality of Crowdsourced Labels

In active learning, most often a perfect oracle is assumed for labeling instances. As this might not hold true in a real-world environment, we conducted a study on labeling accuracy. When it comes to labeling accuracy, the general assumption is that labeling quality in crowdsourcing environments might be dependent on the domain knowledge of the annotators (Zhao et al., 2011). Thus, one of the goals of the study is to analyze if the labeling quality of non-experts differs significantly from domain experts. To answer this question, we evaluated two user groups in our study: *domain experts* and *regular crowd users* with no or limited domain knowledge. Second, there is no work describing error rates for labeling of incident-related microposts. Thus, we want to quantify the error rates, so we can use them for our simulations. For this, we evaluated the *random error*, i.e., the error that results from the annotator carelessness. E.g., a wrong label is occasionally assigned. The random error is regarded as i.i.d. noise on each label, thus, we assume a fixed probability $RE \in [0, 1]$.

We assume a different labeling quality for crowd users (CU) and domain experts (EX) and test the following hypothesis H: The means (μ) of the *random error* are different across both user groups ($H_0 : \mu_{RE,CU} = \mu_{RE,EX}, H_A : \mu_{RE,CU} \neq \mu_{RE,EX}$).

We created a survey to conduct the labeling of our complete ground truth dataset according to the incident types. Fourteen users participated in the study. Eight participants were crowd users with no or low experience in the crisis management domain and six users were domain experts with more than three years experience in the domain. At least three crowd users and at least two domain experts labeled each tweet. Based on the results, we calculate the random error (cf. Table 1) compared to the ground truth labels.

[2] All datasets will be published at http://www.doc.gold.ac.uk/~cguck001/IncidentTweets/

For evaluating our hypothesis, we first confirmed normal distribution for all error types and both user groups using the *Anderson-Darling* as well as the *Shapiro-Wilk Normality* test. Furthermore, we conducted a two-sample F-test for variances to verify same variances for all combinations with $p < 0.01$. For each combination we conducted the two-sample t-test assuming equal variances. For all combinations the null hypotheses could not be rejected with $p < 0.01$. Thus, for all error types, we cannot assume a difference between both user groups. This means that in our study there is no conceivable difference between domain experts and common crowd users.

One reason might be the rather low sample size. Others might be found in the nature of microposts as they are short and the amount of available information per tweet is limited. Thus, the complexity of the information is low and it is possible to understand the content even as a non-expert. Furthermore, as tweets are send by lots of different individuals, the number of domain specific terms could be rather low compared to specialized texts. Also, as incident-related tweets are common topics compared to physics or medicine, people are somehow used to the vocabulary.

To reflect a real-world situation best, we combined the results of both groups as in typical crowdsourcing studies both groups might be present. Also, the labels of the experts are available anyway for our dataset. This gave us a final error rate of 0.0331 for the random error.

5. Event-Based Clustering

In this section, we show how active learning can be utilized to classify the incident type of microposts. We also introduce our approach and present how we cope with the initial selection problem, i.e., how to select the initial training set, as well as with the query selection problem, i.e., how to choose appropriate instances for labeling in each iteration.

5.1. Active Learning for Event Type Classification

Active learning is an iterative process to build classification models by selecting small subsets of the available instances to label. Two major steps are conducted: (1) a learning step, where a classifier is built and (2) an improvement step, in which the classifier is optimized. We follow a pool-based sampling approach. First, a large number of microposts are collected as an initial pool of unlabeled data U. From this information base, a set of training examples L is chosen for learning an initial model. It is highly important how to choose this set, because with a well-selected initial training set, the learner can reach higher performance faster with fewer queries (Kang et al., 2004).

For training a classifier using this initial set, we reuse the classification approach presented in (Schulz et al., 2013b). Here, microposts are processed with standard Natural Language Processing (NLP) techniques such as stopword removal, POS-tagging, and lemmatization. Afterwards, several features are extracted from the preprocessed instances such as word-3-grams after POS-filtering, TF-IDF scores, syntactic features as well as semantic features. The syntactic features are the number of exclamation and question marks as well as the number of upper case characters. The semantic features are a feature group derived using different means of Semantic Abstraction (Schulz et al., 2015). Furthermore, the existing approach allows us to extract a likely date of an event mentioned in a micropost. To identify the temporal information in a tweet, we adapted the HeidelTime framework for temporal extraction as presented in (Schulz et al., 2013b).

As the number of geotagged microposts is rather low (about 1-2%), we reuse an extension of our approach for geolocalization (Schulz et al., 2013a) of microposts as well as for extracting location mentions as features used in the classification. For geolocalization an estimation of the city and the country where a tweet was send from was used and additionally location mentions extracted from the tweet message were considered. First, we use a Stanford NER[3] model to identify all location mentions. Then, the discovered locations are geocoded using the geographical database GeoNames[4], and the MapQuest Nominatim API[5] for more fine-grained locations, like streets. The intersection of all locations extracted from the tweet is used as an estimation of the location where an event mentioned in a tweet has happened.

After the initial training, the classifier is retrained in several iterations using newly labeled instances. After each iteration, the labeled instances are removed from the pool of unlabeled instances U and added to L, thus, more instances can be used for learning. A selection strategy is used on U to query labels for a number of instances in each iteration. For coping with this query selection problem, several strategies can be chosen based on informativeness and representativeness (Huang et al., 2010).

For informativeness as selection criteria, uncertainty sampling (Lewis & Catlett, 1994) is commonly applied that selects particularly those examples for labeling for which the learner is most uncertain. However, the main issue with the informativeness approach is that only a single instance is considered at a time (Settles, 2012). Thus, outliers could be selected erroneously as the context is not taken into account. In contrary, clustering helps to identify representative instances. According to Nguyen and Smeulders (Nguyen & Smeulders, 2004), the most representative

[3] http://nlp.stanford.edu
[4] http://www.geonames.org/
[5] http://developer.mapquest.com

examples are those in the center of cluster, which are the instances most similar to all other instances in the cluster. Nevertheless, selecting always the centers of the clusters might result in selecting always very similar instances for each iteration, thus, the model might not improve very much. Furthermore, it remains unclear how many clusters have to be built. Also, the resulting clusters not necessarily correlate to the real-world events as spatial and temporal information is neglected.

To overcome the individual problems of each approach, related work proposes to select the most informative *and* representative instances. This results in selecting the instances that are representative for the whole dataset as well as have the highest chance to improve the model. In our approach, we use metadata provided in microposts to cluster instances based on both criteria and to choose the most valuable instances for training the classifier. The whole process of active learning continues until a stopping criteria is met, e.g., a maximum number of iterations is reached or when the model does not improve any more.

5.2. Event-based Clustering

Clustering-based approaches are frequently used for identifying representative instances. However, there might not be an obvious clustering of event-related data, thus, clustering might be performed at various levels of granularity as the optimal number of cluster is unknown.

Consequently, we use event-related information such as temporal and spatial information in combination with the event type to perform an *event-based clustering* to take the properties of real-world events into account. This way, we are directly able to find a number of clusters without the need of specifying the number beforehand. Furthermore, our event-based clustering is based on both selection criteria, so we overcome the limitations of each individual one.

The design of our approach follows the assumption that every event-related information is either related to a real-world event or not. Thus, we propose to cluster all instances based on the three dimensions that define an event: temporal, spatial and thematic extent. As a result, each instance is aggregated to a cluster.

If a micropost lies within the spatial, temporal, and thematic extent of another micropost, it is assumed to provide information about the same event. This assertion can be formalized as a triple of the form $\{event\_type, radius, time\}$. The spatial extent is a radius in meters drawn around the spatial location of the event. The temporal extent is a timespan in minutes calculated from the creation time of the initial event. The thematic extent is the type of an event. For example, for our approach we use the rule $\{Car\_Crash, 200m, 20min\}$, which asserts that each incoming micropost of the event type *Car Crash* is aggregated to a previously reported incident if it is of the same type, within a range of 200 meters, and within a time of 20 minutes. Clearly, altering the radius or the time will have a strong effect on the final clustering. However, as emergency management experts suggested to use these values, we did not change them. Inspecting the effects of different parameterizations remains subject for future work, however, we are confident that our proposed approach is not affected negatively by a change of these parameters. With the help of these three assertion types, a rule engine computes whether microposts are clustered as they describe the same event or not.

Microposts containing no thematic information are assigned the $unknown\_event$ type. Missing spatial information is replaced with a common spatial center (the center of a city). Missing temporal information is replaced with the creation date of the micropost. Thus, even with one or two missing dimensions, we are still able to build clusters. Based on this clustering approach, we are able to cluster all microposts related to a specific event. This helps to identify those microposts that might be helpful for better training. Opposed, microposts not related to events are assigned to larger clusters, containing lots of noise and being less valuable for the learning process.

Algorithm 1 Algorithm for initial selection strategy.

Data: Unlabeled instances U, Clusters C generated by event-based clustering, Size of initial training set b_i
Result: Instances to label L

for all $clusters\ c \in C$ **do**
  for all $instances\ i \in c$ **do**
    Calculate information density $DS(i)$
  end for
end for
for all $clusters\ c \in C$ **do**
  Calculate average information density $DSC(c)$
end for
Order clusters in C based on DSC
while $|L| \leq b_i$ **do**
  for $cluster\ c \in C$ **do**
    Add one instance from c to L
  end for
end while

5.3. Initial Selection Strategy

The initial dataset that needs to be labeled is selected first. Related approaches rely on random sampling or clustering techniques (Zhu et al., 2008). However, this does not guarantee the selection of appropriate instances, because the initial sample size is rather small, whereas the size of the clusters is large. In contrast, event-based clustering uses the properties of real-world events to perform an initial clustering.

Our approach for selecting the initial dataset is shown in Algorithm 1. Based on the set of clusters resulting from our

event-based clustering, the most representative instances for the complete and unlabeled dataset are identified for training the initial model. For this, we use the event clusters ordered by information density of their containing instances to obtain a good initial set. Selecting informative instances clearly is not possible yet, as a classifier cannot be trained at this point. In the following, we describe the algorithm in detail.

First, our clustering approach is applied on the complete unlabeled set U without a thematic specification as this is not present yet. Thus, the $unknown\_event$ type is used. Second, for all instances in each cluster the information density is calculated. This is done based on how many instances are similar or near to each other, thus, outliers are regarded as less valuable. We used a k-Nearest-Neighbor-based density estimation (Zhu et al., 2008): $DS(x) = \frac{\sum_{s \in S(x)} \text{Similarity}(x,s)}{k}$

The density $DS(x)$ of instance x is estimated based on the k most similar instances in the same cluster[6] $S(x) = \{s_1, s_2, ..., s_k\}$. As a similarity measure, we use the cosine similarity between two instances. The information density DSC of each cluster c is then calculated based on the average of the information density of each instance as follows: $DSC(c) = \frac{\sum_{x \in c} DS(x)}{k}$

Doing this, we are able to avoid noisy clusters with lots of unrelated items, which would typically be clusters not related to an event. Based on $DSC(c)$ the clusters are sorted. Then we iterate over the ordered list and select instances until b_i (initial training size) instances are selected. Proceeding this way, we achieve a good distribution over all valuable event clusters as it is guaranteed that the instances are selected from the most representative clusters. Based on these instances, the initial model is build.

5.4. Query Selection Strategy

For the query selection strategy we choose representative using clustering as well as informative instances using uncertainty-based sampling. The pseudo-code is shown in Algorithm 2. In every iteration, the classifier trained on the currently labeled instances is applied to label all unlabeled instances. As a result, every instance is assigned a thematic dimension. Based on this, the event clustering is applied using the spatial, temporal, and thematic information resulting in a set of clusters C.

Next, for the query selection strategy, we calculate the information density DS per instance. For identifying informative instances, we use the instances for which the classifier is most uncertain. As an uncertainty measure the entropy calculated for each instance x and each class $y \in Y = \{y_1, y_2, ..., y_i\}$ was employed: $H(x) = -\sum_{y \epsilon Y} P(y|x) \log P(y|x)$

[6] k is equal to the number of instances in the cluster.

Algorithm 2 Algorithm for one iteration of the query selection strategy.

Data: Unlabeled instances U, Labeled instances L, Clusters C generated by event-based clustering, Number of instances to label per iteration b_i, Trained Model for iteration M, Mean average size of all cluster in iteration ms
Result: Instances to label L

```
Use L to train classifier M
for all clusters c ∈ C do
  for all instances i ∈ c do
    Calculate information density DS(i)
    Calculate entropy H(i) using M
    Calculate density×entropy measure DSH(i)
  end for
end for
for all clusters c ∈ C do
  Calculate DSHC(c)
end for
Order clusters based on DSHC
while |L| ≤ b_i do
  for all clusters c ∈ C do
    n = log_ms(|c|)
    Add n instances from c to L
  end for
end while
```

Based on the information density and the entropy, the density×entropy measure $DSH(x) = DS(x) \times H(x)$ (Zhu et al., 2008) is calculated for each instance x. The informativeness and representativeness of each cluster is then computed based on the mean average of DSH of each instance i in the cluster c: $DSHC(c) = \frac{\sum_{i \in c} DSH(i)}{|c|}$

For selecting the appropriate instances to query, the clusters are sorted by the $DSHC$ of each cluster. The number of instances to draw per cluster is calculated as $n = \log_{(ms)} CS$. To determine how many instances have to be selected per cluster (n), we calculate the average size of all clusters ms and the size of the current cluster CS. We decided to use a logarithmic scale by using a logarithm at basis ms to avoid drawing too many instances from larger clusters as would be the case with a linear approach. We assume that drawing only small numbers per cluster is sufficient, as at some point additional instances will not yield any additional information, as the instances will be too similar to each other. Instances are selected until the number of instances to label per iteration is reached. Based on the previous and the new instances the model is retrained. The whole process is repeated until all iterations are finished.

6. Experiments

We conducted two experiments regarding incident type classification. First, we compared related approaches to show that event-based clustering outperforms other clustering-based active learning approaches. Additionally, the effect of the number of labeled instances on the classifier performance is examined. In the second experiment,

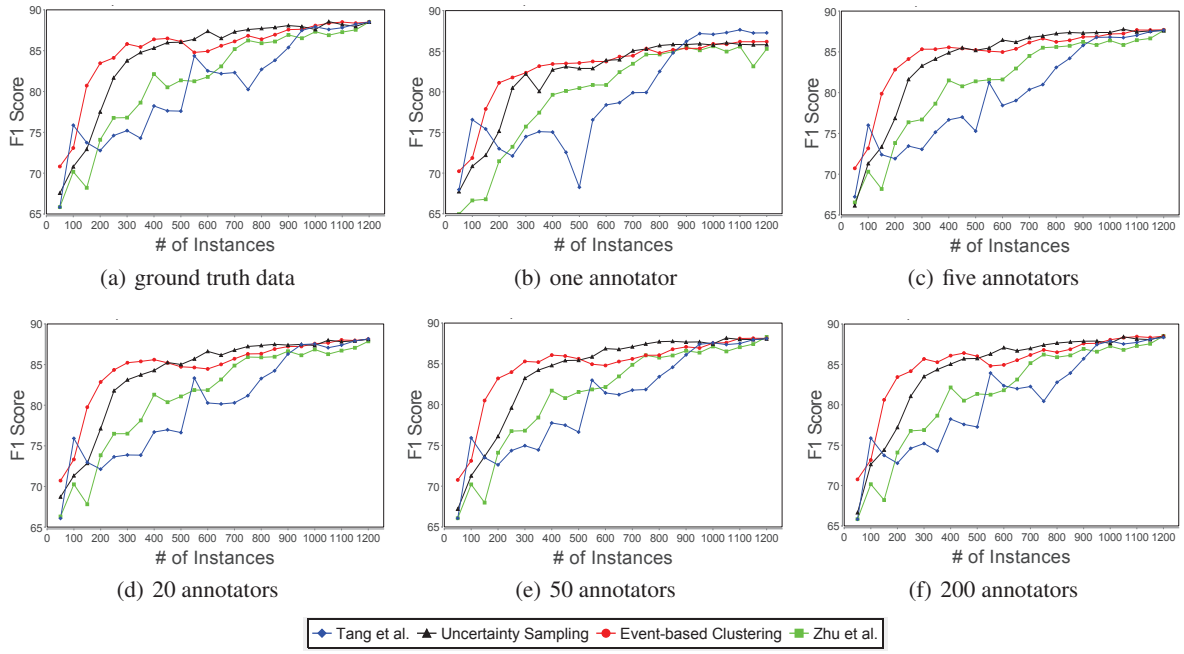


Figure 1. Evaluation results of state-of-the-art selection strategies and our approach. The graphs for different number of annotators (regular crowd users) are shown. Note that the more annotators labeled an instance the lower is the probability for a noisy labeling.

the influence of noise is inspected. Noise results from a low number of non-expert labelers. For keeping costs low a classifier should not be affected by a bad labeling.

6.1. Classification and Metrics

The active learning algorithms select instances from the training set to query for labels. Based on these, a classifier was trained and evaluated on the test set. As classifier we used Weka's implementation of John Platt's sequential minimal optimization (SMO) algorithm for training a support vector machine (Platt, 1998). Due to the complexity of determining best parameters for each iteration and each approach, we followed related approaches (see (Huang et al., 2010) and (Donmez et al., 2007)), and decided to compare all algorithms on fixed parameters. Consequently, the SVM was used with standard settings.

For comparison, the deficiency metric (Raghavan et al., 2006) is calculated using the achieved F1 score of all iterations of a reference baseline algorithm (REF) and the compared active learning approach (AL). The result is normalized using the largest F1 score and the learning curve of the reference algorithm REF. Thus, the measure is non-negative and values smaller than 1 indicate more efficient algorithms compared to the baseline strategy, whereas a value larger than 1 indicates a performance decrease compared to the baseline strategy.

6.2. Algorithms and Parameters

In order to evaluate the performance of our approach, we re-implemented the following related approaches:

(Tang et al., 2002): For initial sampling a k-means clustering is used. For query selection, first the most uncertain instances for each cluster are selected. Then, information density is used to weight the examples. We set $k = 4$, because we have four different event types.

(Zhu et al., 2008): For initial sampling a k-means clustering is used ($k = 4$). During the iterations, the entropy $\times$ density measure is used as selection criteria and no clustering is applied.

Uncertainty: Random instances (initial) and the entropy-based uncertainty sampling (iterations) is used.

Event-based clustering: Our event-based clustering is applied with a spatial extent of 200m and a temporal extent of 20min.[7][8]

Following the experimental settings of (Hu et al., 2013) and (Huang et al., 2010), we set the size of the initial training set and the size during the iterations to 50. No further tuning or parameterization was applied. Each iteration for

[7] As a result, the 1,200 tweets of the training set are divided into 438 distinct event clusters.

[8] The spatial and temporal extent are a result of discussions with emergency managers.

Table 2. Deficiencies with Tang et al. as a baseline strategy.

Approach	Deficiency
(Tang et al., 2002)	1
Uncertainty Sampling	0.53
(Zhu et al., 2008)	0.90
Event-based Clustering	0.44

each algorithm was repeated 10 times, as for instance, the uncertainty approach is highly dependent on the selected instances. We used averaged F1 based on the repetitions.

6.3. Comparison to state-of-the-art approaches

The performance graph for the ground truth data is shown in Figure 1 (a). Note that the x-axis shows the *total* number of instances combining the 50 instances of the initial training set and 50 instances drawn per iteration. Thus, iterations from 0 to 23 are depicted. As shown, the performance after selecting the initial training set is superior with our approach. Also, in regions where only a few instances were labeled, the event-based clustering has a higher F1 value. This shows that a high-quality selection of the iteration instances is possible with our method.

Table 2 shows the deficiencies. With respect to the performance of the iterations, our approach has a decreased deficiency compared to other clustering approaches (0.44 *vs.* 0.53). The approach of Zhu et al. outperforms the approach of Tang et al. in most iterations and also with respect to the deficiency. We attribute this to the improved strategy for query selection. A surprising result is the performance of uncertainty sampling that outperforms the other two clustering strategies. Apparently, only focusing on the informativeness seems to be a good strategy for our dataset. In contrast, using the number of distinct events as the number of clusters might not be the most efficient approach.

The graph also shows that our approach has a steep learning curve as for instance only a sixth of all instances are needed to achieve a F1 score of about 84%. This is especially important when it comes to labeling costs, as only a limited amount of data would need to be labeled. One can see a drop at 500 instances. This is most likely because with more instances the number of clusters is decreasing, thus, selecting appropriate instances is more difficult.

We can conclude that event-based clustering that takes representative as well as informative instances into account is a promising strategy for active learning. We also showed that our approach outperforms state-of-the-art for selecting an initial training set and for choosing appropriate instances for labeling in each iteration.

Influence of noise in the labels In Figure 1 (b) and 1 (c), the learning curves for the very error-prone cases with one respective five annotators are shown. As can be seen in the curve of the approach of Tang et al., the influence of noise is notable in the big drop with 500 instances. Also Zhu et al.'s approach has a much lower initial F1 score compared to all others, which is an indicator for an inappropriate initial selection strategy. The results indicate that even with noisy labels, our approach outperforms the state-of-the-art as the situation in the graphs of the lower part of Figure 1 does not change much. In all these cases, the learning curves are quite similar, which is a result of the decreased number of wrongly labeled instances. Clearly, the performance of all approaches increases with a lower number of errors.

As we showed, our approach outperforms related work also if noise is taken into account. Not surprisingly, we found that with an increasing number of annotators, noise is negligible. With only one annotator, the deficiency is worse by 57% and with five annotators still worse by 26%. Even with 50 annotators, the deficiency still is worse by 10%. For more than ten annotators, an F1 score of 85% is reached comparably fast. With a maximum of five annotators, this level is only reached at the end of the simulation. For one annotator, this maximum is never achieved. These results indicate that a minimum number of annotators is needed for achieving good results by crowdsourced labeling tasks. In our experiments, ten annotators seem to be sufficient, while in other domains with different error rates, there might be a need for much more annotators.

7. Conclusion

We presented an event-based clustering strategy for event type classification of microposts and coped with several problems of active learning in the emergency management domain. First, it was shown that domain experts do not differ significantly from regular crowd users when it comes to labeling quality. Second, we presented a novel selection strategy for active learning based on temporal, spatial, and thematic information. Our event-based clustering that identifies representative as well as informative instances outperforms state-of-the-art clustering approaches. On incident-related microposts we showed that a better initial training set is selected as well as to appropriate instances for labeling in each iteration are chosen. The learning curve indicated that only a sixth of all instances are needed to achieve a F1 score of about 84%, which is especially important when it comes to labeling costs, as only a limited amount of data would need to be labeled to achieve good classification results.

In the future, we aim at using our active learning framework in addition to labeling of single features. Furthermore, though our framework follows a general approach, we only evaluated it on incident-related data, thus, we also want so show the applicability on other types of events.

References

Donmez, Pinar and Carbonell, Jaime G. Proactive learning: cost-sensitive active learning with multiple imperfect oracles. In *CIKM'08*, pp. 619–628, 2008.

Donmez, Pinar, Carbonell, Jaime G., and Bennett, Paul N. Dual strategy active learning. In *ECML'07*, pp. 116–127, 2007.

Hoi, Steven C. H., Jin, Rong, and Lyu, Michael R. Large-scale text categorization by batch mode active learning. In *WWW'06*, pp. 633–642, 2006.

Hsueh, Pei-Yun, Melville, Prem, and Sindhwani, Vikas. Data quality from crowdsourcing: A study of annotation selection criteria. In *NAACL HLT'09*, pp. 27–35, 2009.

Hu, Xia, Tang, Jiliang, Gao, Huiji, and Liu, Huan. Actnet: Active learning for networked texts in microblogging. In *SIAM'13*, 2013.

Huang, Sheng-Jun, Jin, Rong, and Zhou, Zhi-Hua. Active learning by querying informative and representative examples. In *NIPS*, pp. 892–900, 2010.

Kang, Jaeho, Ryu, Kwang R., and Kwon, Hyuk C. Using Cluster-Based Sampling to Select Initial Training Set for Active Learning in Text Classification. In *PAKDD'04*, pp. 384–388, 2004.

Lewis, David D. and Catlett, Jason. Heterogeneous uncertainty sampling for supervised learning. In *ICML'94*, pp. 148–156, 1994.

Nguyen, Hieu T. and Smeulders, Arnold. Active learning using pre-clustering. In *ICML'04*, pp. 79–86, 2004.

Platt, J. Fast training of support vector machines using sequential minimal optimization. In Schoelkopf, B., Burges, C., and Smola, A. (eds.), *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998.

Raghavan, Hema et al. Active learning with feedback on both features and instances. *J. of Machine Learning Research*, 7:1655–1686, 2006.

Schulz, Axel. *Mining User-Generated Content for Incidents*. PhD thesis, TU Darmstadt, 2014. URL http://tuprints.ulb.tu-darmstadt.de/4107/.

Schulz, Axel, Hadjakos, Aristotelis, Paulheim, Heiko, Nachtwey, Johannes, and Mühlhäuser, Max. A multi-indicator approach for geolocalization of tweets. In *Proceedings of the Eight International Conference on Weblogs and Social Media (ICWSM)*, pp. 573–582, Menlo Park, California, USA, 2013a. AAAI Press.

Schulz, Axel, Ristoski, Petar, and Paulheim, Heiko. I see a car crash: Real-time detection of small scale incidents in microblogs. In *The Semantic Web: ESWC 2013 Satellite Events*, volume 7955 of *Lecture Notes in Computer Science*, pp. 22–33. Springer Berlin Heidelberg, 2013b.

Schulz, Axel, Guckelsberger, Christian, and Janssen, Frederik. Semantic abstraction for generalization of tweet classification: An evaluation on incident-related tweets. In *Semantic Web Journal: Special Issue on The Role of Semantics in Smart Cities (to appear)*. IOS Press, 2015.

Settles, Burr. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.

Shen, Dan, Zhang, Jie, Su, Jian, Zhou, Guodong, and Tan, Chew-Lim. Multi-criteria-based active learning for named entity recognition. In *ACL'04*, 2004.

Tang, Min, Luo, Xiaoqiang, and Roukos, Salim. Active learning for statistical natural language parsing. In *ACL'02*, pp. 120–127, 2002.

Thongsuk, Chanattha, Haruechaiyasak, Choochart, and Meesad, Phayung. Classifying business types from twitter posts using active learning. In *I2CS'10*, pp. 180–189, 2010.

Tong, Simon and Koller, Daphne. Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.*, 2:45–66, 2002.

Wallace, Byron C., Small, Kevin, Brodley, Carla E., and Trikalinos, Thomas A. Who should label what? instance allocation in multiple expert active learning. In *SDM'11*, 2011.

Zhao, Liyue, Sukthankar, G., and Sukthankar, R. Incremental Relabeling for Active Learning with Noisy Crowdsourced Annotations. In *SocialCom'11*, pp. 728–733, 2011.

Zhu, Jingbo, Wang, Huizhen, Yao, Tianshun, and Tsou, Benjamin K. Active learning with sampling by uncertainty and density for word sense disambiguation and text classification. In *COLING'08*, pp. 1137–1144, 2008.

Towards detection of faulty traffic sensors in real-time

Nikolas Zygouras, Nikolaos Panagiotou,
Ioannis Katakis, Dimitrios Gunopulos {NZYGOURAS,N.PANAGIOTOU,KATAK,DG}@DI.UOA.GR
University of Athens, Athens, Greece

Nikos Zacheilas, Ioannis Boutsis, Vana Kalogeraki {ZACHEILAS,MPOUTSIS,VANA}@AUEB.GR
Athens University of Economics and Business, Athens, Greece

Abstract

Detecting traffic events using the sensor network infrastructure is an important service in urban environments that enables the authorities to handle traffic incidents. However, irregular measurements in such settings can derive either from faulty sensors or from unpredictable events. In this paper, we propose an efficient solution to resolve in real-time the source of such irregular readings by examining the correlation and the consistency among neighbor sensors and exploiting the wisdom of the crowd. Our experimental evaluation illustrates the efficiency and practicality of our approach.

1. Introduction

Sensor network infrastructures have been widely used for traffic management in smart cities to provide important services for the benefit of pedestrians, cyclists, motorists and public transport. Such services are typically provided by analyzing data provided by heterogeneous static and mobile sensors. This enables the implementation of numerous applications like proposing alternative routes, altering traffic lights, etc.

The most common type of sensor which is utilized in such environments is the SCATS sensor. They are static sensors embedded at the city roads providing rich, real-time information such as traffic flow measurements based on vehicles that cross a specific segment. Despite their utility in many traffic applications, SCATS sensors can be faulty. Thus, one fundamental challenge in these settings is how to efficiently distinguish between irregular and faulty measurements before taking any unnecessary actions.

Proceedings of the 2^{nd} International Workshop on Mining Urban Data, Lille, France, 2015.

Automatic identification of anomalies in streaming data is an emerging field of research due to the large number of applications (intrusion detection, event identification, etc). Many algorithms that utilize machine learning and time series analysis techniques have been successfully applied for the detection of unexpected events during the last years (Yi et al., 2000). These methods offer high quality results and are able to perform on massive data streams in real-time. An interesting use-case is the automatic analysis of traffic data generated by Smart Cities infrastructures. Human personnel are unable to monitor and efficiently identify problems on these data. The utilization of anomaly detection techniques would provide great assistance to traffic operators as it would enable the automatic real-time identification of traffic issues.

Recently, Crowdsourcing has emerged as an attractive paradigm to exploit the intelligence of ubiquitous human crowd (citizens) to extract useful information. Traditional Crowdsourcing systems such as AMT[1], CrowdFlower[2], etc., constitute marketplaces for human intelligence tasks (HITs), that allow a requester to define a task, which is performed by other human workers in exchange for a reward. For example, mobile human workers with different characteristics can be queried for geo-located tasks to extract real-time information without needing an expensive infrastructure(Boutsis & Kalogeraki, 2014).

In this paper we develop an efficient approach that identifies faulty readings from traffic sensors by examining the correlations among them and by taking advantage of the ubiquitous citizens through Crowdsourcing. We summarize our contributions below:

- We present an efficient approach that identifies anomalous sensors and uses Crowdsourcing to resolve whether irregular measurements are due to faulty sensors or irregular traffic.

[1] http://www.mturk.com/
[2] http://www.crowdflower.com/

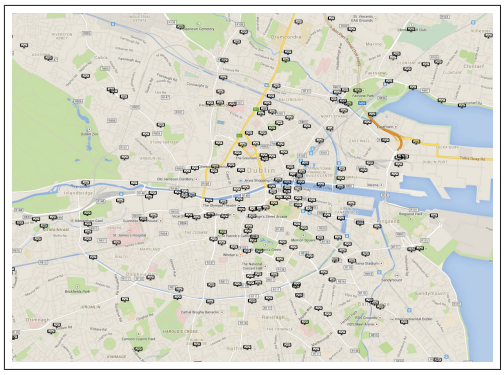


Figure 1. The SCATS sensors locations at Dublin's city centre

- We tackle the problem of automatically detecting anomalous SCATS sensors with three methods: (i) Pearson's correlation, (ii) cross-correlation and (iii) multivariate ARIMA model. The proposed methods have to tackle the task efficiently in real-time.
- We develop our approach using the Lambda-Architecture which combines a batch processing framework (i.e. Hadoop[3]) and a distributed stream processing system (i.e. Storm[4]) for efficiently processing both historical and real-time data.
- We develop a Crowdsourcing system used to extract answers from the human crowd based on the MapReduce paradigm.
- We provide an experimental evaluation, which illustrates that our approach is practical and can effectively identify irregular measurements in real-time.

2. Problem Description and System Model

2.1. Smart City

Smart cities exploit digital sensor devices that can be either embedded at the city infrastructure or they can be mobile (*e.g.*, smartphones) in order to provide services for their citizens that enhance their well-being. Such services may relate to traffic management, housekeeping information, etc.

In this paper we focus on Dublin, a smart city that utilizes sensors for supervising and managing road traffic (Kinane et al., 2014). In Dublin the traffic is controlled by the Dublin City Council (DCC), which is responsible to develop, maintain and manage the city road network. To achieve that they exploit several heterogeneous data sources that include: (i) SCATS sensors which are embedded on the road and monitor real-time traffic density, (ii) GPS traces from sensors embedded on buses, (iii) the LiveDrive radio where users can report traffic, and (iv) pedestrian counters.

2.2. System model

In this section we provide our system model for the data sensors that we examine, namely the SCATS sensors and Crowdsourcing.

SCATS Sensors. SCATS (Sydney Coordinated Adaptive Traffic System) is an innovative computerized traffic management system developed by Roads and Maritime Services (RMS) Australia. SCATS sensors are fixed magnetic sensors deployed on intersections to measure the traffic flow and the degree of saturation of roads' lanes. In Dublin city, each SCATS sensor produces and transmits a new record every minute. Each record contains information related to the timestamp t of the measurement, the sensor's ID i and finally the degree of saturation and traffic flow measurements. In the provided dataset there are approximately 300 SCATS controlled intersections and 1000 different SCATS sensors throughout the road network. The GPS locations of the SCATS sensors are presented in Figure 1. Degree of saturation measures how much a road's lane is utilized, while traffic flow measures the vehicles' volume divided by the highest volume that has been measured in a sliding window of a week[5]. In this work we decided to monitor the degree of saturation value, noted as s, as it more reliable and informative than the traffic flow. The degree of saturation of a particular SCATS sensor with $ID\ i$ at the timestamp t is noted $s_{i,t}$.

Crowdsourcing. Our crowdsourcing system comprises a set of human workers denoted as w_j which are able to receive task assignments. Tasks are being inserted to the system by an authority, such as the DCC. Each task t_k is associated with a number of attributes as $< id_k, latitude_k, longitude_k, reward_k, description_k >$. Hence, every task posses a unique identifier (id_k), the geographical coordinates of the location that the task involves ($latitude_k$, $longitude_k$), the corresponding reward ($reward_k$) for executing the task and a task description that describes the information that needs to be provided by the human worker. An example of such a task description is: *"Is there traffic in O'Connell Street? Yes/No"*. Finally each response provided by a worker is captured with a record by our system using the worker and the task identifiers, coupled with the response as follows: $< w_j, id_k, response_{jk} >$.

3. Architecture

In Figure 2 we display our system architecture which consists of the following components: (i) a Distributed

[3] https://hadoop.apache.org/

[4] https://github.com/nathanmarz/storm

[5] http://dublinked.com/datastore/datasets/dataset-274.php

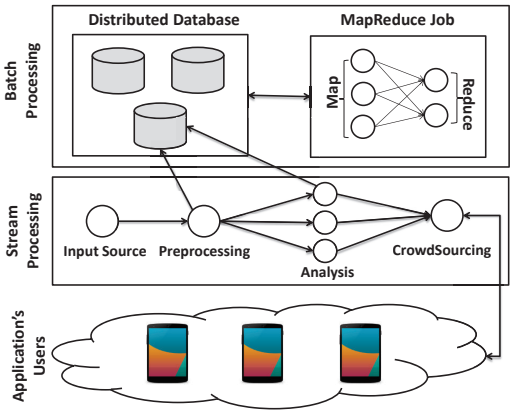


Figure 2. System Architecture

Stream Processing System (DSPS), (ii) a batch processing framework, (iii) a distributed database system, and (iv) the Crowdsourcing component that consists of the users' mobile devices. Our architecture is an instance of the Lambda-Architecture[6] as we exploit the fast processing offered by DSPS and the fault-tolerance and parallelism provided by current batch processing frameworks.

Incoming SCATS-sensor data are forwarded to a stream processing graph. These data are pre-processed and stored in the Distributed Database (i.e. Preprocessing component in Figure 2) for further processing by the batch processing component. We analyze the reported metrics via the Analysis component which examines if one of the sensors deviates significantly from its neighbors so it could possibly be a faulty sensor. This component uses both the current conditions and historical data for identifying such conditions. In case that one such sensor is detected, the Analysis component informs the Crowdsourcing component about this situation. The latter is responsible to send the appropriate Crowdsourcing tasks that will enable us to detect if the sensor is a faulty-one. Finally, the batch processing component periodically computes new statistics about the historical sensor data.

There are multiple DSPSs which support low latency processing in real-time. Some of these systems are Apache Storm, Spark Streaming[7] and TUD-Streams (Bockermann & Blom, 2012). We used Storm as the DSPS that will perform the real-time processing of incoming sensor data. Storm is one of the most commonly used DSPS, and is supported by major companies such as Twitter[8]. It has been successfully applied for processing high volume of data in different application domains, achieving high throughput and low response latencies (McCreadie et al., 2013). Furthermore, we decided to use Storm due to its scalability features that we also exploit in our previous work (Zygouras et al., 2015). Storm users can change the parallelism of the processing components to adapt to possibly workload bursts.

Finally, for the analysis of the historical sensor data we used the most commonly used open-source implementation of the MapReduce programming model, Hadoop. We execute periodical (i.e. at the end of each day) Hadoop jobs for computing the basic metrics required by our proposed techniques, described in more detail in Section 4. Our jobs retrieve historical data from a distributed database, more specifically MongoDB[9]. We decided to use MongoDB instead of the Hadoop Distributed Filesystem (HDFS), as we want to have fast access to the data from the DSPS component of our architecture, for computing and storing short-term statistics in real-time.

4. Methodology

The goal of this work is to monitor the streaming traffic data and automatically pose Crowdsourcing tasks when anomalous sensors are identified. In order to identify anomalous sensors we propose three different outlier tests that examine whether the SCATS sensors behave differently from their normal behavior. These outlier tests are based on the following statistical measurements: (i) Pearson's Correlation (ii) Cross-Correlation and (iii) the ARIMA Model. The normal behavior for each sensor is calculated offline using the historical data. These methods are implemented using the Lambda architecture and Crowdsourcing tasks are assigned to users when anomalous SCATS sensors are identified.

4.1. Identifying Anomalous Sensors

In this section we describe the three statistical measurements that are used and we explain how these are utilized to detect anomalous SCATS sensors. Initially we applied a simple statistic measurement named Pearson's correlation that identifies the correlation between pairs of SCATS sensors. Then we used an extension of the first method, named cross-correlation, to identify how many lags we should shift backward a sensor's values to maximize its pairwise correlation with another adjacent sensor. The first two approaches use two well known measures in time series analysis. The disadvantage is that they check pairs of sensors and not the group of sensors as a whole. For this reason we applied a third approach that can be thought as a multivariate ARIMA model which deals with the aforementioned problem and is faster than the other approaches.

[6] lambda-architecture.net
[7] https://spark.apache.org
[8] http://twitter.com
[9] http://www.mongodb.org/

4.1.1. PEARSON'S CORRELATION

The Pearson's correlation coefficient is a well known statistic that measures the linear relationship of two variables X and Y. It takes values in $[-1, 1]$, where 1 means that the variables are positively correlated, -1 stands for negative correlation and 0 for no correlation between X and Y. The Pearson's correlation, noted $\rho_{X,Y}$, is calculated by dividing the covariance of X and Y with the product of the standard deviations of X and Y (see Equations 1 and 2).

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{1}$$

$$cov(X,Y) = E[(X - \mu_X)(Y - \mu_Y)] \tag{2}$$

In our scenario we calculated the pair-wise correlation between all SCATS sensors X and Y whose spatial distance does not exceed a predefined threshold. This restriction creates a sparse correlation matrix that contains non-zero elements when SCATS sensors are spatially adjacent. We calculate the sparse correlation matrix from the historical data. Then, utilizing the streaming data that arrive continuously in our system we periodically calculate the streaming correlation of the adjacent SCATS sensors. We note as noisy sensors' pairs those that their streaming correlations disagree significantly with the correlations calculated from the historical data. If a particular sensor disagrees significantly with the majority of his neighbors then a crowdsourcing task is posed.

4.1.2. CROSS-CORRELATION

Cross-correlation is a statistical measure of similarity between two variables X and Y as a function of the lag of one relative to the other. More specifically cross-correlation between X and Y is calculated by shifting forward or backward Y and calculating its correlation coefficient with X. Cross-correlation with lag d, noted $\rho_{X,Y}(d)$, is calculated as seen in Equation 3. The numerator of the equation calculates the covariance of X and Y shifted d time bins backward. Finally the denominator is the product of the standard deviations of X and the lagged Y.

$$\rho_{X,Y}(d) = \frac{\sum_i [x(i) - \mu_X)(y(i+d) - \mu_Y)]}{\sqrt{\sum_i (x(i) - \mu_X)^2} \sqrt{\sum_i (y(i+d) - \mu_Y)^2}} \tag{3}$$

A traffic anomaly at a particular location, in a road network, may require some time in order to be propagated to the adjacent sensors. This observation motivates us to consider the cross-correlation between adjacent SCATS sensors. More specifically we calculated the d_{max} that maximized the correlation between two adjacent sensors X and Y (see Equation 4).

$$d_{max} = \arg\max_d (\rho_{X,Y}(d)) \tag{4}$$

In order to identify anomalies with cross-correlation we followed a similar approach to the one utilizing the Pearson's correlation measure, described before. The main difference is that we identified, using historical data, the lag d_{max} that maximized the correlation between two SCATS sensors X and Y. In the streaming analysis in order to calculate the cross-correlation between the sensors we shifted d_{max} lags backward the Y and we calculated its correlation with X. Finally, we measured how much the streaming cross-correlation deviates from the offline calculated cross-correlation between X and Y using the optimal lag value d_{max}.

4.1.3. MULTIVARIATE ARIMA MODEL

A common strategy to detect outliers in multivariate time series (Yi et al., 2000) is to build a regression model for each time series and evaluate whether the actual values vary significantly from the predictions. The model receives as input the previous L degree of saturation measurements for a particular sensor with $ID = 0$ and the sensor's N nearest SCATS sensors $\{s_{i,j} : i \in [0, N], j \in [0, L], i, j \in \mathbb{Z}\}$. The goal of the model is to make the best prediction for $s_{0,t}$, denoted as $\hat{s_{0,t}}$. The model is presented in detail in Equation 5. This model can be thought as a multivariate ARIMA model, as multiple sensors are used in order to make the predictions.

$$\begin{aligned} \hat{s_{0,t}} =& \phi_{0,1} s_{0,t-1} + \cdots + \phi_{0,L} s_{0,t-L} + \\ & \phi_{1,0} s_{1,t} + \phi_{1,1} s_{1,t-1} + \cdots + \phi_{1,L} s_{1,t-L} + \\ & \cdots \\ & \phi_{N,0} s_{N,t} + \phi_{N,1} s_{N,t-1} + \cdots + \phi_{N,L} s_{N,t-L} \end{aligned} \tag{5}$$

In the training phase we use the historical degree of saturation values in order to calculate the coefficients Φ of Equation 5. In order to solve this problem we created the matrix A and vector b containing the input data (degree of saturation values) and the target values respectively. The Φ parameters are the values that optimally solve Equation 6. The solution of this system is given with the pseudo-inverse transformation of the input presented in Equation 7. The key property of this approach, in contrast to the two previously described techniques, is that it monitors the different sensors together as a whole. The Pearson's correlation and the cross-correlation approaches investigated only pairwise correlation between SCATS sensors, ignoring potentially useful information. On the other hand, the ARIMA-based method aims at exploiting this information.

$$\Phi = \begin{bmatrix} \phi_{0,1} & \dots & \phi_{0,L} & \dots & \phi_{2,0} & \dots & \phi_{2,L} \end{bmatrix}$$

$$A = \begin{bmatrix} s_{0,t-1} & \dots & s_{0,t-L} & \dots & s_{N,t} & \dots & s_{N,t-L} \\ s_{0,t-2} & \dots & s_{0,t-L-1} & \dots & s_{N,t-1} & \dots & s_{N,t-L-1} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ s_{0,L} & \dots & s_{0,0} & \dots & s_{N,L} & \dots & s_{N,0} \end{bmatrix}$$

$$b = \begin{bmatrix} s_{0,t} & \dots & s_{0,L+1} \end{bmatrix}^\top$$

$$b = A\Phi \tag{6}$$

$$\hat{\Phi} = (A^T A)^{-1} A^T b \tag{7}$$

In order to integrate this approach we split the historical data in training and test set. Initially we calculated offline, using the training set, the $\hat{\Phi}$ parameters. These parameters are the coefficients regarding the sensor's previous measurements and its adjacent sensors' past measurements. Then we calculated how well the data fitted to these models computing for each sensor its Mean Absolute Error (MAE). Finally, in order to identify anomalous SCATS sensors while monitoring the streaming data we compute for each sensor its MAE at a particular time window. We label a sensor as 'anomalous' if its streaming MAE noticeably differs from its MAE measured using the testing set.

4.2. Implementation

Our system calculates the correlation among adjacent SCATS sensors. This is achieved by adding the SCATS sensors' GPS locations in a k-d tree data structure during system initialization and calculating the k nearest SCATS sensors for each sensor. Furthermore, we developed our system using the Lambda architecture. So we should ensure that the required data are transmitted to the appropriate cluster nodes. Thus, we created a mapping of each SCATS sensor ID to one or more cluster nodes. This guarantees that each computing node contains all the required data for a sensor's adjacent sensors.

We define three parameters that help us configure the components of our system. The first one is $job\_periodicity$ and defines when the batch jobs should re-execute (e.g. each day, every week). The other two control the stream processing computations. More specifically, the $stream\_threshold$ parameter defines how often we should re-compute the examined metrics (e.g. every ten minutes), while $time\_window$ defines the sliding time window (e.g. the previous hour) that will be used for keeping the past sensor data necessary for the computations.

As we described in Section 3, we periodically invoke Hadoop jobs that compute the different metrics we explained in Section 4.1. Map tasks read the pre-processed sensor data from the MongoDB, and send them to the reduce tasks. We partition the data based on the SCATS sensor ID to cluster node mapping. The idea is that neighboring sensors should always end up on the same reduce task in order to appropriately compute the examined metrics. Each sensor may belong to more than one nodes in such cases we send the tuple multiple times (i.e. equal to the number of nodes it is part of) to avoid information loss.

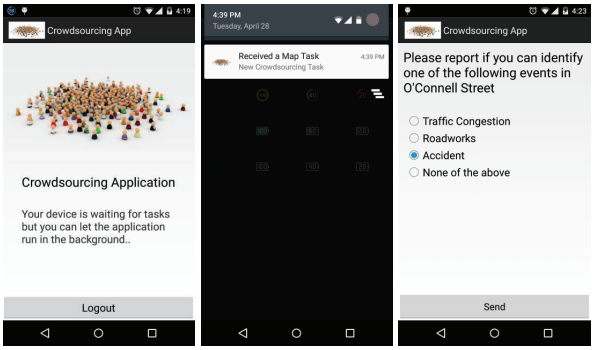


Figure 3. Crowdsourcing Application (a) Main Application, (b) Push Notification, (c) Map Task

Reduce tasks are responsible for computing the metrics described in Section 4.1 and store the results in MongoDB.

In the stream processing component, we implemented a Storm topology (see Figure 2) that processes the real-time sensor data. We exploit the parallelism offered by Storm by having multiple instances of our Analysis component, running in parallel, in order to decrease latency. The topology pre-processes the incoming data and stores them in MongoDB. Also the pre-processed data are sent to the component that invokes the three different techniques. Again we partition the data based on the offline mapping, to guarantee that all neighboring data will be processed by the same component's instance. Detected events are forwarded to the Crowdsourcing component that is responsible to inform the users that will help us detect if the sensor is faulty.

4.3. Crowdsourcing System

Misco. Our Crowdsourcing system has been developed using the Misco framework (Dou et al., 2011; 2010; Kakantousis et al., 2012), which is based on the MapReduce paradigm and tailored for mobile devices to provide an extensible and efficient way to develop distributed applications.

Our Crowdsourcing system is structured using (i) a *Master Server* that keeps track of the tasks t_k submitted when anomalies are detected from SCATS sensors, assigns them to human workers w_j and returns the responses to the system, and (ii) the *Workers* who are the human contributors that process the crowdsourcing tasks. Each Worker is responsible to process queries and return the results to the server. These tasks are executed by workers through their personal smartphone devices or tablets.

Task assignment. Suppose that we need to exploit Crowdsourcing to determine the source of an event using a task t_j. We describe the step-by-step sequence followed so as to process the task and return the results. In the implementation described below we considered Android-based devices

and thus we have utilized the Android SDK[10].

For every task t_j, the Master Server spans the task to a set of *map* tasks that need to be forwarded to the human workers w_k. Since these tasks are geo-located only the workers that reside close to the specific selection need to be selected by the Master Server to provide information. However, in order to avoid tracking the users we follow a different policy. We forward the task to all the workers and the tasks are locally filtered at the mobile devices if their location is far from the location of the task t_j.

We use Push Notifications services to initiate the communication with the human workers, to be able to send the Map task to the users without being restricted by their connection (WiFi, 3G, etc). Such services exist in all major mobile operator systems and allow users to register for message delivery when they are online through a connection server.

In order to be able to receive map tasks, each user first needs to login to our system so that the Master server will be aware of the user. At the same time the user also registers in the push notification service to retrieve its unique id. During normal operation the Crowdsourcing applications runs in the background (Figure 3a).

When the Master Server retrieves a new task t_j from the requester, it delivers a push notification to the user devices with the task, through the Push Notification service. Once the device receives the notification it examines whether the user current location is close to the location of the task so as to alert the user (Figure 3b). Next, if the user selects the notification on his mobile device the Crowdsourcing application is triggered and the task will be displayed in the user screen to process the task (Figure 3c).

Finally, the responses for each *map* task are forwarded to the Master Server that initiates the *reduce* phase to aggregate the answers. The reduce phase is performed through Majority Voting. Hence the Master Server identifies the response $response_{jk}$ for task t_k with the maximum amount of answers from all users w_j and forwards the response that represents the cause of the event to the system.

Crowd Feedback. The response retrieved by the crowdsourcing component enables the system to determine whether the irregular readings derive from an unexpected event (*e.g., roadworks*) or if the sensor is indeed faulty when most of the workers answer "None of the above".

5. Evaluation

We have evaluated our proposals on our local cluster consisting of 4 VMs. Each VM had two CPU processors attached and 3,096 MB of RAM. All VMs were connected to the same LAN and their clocks were synchronized using the NTP protocol. The frameworks we used were the following: Storm 0.8.2, Esper 5.1 and MongoDB 2.6.5.

[10]Android platform: http://www.android.com/

Parameter	Value
$job\_periodicity$	24 hours
$stream\_threshold$	10 minutes
$time\_window$	1 hour

Table 1. Basic Configuration Parameters

In Table 1, you can see the values of the basic configuration parameters described in Section 4.2. For the experiments, we used SCATS data from the period of April and May of 2014. The distance threshold used for the neighbouring sensors computation was set to 250 meters. Data from April were used in order to calculate the historical correlations, cross-correlations as well as the ARIMA models. On the other hand, data from May were used for different experimental runs (see below).

For the Pearson Correlation method we have stored the historical correlations of the neighbour-pairs in the MongoDB component. 7116 neighbour pairs were identified under the distance threshold from a set of 900 SCATS sensors. The correlation value ranged from almost perfect correlation, for sensors of the same junction under different lane, to no correlation at all for more distant sensors. Negative correlation values between nearby sensors were also observed. This could be explained by the opposite direction of the lane the sensors are responsible for. In Figure 4, the correlation matrix for a set of 30 nearby sensors is presented. As expected, clusters are formed by adjacent sensors that are highly correlated. Thus, it is reasonable to argue that when the expected correlation is not observed there might be a problem with the sensor. For the Cross-Correlation method apart from storing the correlation value itself we have also stored the time lag that maximizes the pair-wise sensor correlation. The time lag range was set to a maximum of 10 minutes since the sensors are quite close to each other and larger time lags are unlikely to significantly favor the correlation value. In addition, the larger the time lag range is, the more computationally demanding the method will be. As it was expected, in most cases the highest cross-correlation was observed with no time lag at all, since most sensor pairs are responsible for different lanes of the same highway junction. However, for more distant sensor-pairs responsible for different highway junctions, small time lags gave a boost on their correlation value. One way to understand this is because vehicles require a short time to reach consecutive junctions. In addition, this behaviour could be also explained by the operation of traffic lights that transfer the traffic from junction to junction on fixed time intervals. Figure 5 depicts the distribution of the optimal time

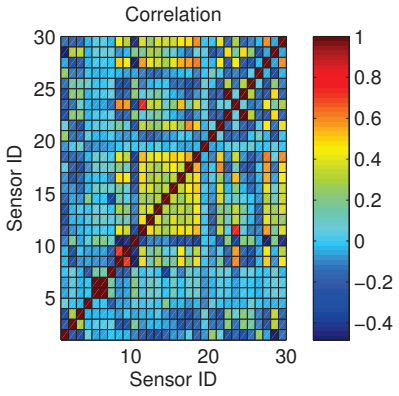


Figure 4. The correlation matrix of a set of 30 neighbors

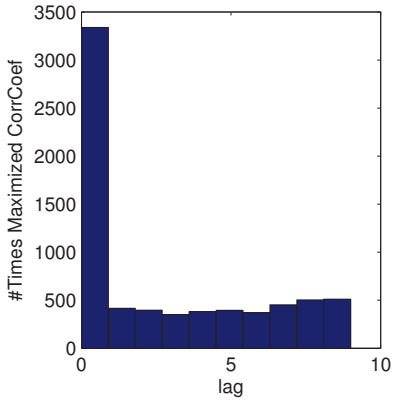


Figure 5. Distribution of the optimal time lag value over all the neighbour-pairs

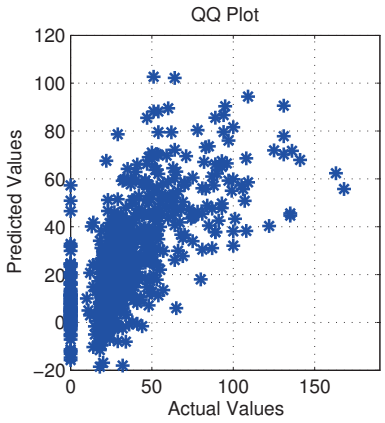


Figure 6. The predicted and the actual degree of saturation values over the validation dataset

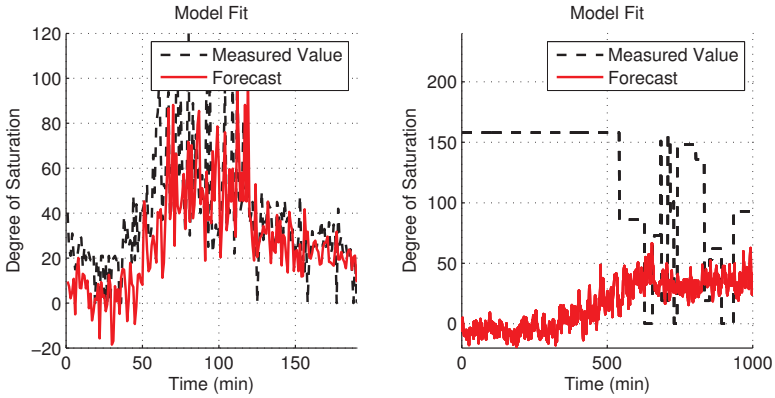


Figure 7. On the left there is a sensor whose values agree well with our forecast. On the right there is a noisy sensor whose values diverge significantly from the predicted and it is considered as faulty

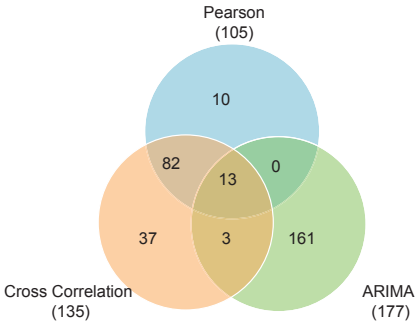


Figure 8. Venn diagram for the three proposed methods

lag value over a sample of neighbor pairs.

In terms of the multivariate ARIMA method, a different model was fitted to each sensor using as features all the neighbor sensors. The performance of all the models was aggregated and measured in terms of Mean Absolute Error (MAE), Root Mean Squared Error(RMSE) and Correlation Coefficient (CC) using a validation dataset. The results follow on Table 2 with a low MAE value of 16.18 indicating a decent fit.

Figure 6 shows the forecasting performance of the ARIMA model on the validation dataset and Figure 7 gives an example on forecasting two different sensors. Sensors such as the one presented in Figure 7 (left) will be considered as non-faulty since the deviation between the observed measurements and the expectation is not significant. On the other hand, sensors such as the one in Figure 7 (right), given that it reports maximum values for a long period of time, it is likely that it is faulty. These sensors are flagged by our system for further manual evaluation or inspection from the traffic operators.

The three methods were compared in terms of the number of faulty sensors they identify. In addition, since the Correlation and the ARIMA approaches focus at a very different aspect of the same problem we measured the overlap between their results. Figure 8 displays the Venn diagram of the results obtained over the period of one day during May of 2014. As it was expected, the results of Pearson Correlation and Cross-Correlation are highly overlapping since for many sensors the optimal time lag is zero. On the other hand, the ARIMA method identified different sensors as erroneous suggesting that the methods are complementary to each other. Interestingly enough, 13 sensors were identified as erroneous from all methods indicating that sensors operate in an unexpected way in many settings and are more likely to be faulty.

Metric	Result
CC	0.68
MAE	16.8
RMSE	23.18

Table 2. The measurements that indicate the performance of the multivariate ARIMA model

6. Related Work

Traffic monitoring has been a field of great interest in the scientific community (Biem et al., 2010), (Patroumpas & Sellis, 2012). These works detect unusual events based on pre-defined *rules* so any updates to the traffic conditions overtime is not taken into account. In contrast, our proposal exploits historical data for updating the expected sensor correlations and detects events only when the real-time conditions deviate significantly from the expected. Authors in (Ma et al., 2013) propose a novel city transportation application that enables sharing of taxi rides in a large city. Their goal was to develop an application that is beneficial for both the citizens and the taxi drivers.

There has been significant work in traffic monitoring in the use-case of Dublin. (Artikis et al., 2014) proposed a traffic management system, based on heterogeneous data, which used Crowdsourcing in order to resolve conflicting sensors reports. (Zygouras et al., 2015) focused on monitoring the traffic conditions of the city by considering the metrics reported from sensors mounted on top of public buses. While (Liebig et al., 2014a) and (Liebig et al., 2014b) perform individual trip planning that considers future traffic hazards in routing. Furthermore, their approach estimates the expected traffic flow in areas with low sensor coverage.

Anomaly detection methods have been widely applied for mining data streams including techniques such as data clustering (Guo et al., 2009), principal component analysis (PCA) (Lakhina et al., 2004), wavelet transform (Novakov et al., 2013) and many others. Some detection methods follow a time series analysis perspective and focus on forecasting methods such as ARIMA (Zare Moayedi & Masnadi-Shirazi, 2008; Fujimaki et al.). ARIMA models are a wide family of analysis and forecasting models that are used widely in forecasting urban traffic time series data (Lee & Fambro, 1999; Williams et al., 1998). This makes ARIMA models suitable for our scenario. (Niennattrakul et al., 2010), used distance-based outlier detection techniques, reducing the size of the original database, in order to efficiently identify outliers in massive streaming datasets. (Schettlinger et al., 2010) proposed an online time series filter, using repeated median regression, which is able to smooth the data and keep intact the signal's trend. (Branch et al., 2013) developed a distributed and in-network model in order to detect outliers on network exchanges among neighboring nodes. (Fried et al., 2015) proposed a Bayesian approach to model time series of counts, using Metropolis-Hastings algorithm in order to estimated the parameters of the model.

7. Conclusions

In this paper we presented an efficient approach for resolving whether irregular sensor measurements are due to faulty sensors or unexpected traffic. Our approach exploits sensors' past measurements and the crowd's wisdom for decision making. We implemented our proposals using the Lambda-Architecture for processing real-time and historical data, and an Android application for extracting answers from the human crowd. We applied three different outlier detection techniques that identified complementary set of faulty sensors. Finally, our detailed experimental evaluation indicates that our approach can effectively resolve the source of irregular measurements in real-time.

Acknowledgments

This research has been co-financed by the European Union (European Social Fund ESF) and Greek national funds through the Operational Program Education and Lifelong Learning of the National Strategic Reference Framework (NSRF) - Research Funding Program: Thalis-DISFER, Thalis-CompGeom, Aristeia-MMD Investing in knowledge society through the European Social Fund, the FP7 INSIGHT project and the ERC IDEAS NGHCS project.

References

Artikis, A., Weidlich, M., Schnitzler, F., Boutsis, I., Liebig, T., Piatkowski, N., Bockermann, C., Morik, K., Kalogeraki, V., Marecek, J., Gal, A., Mannor, S., Kinane, D., and Gunopulos, D. Heterogeneous Stream Processing and Crowdsourcing for Urban Traffic Management. *in Proc. 17th International Conference on Extending Database Technology (EDBT), Athens, Greece, March 24-28, pp. 712-723*, 2014.

Biem, Alain, Bouillet, Eric, Feng, Hanhua, Ranganathan, Anand, Riabov, Anton, Verscheure, Olivier, Koutsopoulos, Haris, and Moran, Carlos. IBM Infosphere Streams for Scalable, Real-time, Intelligent Transportation Services. *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, 2010.

Bockermann, C. and Blom, H. The streams framework. *Technical Report 5, TU Dortmund University*, 2012.

Boutsis, Ioannis and Kalogeraki, Vana. On task assignment for real-time reliable crowdsourcing. In *ICDCS*, pp. 1–10, Madrid, Spain, June 2014.

Branch, JoelW., Giannella, Chris, Szymanski, Boleslaw, Wolff, Ran, and Kargupta, Hillol. In-network outlier detection in wireless sensor networks. *Knowledge and Information Systems*, 34(1):23–54, 2013. ISSN 0219-1377. doi: 10.1007/s10115-011-0474-5. URL http://dx.doi.org/10.1007/s10115-011-0474-5.

Dou, Adam, Kalogeraki, Vana, Gunopulos, Dimitrios, Mielikainen, Taneli, and Tuulos, Ville H. Misco: a mapreduce framework for mobile systems. In *PETRA*, June 2010.

Dou, Adam Ji, Kalogeraki, Vana, Gunopulos, Dimitrios, Mielikinen, Taneli, Tuulos, Ville, Foley, Sean, and Yu, Curtis. Data clustering on a network of mobile smartphones. In *SAINT*, pp. 118–127, Munich, Germany, July 2011.

Fried, Roland, Agueusop, Inoncent, Bornkamp, Bjrn, Fokianos, Konstantinos, Fruth, Jana, and Ickstadt, Katja. Retrospective bayesian outlier detection in ingarch series. *Statistics and Computing*, 25(2): 365–374, 2015. ISSN 0960-3174. doi: 10.1007/s11222-013-9437-x. URL http://dx.doi.org/10.1007/s11222-013-9437-x.

Fujimaki, Ryohei, Yairi, Takehisa, and Machida, Kazuo. An anomaly detection method for spacecraft using relevance vector. In *Learning, The Ninth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD*, pp. 785–790. Springer.

Guo, Feng, Yang, Yingzhen, and Duan, Lian. Anomaly detection by clustering in the network. In *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on*, pp. 1–4. IEEE, 2009.

Kakantousis, Theofilos, Boutsis, Ioannis, Kalogeraki, Vana, Gunopulos, Dimitrios, Gasparis, Giorgos, and Dou, Adam. Misco: A system for data analysis applications on networks of smartphones using mapreduce. In *MDM*, pp. 356–359, Bengaluru, India, July 2012. IEEE.

Kinane, D., Schnitzler, F., Mannor, S., Liebig, T., Morik, K., Marecek, J., Gorman, B., Zygouras, N., Katakis, Y., Kalogeraki, V., and Gunopulos, D. Intelligent synthesis and real-time response using massive streaming of heterogeneous data (insight) and its anticipated effect on intelligent transport systems (its) in dublin city, ireland. In *ITS*, Dresden, Germany, November 2014.

Lakhina, Anukool, Crovella, Mark, and Diot, Christiphe. Characterization of network-wide anomalies in traffic flows. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pp. 201–206. ACM, 2004.

Lee, Sangsoo and Fambro, Daniel B. Application of subset autoregressive integrated moving average model for short-term freeway traffic volume forecasting. *Transportation Research Record: Journal of the Transportation Research Board*, 1678(1):179–188, 1999.

Liebig, Thomas, Piatkowski, Nico, Bockermann, Christian, and Morik, Katharina. Predictive trip planning-smart routing in smart cities. In *EDBT/ICDT Workshops*, pp. 331–338, 2014a.

Liebig, Thomas, Piatkowski, Nico, Bockermann, Christian, and Morik, Katharina. Route planning with real-time traffic predictions. In *Proceedings of the 16th LWA Workshops: KDML, IR and FGWM, Aachen, Germany*, pp. 83–94, 2014b.

Ma, Shuo, Zheng, Yu, and Wolfson, Ouri. T-Share: A Large-Scale Dynamic Taxi Ridesharing Service. *ICDE*, 2013.

McCreadie, Richard, Macdonald, Craig, Ounis, Iadh, Osborne, Miles, and Petrovic, Sasa. Scalable Distributed Event Detection for Twitter. *BigData Conference: 543-549*, 2013.

Niennattrakul, V., Keogh, E., and Ratanamahatana, C.A. Data editing techniques to allow the application of distance-based outlier detection to streams. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pp. 947–952, Dec 2010. doi: 10.1109/ICDM.2010.56.

Novakov, Stevan, Lung, Chung-Horng, Lambadaris, Ioannis, and Seddigh, Nabil. Studies in applying pca and wavelet algorithms for network traffic anomaly detection. In *High Performance Switching and Routing (HPSR), 2013 IEEE 14th International Conference on*, pp. 185–190. IEEE, 2013.

Patroumpas, Kostas and Sellis, Timos. Event Processing and Real-time Monitoring over Streaming Traffic Data. *Web and Wireless Geographical Information Systems Lecture Notes in Computer Science Volume 7236, pp 116-133*, 2012.

Schettlinger, K., Fried, R., and Gather, U. Real-time signal processing by adaptive repeated median filters. *International Journal of Adaptive Control and Signal Processing*, 24(5):346–362, 2010. ISSN 1099-1115. doi: 10.1002/acs.1105. URL http://dx.doi.org/10.1002/acs.1105.

Williams, Billy M, Durvasula, Priya K, and Brown, Donald E. Urban freeway traffic flow prediction: application of seasonal autoregressive integrated moving average and exponential smoothing models. *Transporta-*

tion Research Record: Journal of the Transportation Research Board, 1644(1):132–141, 1998.

Yi, B.-K., Sidiropoulos, N.D., Johnson, T., Jagadish, H.V., Faloutsos, C., and Biliris, A. Online data mining for co-evolving time sequences. In *Data Engineering, 2000. Proceedings. 16th International Conference on*, pp. 13–22, 2000.

Zare Moayedi, H. and Masnadi-Shirazi, M.A. Arima model for network traffic prediction and anomaly detection. In *Information Technology, 2008. ITSim 2008. International Symposium on*, volume 4, pp. 1–6, Aug 2008.

Zygouras, Nikolas, Zacheilas, Nikos, Kalogeraki, Vana, Kinane, Dermot, and Gunopulos, Dimitrios. Insights on a Scalable and Dynamic Traffic Management System. *EDBT*, 2015.

Profiling users of the Vélo'v bike sharing system

Albrecht Zimmermann, Mehdi Kaytoue, Céline Robardet, Jean-François Boulicaut FIRSTNAME.NAME@INSA-LYON.FR
INSA-Lyon, CNRS, LIRIS UMR5205, F-69621, France

Marc Plantevit MPLANTEV@LIRIS.CNRS.FR
Université Lyon 1, CNRS, LIRIS UMR5205, F-69622, France

Abstract

Detecting and characterizing geographical areas that are attractive places for specific people, in specific contexts, is an important but challenging new problem. Mobility traces and their related circumstances can be modeled thanks to an augmented graph in which nodes denote geographic locations and edges are represented by a set of transactions that describe users' demographic information (e.g. age, gender, etc.) as well as the conditions of the movement (e.g. day/night, holiday, transportation mode, etc.). We propose to extract connected subgraphs that are related to some user profiles, and use it to understand the usages of the Vélo'v bike sharing system.

1. Introduction

The problem considered hereafter is how to detect and characterize geographical areas that are attractive places and routes for specific contexts. Such areas are frequently accessed together in certain conditions by users of similar profiles compared to all contexts and users. Starting from a relational database that gathers information on people movements – such as origin, destination, date and time of travel, means of transport, reasons for traveling, etc. – as well as demographic data, we adopt a graph-based representation that results from the aggregation of individual travels. In such a graph, the vertices are locations or points of interest (POI) and the edges stand for user's co-visitations. Travel information as well as user demographics are labels associated to the edges of the graph. Figure 1 (a) depicts an example of travels undertaken by users (denoted $u_1, \ldots, u_4$). For each user, we know her age and gender, the context of the move (day or night) and the set of movements, identified by a pair origin/destination, that occur in this context. Capital letters, from A to E, represent POI. This table can also be viewed as an edge-attributed graph where edges stand for movements and are labeled by the attribute values of the context. For instance, we have a directed edge (A, C) labeled by $(F, 20, Day)$ for the user u_1. Given a specific context, the edge-attributed graph can be transformed into an aggregate graph whose edges are weighted by the number of attributed edges that hold for the context. Three examples of aggregated graphs are given in Figure 1 (b),(c) and (d). The weights of the aggregated graph can be seen as the support of the context in the graph.

The problem is thus to identify the contexts and sub-graphs that are specific to one another. By specific, we mean that a large proportion of the weight of each sub-graph edge mainly corresponds to users that satisfy the context. The adequacy of a context to an edge is assessed by a χ^2 test and some novel quality measures that makes it possible to identify the so-called *demographic and contextualized specific areas (DCSA)*. Two *DCSA* patterns are presented in Figure 1 (c) and (d) (in bold): The first one identifies a sub-graph that is traveled during the *day* , mainly by people with *age greater than* 45. In the second sub-graph, bold edges are very specific to *male* persons' behavior, whatever the travel time.

2. Travel patterns in the VÉLO'V system

VÉLO'V is the bicycle sharing and renting system run by the city of Lyon (France) and the company JCDecaux.[1] The VÉLO'V dataset contains movement data collected between Jan. 2011 and Dec. 2012. Each movement includes both bicycle stations and timestamps for departure and arrival, as well as some basic demographics about the user of the bike. We aggregated all movements a user performed between any two stations for the entire time period. Hence, the VÉLO'V stations are nodes in the graph (342 in total),

Proceedings of the 2^{nd} International Workshop on Mining Urban Data, Lille, France, 2015.

[1] http://www.velov.grandlyon.com/

User	Gender	Age	Time	Travels
u_1	F	20	Day	(A,C),(B,A), (C,B)
u_1	F	20	Night	(D,C),(D,E),(E,A), (E,D)
u_2	M	23	Day	(A,B),(B,C),(C,A), (C,B)
u_2	M	23	Night	(A,B),(B,C),(C,B) (C,D),(D,C),(D,E), (E,D)
u_3	F	45	Day	(A,B),(B,C),(C,D), (D,A),(D,E),(E,D)
u_3	F	45	Night	(B,D),(D,B)
u_4	M	50	Day	(A,B),(B,C),(C,B), (C,D),(D,A),(D,E), (E,D)
u_4	M	50	Night	(A,C),(C,A)

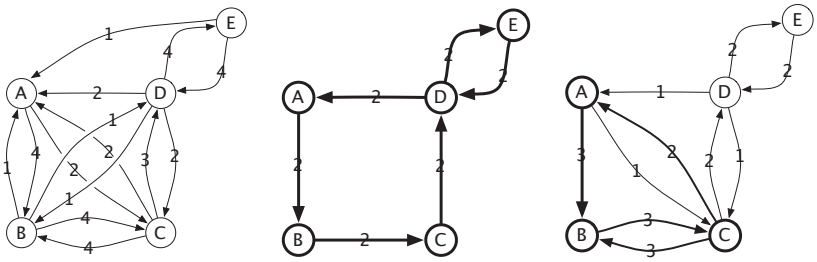
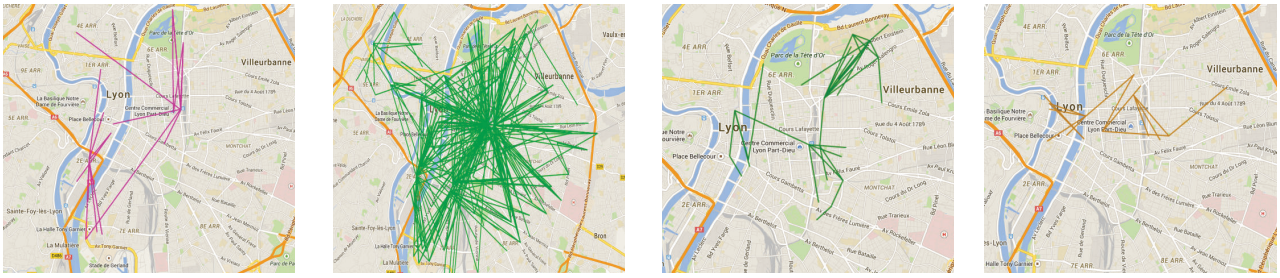


Figure 1. Example of contextualized trajectories: (a) Transactional view; (b) Aggregate graph w.r.t the most general context $\star = (Age \in [20, 50],\ Gender \in \{F, M\},\ Time \in \{Day,\ Night\})$; (c) Aggregate graph w.r.t. context $(Age \in [45, 50], Time = Day)$; (d) Aggregate graph w.r.t. context $(Gender = M)$;



(a) YoB ≥ 1968, ZIP $= 42400$ **(b)**YoB ≥ 1962, CAT $=$ OURA **(c)**YoB ≥ 1980, TYP $=$ std **(d)**YoB ≥ 1992, ZIP $= 69003$

Figure 2. DCSA discovered from VÉLO'V

and edges link two stations if a VÉLO'V customer checked out a bicycle at the first station and returned it at the second one. We treat the edges as undirected. Customers are described by nominal attributes such as gender, type of membership card, ZIP code and country of residence, as well as a numerical one: year of birth. There are a total $50,601$ customers. The data set comprises around 2 million contextualized edges.

Given the characteristics of different users, we aim to identify populations that use the rental bicycles in a particular manner. Figure 2 shows 4 different *DCSA* from VÉLO'V. Pattern (a) identifies people born after 1968, living in a city (Saint Chamond) located approximately 50km from Lyon. It is therefore not surprising that the edges involve the two main train stations of Lyon: Perrache (south-west) and Part-Dieu (center), from which users take bicycles to areas that are not easily reached by metro or tram, such as the 1^{st} and 4^{th} arrondissements. The edges of pattern (b) radiate from all of Lyon's train stations, not only the major ones. Its description refers to holders of a regional train subscription, and the pattern notably involves 200 nodes, almost 60% of the stations. It is very likely that this pattern identifies commuters. Pattern (c) involves users born after 1980 and we can identify three main areas: the scientific campus in the north, the Presqu'ile and its pubs, and the shopping area in the city center. It is notable that several of the long edges correspond to very comfortable cycling routes. Pattern (d) does not seem to be very exciting: young people that live in the 3^{rd} arrondissement use VÉLO'V bicycles to move around in their area. At a second glance, however, this is the closest that we will come to a ground truth in real-world data: the ZIP code of users aligns with the area where the bicycles are used!

3. Conclusion

The problem of finding *DCSA* in edge-attributed graphs has many applications in location based social networks and recommendation systems. It allows to find connected components highly characteristic of a given category of users. The proposed inductive approach is solved thanks to an efficient data mining algorithm that avoids materializing all contexts/induced-graph pairs and benefits from pruning and upper bound computations techniques. Its use for the analysis of the bicycle sharing system Vélo'v demonstrates its capabilities to provide new valuable insights. [2]

[2]This work has been partially supported by the projects GRAISearch (FP7-PEOPLE-2013-IAPP) and VEL'INNOV (ANR INOV 2012).

Accessibility by public transport predicts residential real estate prices: a case study in Helsinki region

Indrė Žliobaitė INDRE.ZLIOBAITE@AALTO.FI
Michael Mathioudakis MICHAEL.MATHIOUDAKIS@HIIT.FI
Tuukka Lehtiniemi TUUKKA.LEHTINIEMI@HIIT.FI
Pekka Parviainen PEKKA.PARVIAINEN@AALTO.FI
Tomi Janhunen TOMI.JANHUNEN@AALTO.FI
Helsinki Institute for Information Technology (HIIT), Espoo, FINLAND

Aalto University, Dept. of Computer Science, Espoo, FINLAND

Abstract

This pilot study investigates how considering accessibility could help to model prices of residential real estate more accurately. We introduce two novelties from the price modeling point of view (1) defining accessibility as travel time by public transport, in addition to geographic distance, and (2) considering dynamic points of interest from check-ins into social networks, in addition to fixed location community centers. Our case study focuses on the Helsinki region. We model price per square meter as a linear function of apartment characteristics, and characteristics of the neighborhood, including accessibility by public transport and social activities. The resulting models show good predictive performance, as compared to baselines not taking accessibility into account. We discover that apartment price relates to the geographical distance from the city center, but accessibility by public transport to local centers of interest is more informative than just the geographical distance to those centers.

1. Introduction

Modeling real estate prices has long been of interest to researchers and practitioners, and it is employed for various purposes related to investment, lending or taxation. Arguably all city residents, even non-specialists, intuitively understand that the price of a residential apartment positively relates to the size of the apartment, and negatively relates to the distance to the city center. Professional real estate price models include many more features of apartments and environment, such as age, construction type, floor, or population characteristics in the neighborhood.

Residential real estate prices are typically modeled using so called *hedonic models* (Case & Quigley, 1991; Sirmans et al., 2005), where the price of a house is assumed to be affected by the structural characteristics of the house itself, characteristics of the neighborhood, and environmental characteristics. While in real estate domain research mainly focuses on identifying factors that impact pricing, in machine learning and data mining research real estate price modeling mainly focuses on developing sophisticated predictive models beyond linear regression (Chopra et al., 2007; Fu et al., 2014).

A literature review on hedonic pricing models (Bartholomew & Ewing, 2011) finds the structural characteristics typically include the age and the size of the house, the number of bedrooms, and the presence of different amenities such as a garage. The effect of the location of the house on housing prices is often captured by physical proximity to a central business district (CBD) or a regional center. The literature review finds evidence of an inverse relationship between pricing and distance to CBD in studies on various cities around the world. Another access-related characteristic often used in hedonic models is the proximity of the house to a transit station, measured in air distance or walking distance. This attribute is used to capture the effect transit has on relative accessibility of a CBD or a regional center. Here the results are more mixed, with the majority of studies suggesting pricing premiums for housing located near to a transit station, and a higher premium for transit stations that provide a higher degree of relative proximity to a CBD.

Proceedings of the 2^{nd} International Workshop on Mining Urban Data, Lille, France, 2015.

The era of big data provides access to new data sources, such as public transport, traffic and social mobility data, that potentially relate to real estate prices (at least intuitively we know that people consider mobility, and social factors when buying an apartment). Integrating such data could help to model residential real estate prices more precisely, and, as a result, better understand urban mobility patterns and activities. Such models can contribute to managing, coordinating and long term planning of mobility, and overall development of modern smart cities.

Our pilot study investigates to what extent accessibility of a neighborhood relates to residential real estate prices. This case study focuses on the Helsinki region. We model price per square meter as a linear function of apartment characteristics, and characteristics of the neighborhood, including accessibility by public transport and social activities. Our main hypothesis is that prices are more related to travel times than travel distances, and local centers of activities than the city center. The resulting models show good predictive performance, as compared to baselines not taking accessibility into account. We discover that an apartment price relates to the geographical distance from the city center, but accessibility by public transport to local centers of interest is more informative than just the geographical distance to those centers.

Our study introduces two conceptual novelties in modeling prices of residential real estate: (1) to measure accessibility, we consider travel times in addition to distances, and (2) we consider dynamic local points of interest, defined by 4square[1] check-ins (people posting their location and activity on a social network), in addition to community centers at fixed locations.

The remainder of the paper is organized as follows. Section 2 describes data acquisition and feature engineering. Section 3 presents the results of the experimental case study, and Section 4 concludes the study.

2. Data acquisition and feature engineering

Our dataset consists of three parts: real estate data describing characteristics of the apartments, location data describing points of interest and community centers, and accessibility data describing point-to-point distances and travel times. We make our dataset publicly available[2] for research.

[1] http://foursquare.com

[2] http://www.zliobaite.com/datahel.zip

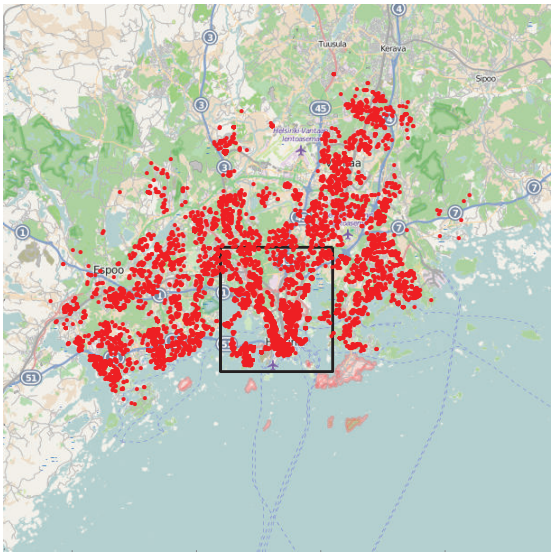


Figure 1. Location of apartments in the dataset. The black rectangle indicates the area from which point of interest data is collected.

2.1. Real estate data

The sales price data comes from a Finnish web portal Oikotie[3], which is the most popular marketplace for residential real estates sales and rental. Our dataset consists of apartments in the capital region (Helsinki, Espoo, Vantaa and Kauniainen municipalities) advertised for sales on October 24, 2014. The pricing data is based on sales ads, as sales transaction prices are not available for the public. We exclude apartments that do not provide a street address (hence no coordinates), and for which size is not available. Moreover, we filter out very large apartments (size more than 300 m2), very old apartments (built earlier than 1850), far away apartments (distance to metro more than 20 km), extremely cheap (price pr square meter less than 1200 eur) and extremely expensive apartments (price per square meter more than 12000 eur), because we aim at focusing on modeling prices of mainstream apartments and avoiding extreme outliers. After filtering our dataset includes 8337 apartments. Figure 1 plots all the apartment locations.

2.2. Location data

We consider two types of location data: fixed location, and dynamic points of interest. Fixed location data includes the city center, for which the Stockmann department store is used as a proxy (coordinates found by hand via Google maps), and local community centers, approximated by H&M shop (a chain of clothing shops) locations in Helsinki region (also found by hand from Google

[3] http://www.oikotie.fi/

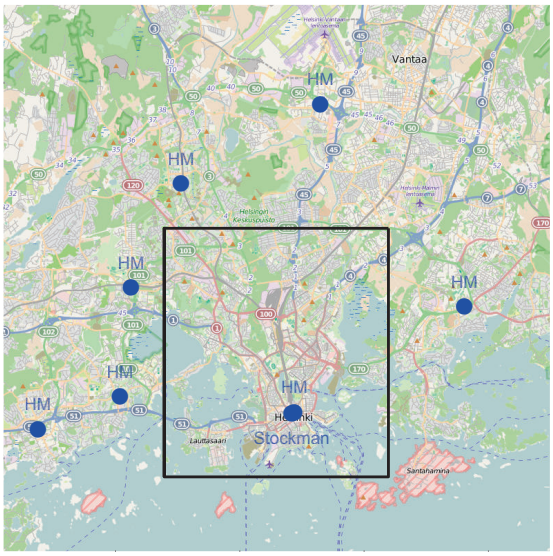


Figure 2. Fixed locations: community centers (H&M) and city center (Stockmann). The black rectangular indicates the area from which point of interest data is collected.

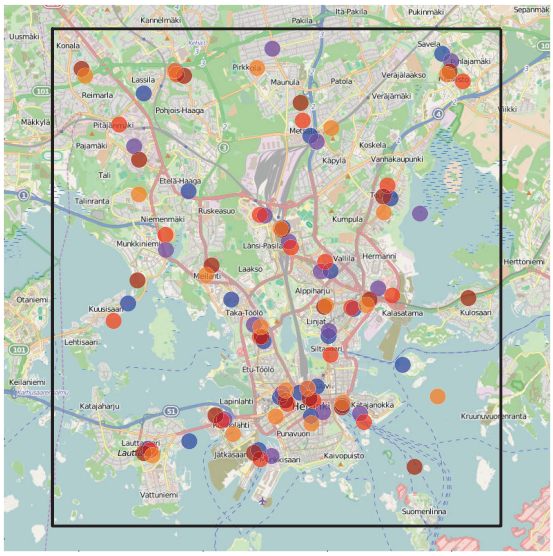


Figure 3. Dynamic points of interest from 4square: blue 2:00-6:00, violet 6:00-10:00, red 10:00-14:00, brown 14:00-18:00, orange 18:00-22:00.

maps). Stockmann is a well-known location in the centre of Helsinki. H&M shops are typically present in larger shopping malls. Shopping malls are local centers of attraction. We hope that H&M serves as a proxy for local centers in the neighborhoods. Figure 2 plots the community centers and the city center location.

Dynamic points of interest are obtained from an existing dataset of 4square check-ins (Le Falher et al., 2015). Each check-in in the dataset corresponds to one user's visit to one venue (restaurant, cafeteria, store, etc) with known geographic location, at a particular time. The data cover user activity between March and July 2014 in the inner Helsinki city. To extract points of interest, we perform k-means clustering on the geographic locations of check-ins, using $k = 20$. Each of the k centroids identified defines one point of interest. Note that we extract points of interest both on top of all check-ins contained in the dataset, regardless of the the time of the day they occur, as well as separately for check-ins that occur at separate time intervals in the day (five 4-hour intervals from 2am to 10pm). Figure 3 plots the points of interest for each time interval.

2.3. Accessibility features

Accessibility data connects apartments with point of interest. We consider two types of accessibility features: air distance from an apartment to the location of a point of interest, and travel time by public transport from an apartment to the point of interest (including walking time).

Air distance is measured in kilometers from coordinate of the apartment to coordinate of the point of interest, as

$$\begin{aligned} D &= R_e \cdot \arccos(s_1 + s_2), \text{where} \\ s_1 &= \cos(lat_1) * \cos(lat_2) * \cos(lon_2 - lon_1), \\ s_2 &= \sin(lat_1) * \sin(lat_2), \end{aligned}$$

where R_e is the radius of Earth (set to $R_e = 6371$km), (lat_1, lon_1) are the coordinates of the apartment, and (lat_2, lon_2) are the coordinates of the point of interest.

Travel time by public transport between two coordinates is measured using a freely available tool Reititin[4], developed by BusFaster Ltd and researchers at University of Helsinki. We use the default settings.

In addition to accessibility between apartments and points of interest we also include the distance from an apartment to the nearest metro station. The address of Metro stations is listed on Helsinki Metro's website[5] and their geographic coordinates are collected via manual queries to the Google Maps API[6]. Note that in the Helsinki region metro runs only to the eastern part of the city, therefore, we do not necessarily expect a regular behavior from this feature. A regular behavior would be a higher price if there is a metro stop nearby.

[4] http://blogs.helsinki.fi/saavutettavuus/tyokaluja/metropaccess-reititin/

[5] http://www.hel.fi/hki/hkl/en/HKL+Metro

[6] https://developers.google.com/maps/

Table 1. Input features (predictors).

Feature	Description	Units
size	apartment size	m2
year	year built	-
fyear	function of year	-
east	easterness	km
north	northerness	km
dmetro	distance to nearest metro	km
ametro	metro within 1 km (0,1)	-
dstock	Stockmann distance	km
tstock	Stockmann travel time	min
dhm	distance to H&M	km
thm	travel time to H&M	min
d4sq	min distance to 4square	km
t4sq	min travel time to 4square	min
d4sq1	distance to center 2-6:00	km
d4sq2	distance to center 6-10:00	km
d4sq3	distance to center 10-14:00	km
d4sq4	distance to center 14-18:00	km
d4sq5	distance to center 18-22:00	km
t4sq1	travel time to center 2-6:00	min
t4sq2	travel time to center 6-10:00	min
t4sq3	travel time to center 10-14:00	min
t4sq4	travel time to center 14-18:00	min
t4sq5	travel time to center 18-22:00	min

2.4. Final set of predictors

Altogether we consider 23 input features (predictors) for modeling apartment prices. The features are listed in Table 1.

Feature *fyear* is a non-linear transformation of the construction year, which is based on observation that apartments built around 1970 are the least valuable, because a major pipe renovation is due in about 50 years from initial construction. Pipe renovation is done for the whole house at once, and brings substantial expenses for the apartment owners. We define the derived feature as $fyear = (year - 1970)^2$, which puts very new and very old apartments together, while apartments that are around 50 years old are put on the opposite end.

Easternness and northerness features try to capture another peculiarity of the Helsinki region, where apartments in the east are on average considered cheaper. South apartments may be considered more expensive due to proximity to the sea.

Figure 4 plots the values of selected input features against the target variable price per square meter for visual inspection. General tendencies are consistent with common intuition. The smaller the apartment, the higher the price per m2. Apartments close to metro are more expensive. The cheapest apartments have been constructed around year 1970. The most expensive apartments are in the city center, and apartments near to the local community centers or points of interest are more expensive.

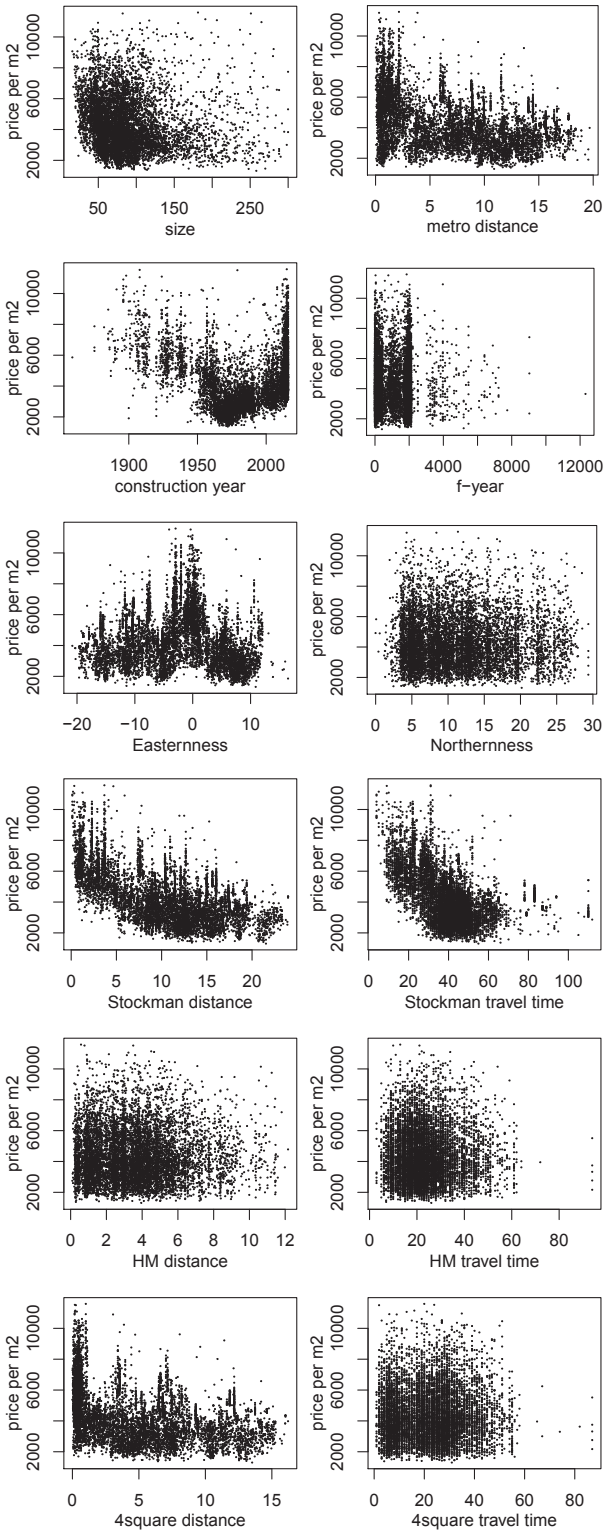


Figure 4. Input features against the target variable.

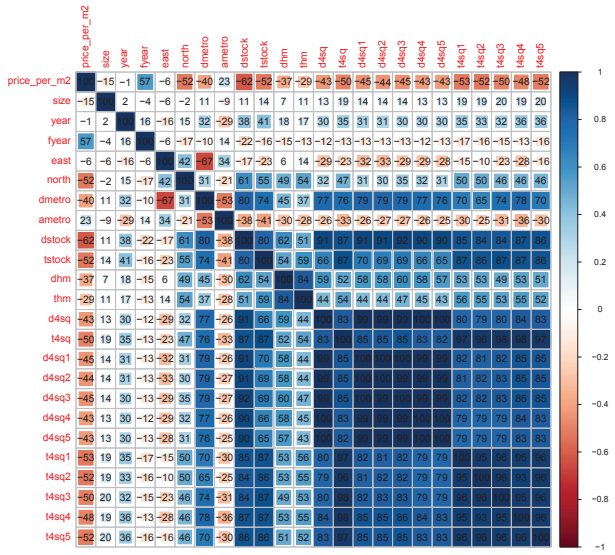


Figure 5. Correlations of features (a number indicates the Pearson correlation coefficient $\times 100$.

Finally, Figure 5 plots correlations across all the features. We see that the location features are strongly correlated with each other, therefore, many may be redundant. Nevertheless, some of those could potentially be expected to be more informative than others, therefore, we consider them all. We can also see than most of the location features are negatively correlated with community centers and dynamic points of interest. We already have seen similar tendencies in the scatterplots. This behavior is along with a common intuition that apartments near points of interest should be more expensive.

The correlation and scatter plots analyzed features one-by-one. In the next section we will consider predictive models that use sets of features for modeling apartment prices.

3. Case study

The goal of this pilot case study is to investigate whether accessibility information helps to model real estate prices, as compared to using only geographical location information. In addition, we investigate informativeness of dynamic points of interest (derived from social networks) as opposed to stationary fixed points of interest.

3.1. Experimental protocol

We model price per square meter. An alternative would be to model the total price. We choose the former as the target variable, because price per square meter is easier to interpret and compare across neighborhoods.

We limit our analysis to linear regression, which is easily interpretable. Note, however, that some of the features are expressed as non-linear functions of simpler features (e.g. *fyear* is a non-linear function of a building's age, as explained above). The ordinary least squares procedure (the standard implementation in R) is used for estimating the model parameters.

For assessing the performance we use two common accuracy measures: coefficient of determination (R2) and mean absolute error (MAE). Coefficient of determination is a relative accuracy measure, where 1 means the best possible performance, and 0 means the performance is equivalent to random. Mean absolute error indicates error in the units of the target variable, 0 is an ideal performance, the higher the MAE, the worse the performance.

We report R2 and MAE measured on the whole dataset used for model fitting (fit) and via 10 fold cross-validation (cv), which iteratively fits a model on 90% of the data, and tests on the remaining part. Cross-validation scores provide an indication of how models would generalize to unseen data.

3.2. Performance of base models

Base models do not use any accessibility information, and use only very basic location information. The first base model (Size-year) does not use location at all, and is based only on size of the apartment and its construction year (*fyear*). The second model in addition uses basic location information, encoded as raw geographical coordinates, centered in the old town of the city.

The resulting models for price per square meter are:

$$price \quad = \quad 3722 - 4.97 \times size + 0.91 \times fyear,$$

and

$$\begin{aligned} price \quad &= \quad 5643 - 5.14 \times size + 0.78 \times fyear + \\ &+ \quad 38.9 \times east - 147.7 \times north. \end{aligned}$$

The models are consistent with common intuition: the larger the apartment, the cheaper the price per square meter; older or newer apartments with respect to 1970 construction year are more expensive; the further to the north from the sea and the city center, the cheaper. Easterness has a positive effect, which is somewhat inconsistent with a common intuition that cheaper neighborhoods are in the east. However, this can be explained by the range of data (see Figure 4). Data extends further to the west than to the east, therefore, western apartments are on average further from the center, and thus cheaper.

Table 2 reports predictive accuracies of the base models. We can make two observations. First, Size-year-location model already performs quite well with the cross-validation

Table 2. Base models for predicting price per m2. R2 - coefficient of determination (the higher, the better), MAE - mean absolute error (the smaller the better).

Model	R2 fit	MAE fit	R2 test	MAE test
Size-year	0.34	1062	0.33	1063
Size-year-location	0.56	836	0.55	837

Table 3. Accessibility for predicting price per m2. Size-year-location model with one additional accessibility feature at a time.

Add feature	R2 fit	MAE fit	R2 test	MAE test
Metro distance	0.58	811	0.58	813
Metro access	0.56	835	0.55	836
H&M distance	0.56	837	0.56	838
H&M travel time	0.56	831	0.56	832
Stockmann distance	0.61	781	0.61	782
Stockmann t. t.	0.58	811	0.58	812
4square dist. all	0.58	813	0.58	814
4square t. t. all	0.58	804	0.58	805
4square dist. peak	0.59	807	0.59	809
4square t. t. peak	0.59	799	0.59	800

R2 result 0.55. Second, the fit and the cross-validation performance differs only a little, which suggests that there is no notable overfitting, and the model could use more informative input features.

3.3. Predictive power of accessibility

Next we test whether adding accessibility information helps to predict more accurately. We test accessibility to the city center (Stockmann), local centers (HM), and dynamic centers of interest (4square check-ins) overall and at morning peak times (from 6:00 to 10:00). We compare informativeness of using air distance as a feature to using the total travel time by public transport.

We use the base model Size-year-location as a starting point, add one feature at a time to it, and measure the accuracy. Table 3 reports the results.

From the resulting accuracies we can see that accessibility has some predictive power, as in all cases the predictive performance improves as compared to the base model. The results indicate that the distance to the city center (Stockmann) is more informative than the travel time by public transport. However, accessibility to the local centers (fixed centers H&M and dynamic centers 4square) by public transport is more informative than just the air distance to those centers. In other words, it seems that an apartment price relates to the overall geographical location, but *accessibility* to local centers of interest is more important than just the geographical distance to those centers. This is an interesting finding for exploring in detail in future studies.

We report selected models. Metro distance is intuitive - the closer to metro, the more expensive is the apartment:

$$\begin{aligned} price &= 5300 - 3.94 \times size + 0.77 \times fyear - \\ &- 62.2 \times east - 50.8 \times north - 158.4 \times metro. \end{aligned}$$

Adding metro distance shrinks other coefficients, which suggests that earlier this feature was indirectly captured. More importantly, adding metro distance changes the direction of the easternness coefficient from positive to negative. Now it is more intuitive keeping in mind peculiarities of Helsinki residential neighborhoods, where overall the east is considered cheaper than the west.

Stockmann distance is as well intuitive - the closer to the center, the more expensive is the apartment:

$$\begin{aligned} price &= 5698 - 3.78 \times size + 0.72 \times fyear + \\ &- 3.3 \times east - 59.8 \times north - 117.8 \times dstock. \end{aligned}$$

Shortest H&M travel time is intuitive - the shorter the travel time to the local center, the more expensive is the apartment:

$$\begin{aligned} price &= 5681 - 5.00 \times size + 0.78 \times fyear + \\ &+ 37.0 \times east - 139.6 \times north - 39.3 \times thm. \end{aligned}$$

Shortest 4square travel time is intuitive - the shorter the travel time to a center of interest, the more expensive is the apartment:

$$\begin{aligned} price &= 5659 - 3.55 \times size + 0.77 \times fyear + \\ &+ 13.5 \times east - 103.3 \times north - 31.4 \times t4sq. \end{aligned}$$

3.4. Final predictive model - everything together

Finally, we collect a set of promising features into one final model, and test its performance. The final model includes the base model (Size-year-location), metro distance, Stockmann distance (a proxy for distance to the city center), and travel times to H&M and 4square (peak) local centers.

While the final feature selection is done after seeing the intermediate performance results, the fit accuracies have been very similar to those of cross-validation, therefore the risk of overfitting is not high.

The final model fitted on all the data is

$$\begin{aligned} price &= 5729 - 4.06 \times size + 0.71 \times fyear + \\ &+ 31.6 \times east - 94.5 \times north + \\ &+ 73.0 \times metro - 139.0 \times dstock + \\ &+ 21.6 \times thm - 12.5 \times t4sq2 \end{aligned}$$

We can see some interesting relations, reflecting peculiarities of the Helsinki region. First, the longer the metro

Table 4. Final model for predicting price per m2.

Model	R2 fit	MAE fit	R2 test	MAE test
Final model	0.63	752	0.62	753

distance, the higher the price, while one could expect the opposite. Our interpretation is that the metro distance captures what was not very successfully captured by the easternness. We observe that the coefficient of easternness shrinks when metro comes into the equation. In Helsinki metro runs only to the eastern suburbs, and these suburbs are considered less prestigious neighborhoods than the west, and, therefore, residential prices there are lower.

Table 4 reports the performance figures. The final model shows the best performance seen so far, and reasonably good accuracy in relative and absolute terms (testing R2 = 0.62).

4. Conclusion

We have experimentally explored several models for real estate prices in Helsinki region, focusing our analysis on accessibility by public transport and dynamic points of interest, obtained via check-ins into social networks. We have found that even a basic account for accessibility features helps to improve the accuracy of price estimates. We have discovered that an apartment price relates to the geographical distance from the city center, but accessibility by public transport to local centers of interest is more informative than just the geographical distance to those centers.

Integrating such data could help to model residential real estate prices more precisely, and, as a result, better understand urban mobility patterns and activities. Such models can contribute to managing, coordinating and long term planning of mobility, and overall development of modern smart cities.

Acknowledgments

The authors thank Antti Ukkonen for insightful discussions. Research leading to these results was partially supported by the Aalto University AEF research programme and the Academy of Finland grants 118653 (ALGODAN) and 251170 (Finnish Centre of Excellence in Computational Inference Research COIN). Maps are credited to ©OpenStreetMap contributors, for more information see http://www.openstreetmap.org/copyright.

References

Bartholomew, Keith and Ewing, Reid. Hedonic price effects of pedestrian and transit-oriented development. *Journal of Planning Literature*, 26(1):18–34, 2011.

Case, Bradford and Quigley, John M. The dynamics of real estate prices. *The Review of Economics and Statistics*, 73(1):50–58, 1991.

Chopra, Sumit, Thampy, Trivikraman, Leahy, John, Caplin, Andrew, and LeCun, Yann. Discovering the hidden structure of house prices with a non-parametric latent manifold model. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pp. 173–182, 2007.

Fu, Yanjie, Xiong, Hui, Ge, Yong, Yao, Zijun, Zheng, Yu, and Zhou, Zhi-Hua. Exploiting geographic dependencies for real estate appraisal: A mutual perspective of ranking and clustering. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pp. 1047–1056, 2014.

Le Falher, Geraud, Gionis, Aris, and Mathioudakis, Michael. Where is the Soho of Rome? : Measures and algorithms for finding similar neighborhoods in cities. In *The 9th International AAAI Conference on Web and Social Media*, ICWSM, 2015.

Sirmans, Stacy, Macpherson, David, and Zietz, Emily. The composition of hedonic pricing models. *Journal of Real Estate Literature*, 13(1):1–44, 2005.

Airvlc: An application for real-time forecasting urban air pollution

Lidia Contreras Ochando LICONOC@UPV.ES
Universitat Politècnica de València. Spain

Cristina I. Font Julián CRIFONJU@EI.UPV.ES
Universitat Politècnica de València. Spain

Francisco Contreras Ochando FRACONOC@GMAIL.COM
Universitat Politècnica de València. Spain

Cèsar Ferri CFERRI@DSIC.UPV.ES
DSIC. Universitat Politècnica de València. Spain

Abstract

This paper presents Airvlc, an application for producing real-time urban air pollution forecasts for the city of Valencia in Spain. Although many cities provide air quality data, in many cases, this information is presented with significant delays (three hours for the city of Valencia) and it is limited to the area where the measurement stations are located. The application employs regression models able to predict the levels of four different pollutants (CO, NO, PM2.5, NO2) in three different locations of the city. These models are trained using features that represent traffic intensity, persistence of pollutants and meteorological parameters such as wind speed and temperature. We compare different learning techniques to get the better performance in the prediction of pollutants. According to our experiments, ensembles of decision trees (Random Forest) outperforms the rest of methods in almost all of our tests. Airvlc incorporates the best regression models and, by a distance-weighted combination of the predictions, is able to generate a real-time pollution map of the city of Valencia. The application also includes a warning system for sending notifications to users when a nearby risk pollution concentration is detected.

Proceedings of the 2^{nd} International Workshop on Mining Urban Data, Lille, France, 2015.

1. Introduction

Air pollution can have important impact (short and long-term) on the health of people. For instance, urban air pollution increases the risk of suffering respiratory diseases such as pneumonia, or chronic, such as lung cancer or cardiovascular disease (World Health Organisation, 2015). A recent work (Wilker et al., 2015) relates long-term exposure to ambient air pollution to structural changes in the brain.The SOER 2015 report (The European Environment Agency , 2015), with data about the European Union countries' air quality in 2011, concludes that although the atmosphere in the continent has improved in the last decades, there are significant traces of the most harmful contaminants. In fact, in 2011, the report estimates that 430.000 Europeans died prematurely because of pollution.

Although some governments are introducing restriction policies that limit the use of vehicles (main source of pollution in most cases), only in Europe, important cities such as Paris, Naples, Moscow, Milan or Barcelona still report significant levels of urban pollution in 2015 (The European Environment Agency , 2015). In this context, it is important for citizens of urban agglomerations to reduce the exposition to urban air pollution as much as possible. This is especially relevant for high risk population such as: kids, elderly people, asthmatics or people suffering respiratory diseases.

In this work we present an application that predicts urban air pollution in real time by employing historical data. The application is based on the city of Valencia in Spain. This city can be considered a medium size urban agglomeration (around 1.000.000 inhabitants). The city provides an open data site containing real-time information about the city in different aspects such as traffic data, noise sensors, pollen

sensors... Although different sensors of urban pollution air are included in the site, this information needs to be carefully verified and it is published with a delay of three hours. This delay can represent a problem since risky high levels of pollutions are not detected in real-time. Additionally, the network of sensors is limited (six in the city of Valencia).

Considering these limitations, we have developed an application able to display in real-time foreseeable levels of pollution in a wide number of points of the city. The application is based on the predictions of regression models that are trained using features that represent traffic intensity, persistence of pollutants and meteorological parameters.

The paper is organised as follows. Section 2 details the process of data recollection of pollution particles and the factors that affect the generation, concentration or dispersion of these pollutants. Experiments in learning regression models for predicting the pollutant concentrations are included in Section 3. The Airvlc application is detailed in 4. Related works are discussed in Section 5. Finally, Section 6 closes the paper with a discussion of the main conclusions and some plans for future work.

2. Data collection

Different particles are associated with urban air pollution. In order to measure air contamination, pollutant parameters found in the lower levels of the troposphere are controlled. Air quality sensors measure concentrations of particles that have an anthropogenic origin and produce effects during or after the inhalation by humans. The historical pollution data for this work has been obtained from the open data web of the Generalitat Valenciana[1]. Following the recommendations of (The European Environment Agency , 2015), we concentrate on the following particles:

- **PM 2.5 (Suspended particles below 2.5 microns):** This parameter has been chosen because of its pollutant power. It is one of the most dangerous particles, since its size makes it almost unstoppable by the natural filters of the body. This fact means that the PM 2.5 are usually able to reach the pulmonary alveoli and in some cases, these particles are attached to these alveoli with a consequent reduction of lung capacity; in worst cases, the particles cross the alveolar membranes and reach the blood stream. Considering that PM 2.5 particles have its origin in anthropogenic activities (especially in the use of fuels in motor vehicles), it is not surprising that its atomic structure contains heavy metals, extremely toxic to the human body. Atmospheric conditions in the Mediterranean coast of Spain can influence the particle levels, due to lower rainfall and wind action with respect to other northern Europe countries, and the North African particles (Saharan dust), PM10 and PM2.5.

- **NO (Nitrogen monoxide):** Nitrogen monoxide is a highly unstable compound; it causes nitrogen dioxide by quickly reacting in the atmosphere. This instability makes the nitrogen monoxide a radical, namely, a high reactive power molecule, whose effects on the body are abnormal DNA, lipids and proteins. This kind of changes derives in the medium and long term as a greater chance of developing cancer.Its origin stems largely from vehicle engines.

- **NO2 (Nitrogen dioxide):** Nitrogen dioxide is not a directly generated pollutant, since its presence in the atmosphere is caused by the oxidation of nitrogen monoxide. In the presence of moisture, this compound results in nitric acid, and its inhalation, even in low concentrations, can cause lung tissue degradation, as well as can reduce the efficacy of the immune system, especially in children.

- **CO (Carbon monoxide):** Carbon monoxide is a primary pollutant. CO is toxic; it prevents oxygen transport by poisoning the blood, since it replaces the haemoglobin. People with cardiovascular and cerebrovascular problems could suffer heart attacks or strokes because of problems related to high concentrations of CO.

The distribution of air pollution is decisively influenced by climatic conditions. We have collected Climatological observations for the meteorological data of Valencia city from Meteorological Agency of the Government of Spain (AEMET)[2]. We consider the following parameters:

- **Temperature:** In an ordinary atmosphere situation, temperature decreases with altitude, favouring ascension of warmer (and less dense) air, and dragging contaminants upwards. In a situation of thermal inversion, a warmer layer of air is over the colder surface air and prevents the rise of this last (denser), so the contamination is confined and increases.

- **Humidity:** Humidity is a weather factor to be considered; in its presence, nitrogen dioxide derives in nitric acid, harmful to human health.

- **Wind speed:** Strong winds can disperse pollutants and transport them away from their emission point.

[1] http://www.cma.gva.es/cidam/emedio/atmosfera/jsp/historicos.jsp

[2] http://www.aemet.es/

- **Precipitations** Precipitations wash contaminants and can dissolve substances and gases.

The two main sources of pollution in developed countries are motor vehicles and industry. Vehicles release large amounts of nitrogen oxides, carbon oxides, hydrocarbons and particulates when burning gasoline and diesel. Therefore, we need to measure the level of traffic in the city in order to predict the air pollution. For this purpose, the City of Valencia provides a network of sensors (electromagnetic coils) that measure the intensity of traffic (Vehicles/hour) in the city. This data can be found in the open data site of the Valencia City Council[3].

3. Experiments

With all the selected parameters, we have built datasets aimed to predict the concentration of pollutants from the intensity of traffic and weather parameters. Concretely, we have collected data for a period of two years (2013 and 2014). Data was collected every 60 minutes, 24 hours a day during those two years. Although Valencia city has six stations for the detection and measurement of air pollution, three of them have not sufficient data for the analysed period and were discarded. In this way we collected data from these stations: *Molì*, *Avd Francia* and *Pista de Silla*. These three stations are located inside the urban agglomeration, and thus most of the pollutants measured in the sensors should be generated by urban activities (mainly traffic). For each one of these stations, we create a dataset with the level of the pollutants measured and parameters that can affect these measurements, we concentrate on traffic level (measured by electromagnetic coils), weather conditions. In order to measure the traffic related to each air pollution station, we average the traffic intensity of the closest six traffic measurament sensors. This is a simplification since, certainly, all the traffic of the city has effect on the measured level of all the stations in the city.

We can see a summary of the three datasets in Table 1. This table includes averages and standard deviation for the three stations of the pollutant particles measured and the intensity of traffic associated with each station. If we analyse traffic intensity, *Avd Francia* is the busiest station, while the other two have similar values. With regard to pollution levels *Pista de Silla* station presents the maximum levels for three parameters. The only exception is PM2.5. This behaviour can probably be associated with the specific location of the stations: While *Pista de Silla* station is located in a the central part of the city, and therefore more vulnerable to the overall city pollution, the other two are in the suburbs of the city where external air streams can reduce the levels of pollutants.

We first study the weekly evolution of pollutants in the three stations. Figure 1 shows the evolution of the average of the four parameters of pollution analysed and the average traffic intensity for *Molì* station depending on the day of the week. Figure 2 presents the same plot for *Avd Francia* station and Figure 3 corresponds to *Pista de Silla* station. In order to make the values comparable in the plot we normalise each parameter by the maximum value of that parameter. The level of pollutants and traffic reach the maximum levels during the working days of the week for the three stations (Friday seems to be the worst day). We can clearly see the dependency of the four parameters of pollution on the traffic intensity level. During the weekend days, the level of traffic drastically descends and associated with this reduction the levels of pollutants significantly drop. Again, the exception is PM2.5. This behaviour can be caused because these particles can be generated by all types of combustion activities (motor vehicles, power plants, wood burning, etc.) and certain industrial processes (US Environmental Protection Agency , 2015).

We have performed a similar analysis considering the evolution of pollutants, traffic intensity and meteorological variables during a day (humidity and wind). Figure 4 shows the evolution of the daily average of these parameters for *Molì* station depending on the hour of the day . Figure 5 corresponds to *Avd Francia* station and Figure 6 to *Pista de Silla* station. Again, we normalise each parameter by the maximum value of that parameter. If we observe traffic intensity, we can discover in all the three plots a similar behaviour, there are three peaks in traffic intensity corresponding to the hours where workers travel to their work places (around 9 am), lunch time (around 2 pm) and an evening period (around 8 pm). In the three stations the maximum of pollution parameters is found at the same period of the first peak in traffic intensity (around 9 am). In the second peak of traffic intensity (around 2 pm) the levels of pollutants does not follow the increase in traffic. In fact, after the maximum period around 9 am, pollutants decrease their levels until around 4 pm where they change the behaviour and start an increasing of the values. The second peak in pollutant values is found around 9 pm. Our intuition with respect to this behaviour is that wind disperses part of the pollutant in the most sunny hours. Valencia is in the Mediterranean coast and in this city it is easy to find (especially in summer) sea breezes. These kind of winds are created over bodies of water (usually sea or big lakes) near land due to differences in air pressure created by their different heat capacity. This phenomenon can be detected in the plots if we observe the increase in wind strength during the midday hours. Finally, we observe a strange and different behaviour of the CO particle in *Molí* station. For this pollutant there is a second peak in the midday period.

[3] http://www.valencia.es/ayuntamiento/DatosAbiertos.nsf/

This behaviour probably corresponds to an extra source of pollution that needs to be further studied.

As stated previously, we are interested in predicting pollution levels in real time. Since these levels are only made public with a delay of three hours, we need to produce a prediction model from real time features. We extract the following set of features from the data collected from different sources (detailed in the previous section):

- **Climatological features:** Temperature (Celsius degrees), Relative humidity (Percentage), Pressure (hPa), Wind speed (km/h), Rain (mm/h)
- **Calendar features:** Year, Month, Day in the month, Day in the week, Hour
- **Traffic intensity features:** Traffic level in the surrounding stations (vehicles/hour), traffic level 1, 2, 3 and 24 hours before
- **Pollution features:** Pollution level in the target station 3 and 24 hours before

With this goal we compare several regression learning techniques from R (R Core Team, 2015) in order to identify the technique that is able to better predict the levels of pollution. To test the prediction ability of different models, we learn the models using as training data the registers of 2013 and the first nine months of 2014. We test the models with the last three months of 2014. We use Mean Squared Error (MSE) as a performance measure. Concretely, we employ the following techniques for learning regression models (all of them with the default parameters, unless stated otherwise): Linear Regression (*lr*) (Hornik et al., 2009), quantile regression (*qr*) (Koenker, 2015) with *lasso* method, K nearest neighbours (*IBKreg*) with $k = 10$ (Hornik et al., 2009) , a decision tree for regression (*M5P*) (Hornik et al., 2009), Random Forest (*RF*) (Liaw & Wiener, 2002), Support Vector Machines (*SVM*) (Meyer et al., 2014) and Neural Networks (Venables & Ripley, 2002). In order to compare the predictive performance of these models, we also introduce three baseline models: A model that always predicts the mean of the train data (*TrainMean*), a model that always predicts the mean of the test data (*TestMean*), and a basic model that predicts the same value of the target pollutant 3 hours before (*X3H*).

Table 2 contains the MSE of the regression models for the prediction of the four target pollution levels of the *Molí* station. Results for *Pista de Silla* station and *Avd Francia* station are shown in Table 4 and 3 respectively. If we analyse these results, we can conclude that learned models are improving the performance of the basic baseline models in almost all cases. When we compare the learning techniques in the three tables, the ensemble of decision trees technique (random forest) is the best model in almost all of cases. These results are in concordance with (Singh et al., 2013) where ensembles of trees outperformed other approaches such as SVMs.

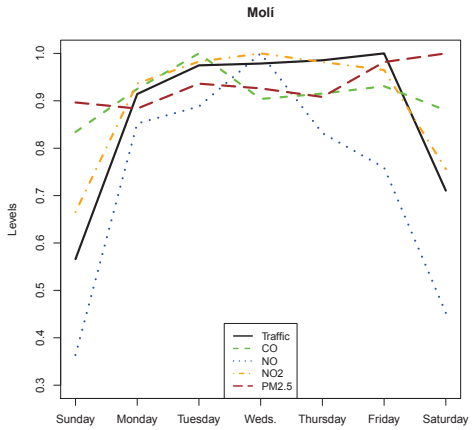


Figure 1. Average weekly traffic intensity and pollution parameters measured in Molí station.

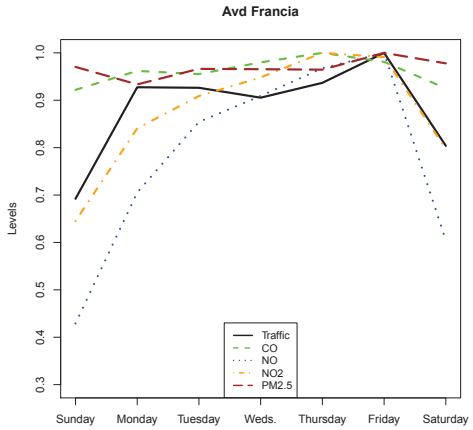


Figure 2. Average weekly traffic intensity and pollution parameters measured in Avd Francia station.

4. Airvlc

In the previous section we have analysed how to obtain real-time air pollution predictions from a given set of features. In this section we summarise Airvlc, a mobile app for Android and iOS and a web application[4]. This application generates from the regression models a map of the city of Valencia showing the predicted intensity of pollution levels. The application also allows the user to configure a set of automatic warnings every time a pollution threshold is reached near the position of the mobile device.

[4] http://airvlc.lidiacontreras.com/

Table 1. Averages and standard deviation of the three pollution detection sensors.

	Traffic		CO		NO		NO2		PM2.5	
	ave	sd	ave	sd	ave	sd	ave	sd	ave	sd
Molí	442.333	339.489	0.116	0.093	8.642	20.311	26.608	21.003	10.650	6.926
Francia	631.569	431.412	0.185	0.122	9.092	19.271	27.840	23.992	7.909	4.235
Silla	484.722	298.768	0.228	0.187	23.559	33.024	45.631	25.376	8.309	6.020

Table 2. Results in MSE of different regression models for Molí Station. The best prediction model is highlighted in bold.

	TrainMean	TestMean	X3h	lr	qr	IBkreg	M5p	RF	SVM	NN
CO	0.086	0.061	0.067	**0.057**	0.060	0.068	0.071	**0.057**	0.063	0.182
NO	30.202	28.739	36.516	25.200	29.805	27.821	25.555	**20.655**	25.870	32.944
NO2	19.918	19.914	25.258	19.680	17.370	15.683	31.242	14.877	**14.488**	32.152
PM2.5	8.803	8.803	8.634	6.889	6.564	6.674	7.248	**6.072**	6.135	13.089

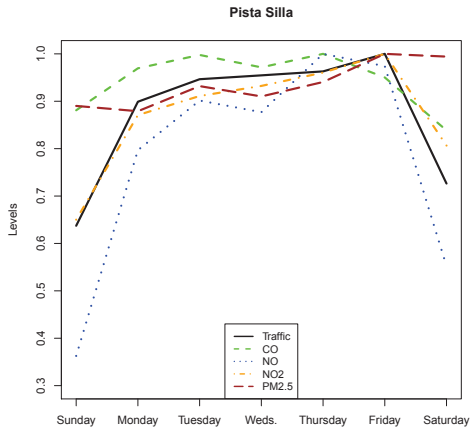


Figure 3. Average weekly traffic intensity and pollution parameters measured in Pista Silla station.

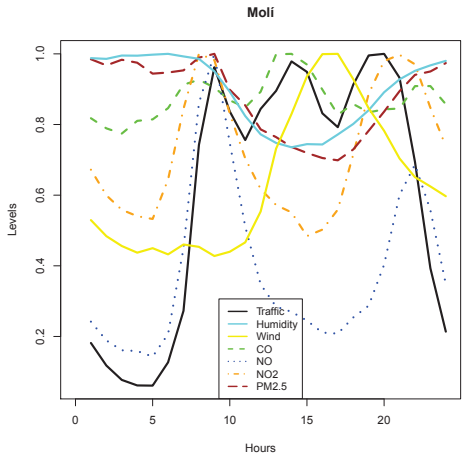


Figure 4. Average daily traffic intensity and pollution parameters measured in Molí station.

4.1. Contamination intensity map

Results of Section 3 show that random forest models obtain the best performance in most cases. Therfore, twelve random forest models are implemented in the Airvlc application. These models are able to predict every hour the level of the four analysed particles at the three pollution detection stations. We want, however, to predict pollution levels at the points of the city where the traffic is measured (1245 points around the city). For that purpose, given any of these points, we extract the features related to traffic intensity from the six nearest traffic sensors. The meteorological features are the same for all the city. The predictions of pollutants in that exact location is computed by the combination of the models corresponding to the three stations. The combination is weighted with respect to the distance of the target point with respect to the measurement stations giving more importance to the closest models. A simpler approach could be to learn a single model from the concatenation of the data from the three stations and then apply this in all the set of target points.

By computing the pollution predictions for a set of strategic and well-distributed locations we are able to estimate a real-time pollution map of the city. The map is generated with *Google Maps* technology. This map shows for each location its pollution level as a dot which colour varies among green, yellow and red depending on the calculated pollution level. If the user selects one of these dots, an extended window is opened where the exact predicted levels are shown. Figure 7 includes a screen-shot of the pollution map of the Airvlc application. The user can also select a second frame in the window of the Airvlc application where he/she can introduce a specific location and then the application computes the predicted pollution levels for that selection. An example of this process is included in Figure 8.

4.2. Risk levels

Figure 8 shows how the pollution levels are presented to users. However, showing just a concentration value of each parameter is not very useful for most users, since most of them are not experts in pollutants and they could not interpret correctly these numbers. In order to improve the comprehensibility of the predictions we have established three ranges of risk represented as speedometer: Low risk (green) corresponds to a measurement that is safe; Medium risk (yellow) when concentrations reach levels to cause harmful effects in people sensitive to air pollution exposure (kids, elderly people...); High risk (red) when concen-

Table 3. Results in MSE of different regression models for Avd. Francia Station. The best prediction model is highlighted in bold.

	TrainMean	TestMean	X3h	lr	qr	IBkreg	M5p	RF	SVM	NN
CO	0.195	0.165	0.224	0.159	0.163	0.168	0.160	**0.153**	0.156	0.324
NO	35.493	33.634	44.955	32.992	36.049	34.092	30.350	**29.517**	33.364	38.262
NO2	23.443	20.900	27.162	16.299	16.718	18.494	23.782	**14.851**	19.100	41.929
PM2.5	3.721	3.718	3.879	3.326	**3.132**	3.523	4.655	3.214	3.265	7.974

Table 4. Results in MSE of different regression models for Pista Silla Station. The best prediction model is highlighted in bold.

	TrainMean	TestMean	X3h	lr	qr	IBkreg	M5p	RF	SVM	NN
CO	0.278	0.222	0.304	0.221	0.227	0.221	0.235	**0.218**	0.268	0.278
NO	49.149	46.232	60.353	39.863	43.524	42.760	44.150	**36.332**	52.438	58.798
NO2	23.135	23.122	30.167	20.861	19.031	18.487	25.972	**16.722**	23.313	49.699
PM2.5	6.911	6.660	7.119	5.663	5.342	5.750	7.189	**5.339**	7.368	11.061

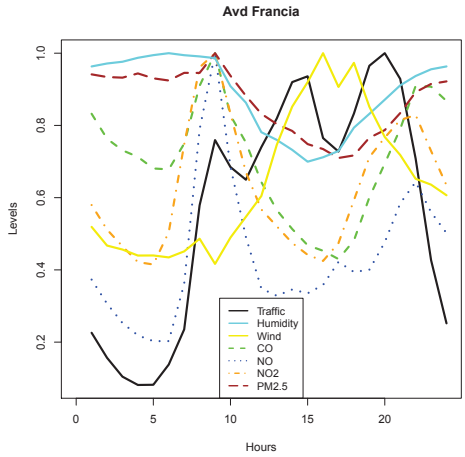


Figure 5. Average daily traffic intensity and pollution parameters measured in Avd Francia station.

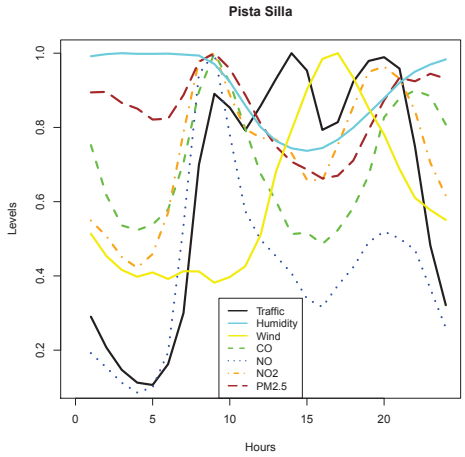


Figure 6. Average daily traffic intensity and pollution parameters measured in Pista Silla station.

trations can cause acute and chronic effects to anyone, especially those with sensitivity.

The ranges of risk shown by the application from the predicted values of the four pollutants are based on the recommendations of the Directive 2008/50/EC (European Comission, 2008). The variable as NOx (oxides of nitrogen) refers to NO or NO2, since the normative establishes the same limits for both levels.

- **Green level:** [NOx] $<$ 14.0 μg/m3 $\wedge$ [CO] $<$ 30.0 mg/m3 $\wedge$ [PM 2.5] $<$ 7.5 μg/m3.
- **Yellow level:** We establish medium risk (yellow level) if the levels do not satisfy the conditions of the green level and the red level.
- **Red level:** [NOx] $\geq$190.0 μg/m3 $\vee$ [CO] $\geq$ 55.0 mg/m3 $\vee$ [PM 2.5] $\geq$ 25.0 μg/m3

4.3. Risk warnings

Airvlc mobile application can be configured to send warnings to users if the device is near to a zone (200 meters approximately) where a high risk level is predicted. These warnings can be personalised by the user in different ways.

For example, the user can establish personal limits for warnings or modify the range of distance for the detection of high risk levels of pollutant concentration. Obviously, the user needs to allow the application to know the actual GPS location of the device

In the case of the web application, given that here it is more complex to know the exact location of the user, we adopt a different strategy. We are working in an automated warning system where the user needs to fix a set of areas, and then the system sends an electronic email whenever a dangerous situation (high risk level by default) is detected.

5. Related work

A wide number of works employs machine learning techniques or statistical approaches for predicting pollution levels. A classical work is (Yi & Prybutok, 1996). In this paper, the authors propose ozone prediction models. Specifically, they develop a neural network model for forecasting daily maximum ozone levels and compare it to previous approaches by regression, and Box-Jenkins ARIMA. The results show that the neural network model improves the performance of the regression and Box-Jenkins ARIMA models tested. Neural networks models have been widely

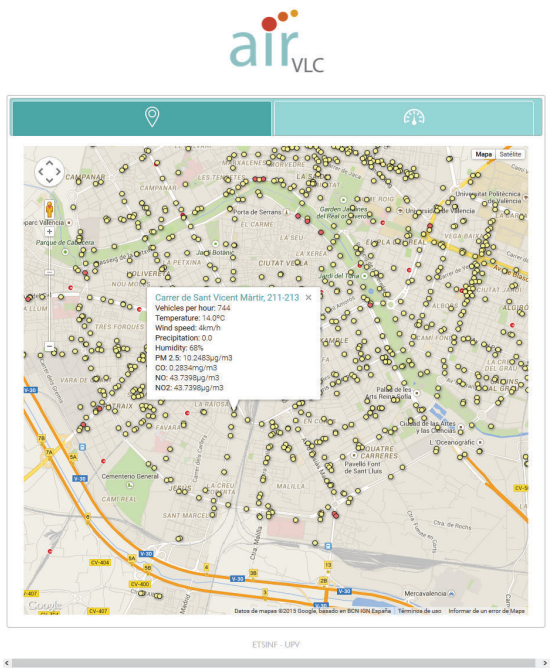


Figure 7. Airvlc application. Real-time pollution map.

employed in this field, a review of these approaches can be found in (Khare & Nagendra, 2006).

A more related work is (Karppinen et al., 2000a). Here the authors propose a modelling system for predicting the traffic volumes, emissions from stationary and vehicular sources, and atmospheric dispersion of pollution in an urban area. They employ four monitoring stations in the Helsinki metropolitan area in 1993. The paper compares the predicted NOx and NO2 concentrations with the results of an urban air quality monitoring network. The agreement of model predictions was better for the two suburban monitoring stations, compared with two urban stations. Some applications of these models are introduced in (Karppinen et al., 2000b). A similar work for the city of Izmir in Turkey is (Elbir, 2003). Here, the authors compare The CALMET meteorological model and its puff dispersion model CALPUFF for predicting dispersion of the sulphur dioxide emissions from industrial and domestic sources.

Another related work, and in this case very recent, is (Donnelly et al., 2015). This paper presents a model for real time air quality forecasts. The predictions are concentrated in nitrogen dioxide (NO2) and they are used to estimate air quality 48 hours in advance. The model is based on a multiple linear regression which uses linearised factors describing variations in concentrations together with meteorological parameters and persistence as predictors.

Our comparison of regression techniques obtains similar

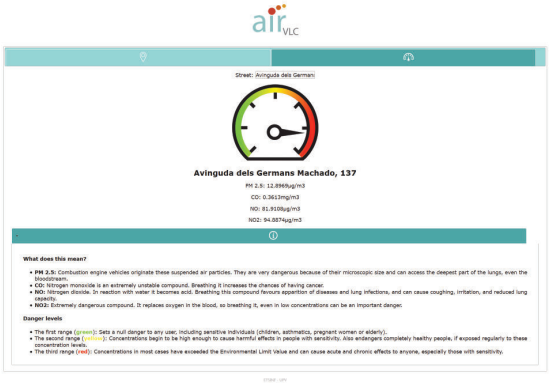


Figure 8. Frame where the user can introduce specific locations to know the predicted levels of pollution.

conclusions to the work presented in (Singh et al., 2013). In this study, principal components analysis (PCA) is performed to identify air pollution sources. From the extracted features, tree based ensemble learning models are induced to predict the urban air quality of Lucknow (India) together with the air quality and meteorological databases for a period of five years.

6. Conclusions

Air pollution can decrease life expectancy since contamination rises the risk of suffering respiratory diseases. Although policies motivating the reduction of emissions of pollutant particles have been introduced in the last years, many cities frequently still present risky levels of air pollution. In these situations, the reduction of the exposure to ambient air pollution is highly recommended. In this work, we have presented Airvlc, an application that predicts in real-time the levels of four dangerous pollutants in a wide set of points in the city of Valencia. The system is able to predict these pollution levels by applying regression models trained from data containing information traffic intensity, persistence of pollutants and meteorological parameters. Airvlc can be a useful tool for avoiding risky locations in terms of air pollution.

As future work we propose the integration of the application in middleware platforms such as Fi-Ware[5], this could help to extend the applicability of the system to other cities or regions. We also are interested in the incorporation of additional features in order to improve the prediction models: wind direction, sand storms, forest wildfires and agricultural burnings... Finally, the use of the tool for the recommendation of routes that minimise the exposure to air pollution.

[5] http://www.fiware.org/

Acknowledgments

We thank the anonymous reviewers for their comments, which have helped to improve this paper significantly. We are also grateful to Ajuntament de València, InnDEA València and specially to Ramón Ferri, Ruth López and Paula Llobet for their help in providing traffic data. This work was supported by the REFRAME project, granted by the European Coordinated Research on Long-term Challenges in Information and Communication Sciences & Technologies ERA-Net (CHIST-ERA), and funded by the Ministerio de Economía y Competitividad in Spain (PCIN-2013-037). It also has been partially supported by the EU (FEDER) and the Spanish MINECO project ref. TIN2013-45732-C4-01 (DAMAS), and by Generalitat Valenciana ref. PROMETEOII/2015/013 (SmartLogic).

References

Donnelly, Aoife, Misstear, Bruce, and Broderick, Brian. Real time air quality forecasting using integrated parametric and non-parametric regression techniques. *Atmospheric Environment*, 103:53–65, 2015.

Elbir, Tolga. Comparison of model predictions with the data of an urban air quality monitoring network in izmir, turkey. *Atmospheric Environment*, 37(15):2149–2157, 2003.

European Comission. Directive 2008/50/ec of the european parliament on ambient air quality and cleaner air for europe. http://ec.europa.eu/environment/air/quality/legislation/directive.htm, 2008.

Hornik, Kurt, Buchta, Christian, and Zeileis, Achim. Open-source machine learning: R meets Weka. *Computational Statistics*, 24(2):225–232, 2009. doi: 10.1007/s00180-008-0119-7.

Karppinen, A, Kukkonen, J, Elolähde, T, Konttinen, M, and Koskentalo, T. A modelling system for predicting urban air pollution:: comparison of model predictions with the data of an urban measurement network in helsinki. *Atmospheric Environment*, 34(22):3735–3743, 2000a.

Karppinen, A, Kukkonen, J, Elolähde, T, Konttinen, M, Koskentalo, T, and Rantakrans, E. A modelling system for predicting urban air pollution: model description and applications in the helsinki metropolitan area. *Atmospheric Environment*, 34(22):3723–3733, 2000b.

Khare, Mukesh and Nagendra, SM Shiva. *Artificial neural networks in vehicular pollution modelling*, volume 41. Springer, 2006.

Koenker, Roger. *quantreg: Quantile Regression*, 2015. URL http://CRAN.R-project.org/package=quantreg. R package version 5.11.

Liaw, Andy and Wiener, Matthew. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002. URL http://CRAN.R-project.org/doc/Rnews/.

Meyer, David, Dimitriadou, Evgenia, Hornik, Kurt, Weingessel, Andreas, and Leisch, Friedrich. *e1071: Misc Functions of the Department of Statistics (e1071), TU Wien*, 2014. URL http://CRAN.R-project.org/package=e1071. R package version 1.6-4.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2015. URL http://www.R-project.org/.

Singh, Kunwar P, Gupta, Shikha, and Rai, Premanjali. Identifying pollution sources and predicting urban air quality using ensemble learning methods. *Atmospheric Environment*, 80:426–437, 2013.

The European Environment Agency . Soer 2015 — the european environment — state and outlook 2015. http://www.eea.europa.eu/soer, 2015.

US Environmental Protection Agency . Particulate matter (pm) regulations. http://www.epa.gov/airquality/particlepollution/index.html, 2015.

Venables, W. N. and Ripley, B. D. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. URL http://www.stats.ox.ac.uk/pub/MASS4. ISBN 0-387-95457-0.

Wilker, Elissa H., Preis, Sarah R., Beiser, Alexa S., Wolf, Philip A., Au, Rhoda, Kloog, Itai, Li, Wenyuan, Schwartz, Joel, Koutrakis, Petros, DeCarli, Charles, Seshadri, Sudha, and Mittleman, Murray A. Long-Term Exposure to Fine Particulate Matter, Residential Proximity to Major Roads and Measures of Brain Structure. *Stroke*, April 2015. doi: 10.1161/strokeaha.114.008348. URL http://dx.doi.org/10.1161/strokeaha.114.008348.

World Health Organisation. Public health, environmental and social determinants of health. http://www.who.int/phe/health_topics/outdoorair/databases/health_impacts/en/, 2015.

Yi, Junsub and Prybutok, Victor R. A neural network model forecasting for prediction of daily maximum ozone concentration in an industrialized urban area. *Environmental Pollution*, 92(3):349–357, 1996.

Stresscapes: Validating Linkages between Place and Stress Expression on Social Media

Martin Sykora M.D.SYKORA@LBORO.AC.UK
Centre for Information Management, SBE, Loughborough University UK

Colin Robertson
Geography and Environmental Studies, Wilfried Laurier University CANADA
Ketan Shankardass
Psychology Department, Wilfried Laurier University CANADA
Rob Feick
School of Planning, University of Waterloo CANADA
Krystelle Shaughnessy
Psychology Department, University of Ottawa CANADA
Becca Coates
CIM, SBE, Loughborough University UK
Haydn Lawrence
Geography and Environmental Studies, Wilfried Laurier University CANADA
Thomas W. Jackson
CIM, SBE, Loughborough University UK

Abstract

Understanding how individuals and groups perceive their surroundings and how different physical and social environments may influence their state-of-mind has intrigued re-searchers for some time. Much of this research has focused on investigating why certain natural and human-built places can engender specific emotive responses (e.g. fear, disgust, joy, etc.) and, by extension, how these responses can be considered in place-making activities such as urban planning and design. Developing a better understanding of the linkages between place and emotional state is challenging in part because both cognitive processes and the concept of place are complex, dynamic and multi-faceted and are mediated by a confluence of contextual, individual and social processes. There is evidence to suggest that social media data produced by individuals in situ and in near real-time may provide novel insights into the nature and dynamics of individuals' responses to their surroundings. The explosion of user-generated digital data and the sensorization of environments, especially in urban settings, provide opportunities to build knowledge of place and state-of-mind linkages that will inform the design and promotion of vibrant place-making by individuals and communities.
In this paper we present a novel study, to be undertaken this summer within the Greater Toronto area in Canada, with 140 recruited participants who are frequent, geo-tagging, Twitter users. The goal of the study will be to assess emotional, acute and chronic stress experienced in urban built-environments and as expressed during daily activities. An existing automated semantic natural language processing tool will be validated through this study, and it is hoped that the methodology developed can be extrapolated to other urban environments as well, with a second validation study already planned to take place next year in London, United Kingdom.

Proceedings of the 2^{nd} International Workshop on Mining Urban Data, Lille, France, 2015.

1. Introduction

In recent years automated processing of rich, geo-tagged, social media text streams, such as Tweets and Facebook status updates is receiving considerable attention in the literature. This is largely motivated by the insights and value that such datasets were shown to provide (Chew & Eysenbach, 2010; O'Connor et al., 2010; Tumasjan et al., 2010; Abel et al., 2012). Social-media streams, in general, al-

low for observing large numbers of spontaneous, real-time interactions and varied expression of opinion, which are often fleeting and private (Miller, 2011). Miller (2011) further points out that some social scientists now see an unprecedented opportunity to study human communication with various applications and contexts, which has been an obstacle up until recently. O'Connor et al. (2010) demonstrated how large-scale trends can be captured from Twitter messages, based on simple sentiment word frequency measures. The researchers evaluated and correlated their Twitter samples against several consumer confidence and political opinion surveys in order to validate their approach, and have pointed out the potential of social-media as a rudimentary yet powerful polling and survey methodology. In her position paper De Choudhury (2013) suggests that mental health studies would benefit from employing social media, as it provides an unbiased collection of an individuals language and behaviour, and Coppersmith et al. (2014) further highlight how social media enables large scale analyses, which has not been previously possible with traditional methods. Eichstaedt et al. (2015) propose a strong argument in favor of employing social media to study heart disease mortality based on psychological characteristics gleaned from Twitter language use. Especially negative emotional language and expressions of stress play an important role. They argue that traditional approaches that use household visits and phone surveys are costly and have limited spatial and temporal precision.

Motivated by this initial evidence we will be investigating emotional acute and chronic stress as expressed in geotagged, in-situ social media language. Our primary focus will be the connection between expressions of stress and the geography of urban built environments; applying geo-spatial analysis methods to define dynamic stress landscapes, or *stresscapes* that will help us to understand how stress varies from place-to-place and from time-to-time within urban centres. As Schwartz and Germaine (2014) rightly point out, studies concerning the combination of social media, identity performance, and place are still rare. Hence, we particularly seek to contribute to recent research related to the linkages between place and expressions of personal or social stress. Research on this topic has traditionally focused on the role of either individual or contextual factors; however, it is necessary to investigate the interplay between individuals and the nature of their immediate surroundings. Assessment of stress is normally overly general, which makes it hard to compare the experience of stress across individuals; whereas focusing on the emotional dimensions of the stress response offers a more specific measure for analysis. We will recontextualize social media expressions through spatial modelling and integration with contextual geospatial datasets describing participants' immediate surroundings. This will lead to new in-sights into how emotional stress is related to particular conditions (e.g. traffic congestion), place types and designs (e.g. public versus private places, high versus low density) and times (e.g. commuting rush hours) within urban communities.

As far as the authors are aware this is the first study of its kind, which will be looking at various forms of stress, linkages to urban environments, and validation of a computational social media analysis tool against 'real' experiences of acute and chronic stress, using already well established and validated measures from literature.

The remainder of the paper is organised as follows. Section 2 introduces some background and prior work on stress in urban environments and the computational tool for emotion based stress detection. Method details and overall validation study design are presented in section 3. Section 4 concludes the paper and suggestions for future work are made.

2. Background

Intensive acute and chronic psychological stress appears to play a causal role in the onset of multiple chronic disease outcomes, such as asthma and obesity (Shankardass et al., 2009; 2014), engendering significant costs related to economic productivity, and health and social service spending (Daar et al., 2007). A body of evidence suggests that the built environment shapes how we experience and respond to stress (Shankardass, 2012). However, there is a critical gap in our understanding of how our environments shape our experience of stressors (e.g., social disorder) and influence how we cope with our perceived stress because of the availability (or lack thereof) of resources, e.g., safe park space (Shankardass, 2012). There is a lack of place-based measures of stress to facilitate research on these interrelationships.

This study uses a *conceptual framework* recently proposed by Shankardass (2012), which builds on Pearlin's stress process heuristic (Pearlin, 1999), where sources of stress that are perceived as stressful can manifest emotion-al, behavioural and physiological responses (e.g., negative affect, smoking and endocrine activation, respectively). Two critical mediators of these responses are resource appraisal and coping behaviours, while the neighbourhood built environment can present stressors and offer resources that condition how we cope in space and time. This conceptual framework guides our hypotheses about which confounders and moderators ought to be considered in building a prediction model of emotional stress on stress-related endocrine activation. These include personality differences, such as trait anxiety and pessimism (Chang, 2002) - *which may confound the relationship* - and low self-esteem (Dumont & Provost, 1999) - *which may increase the effect of per-*

ceived stress on chronic endocrine activation, as well as low social support (ibid.) - *which may increase the effect of perceived stress on chronic endocrine activation*, and sex and gender (Baum & Grunberg, 1991) with hard-to-predict moderating effects on the relationship. Chronic endocrine activation may be more likely where individuals adopt coping styles that do not effectively deal with stressors (e.g., avoidance coping, rather than approach coping or problem-oriented coping.

Taking all this into account, our overall goal is to further develop and validate an ontology of emotional stress (based on presence of negative and the lack of positive affect) that will facilitate measurement through semantic analysis of geo-tagged Twitter posts (Sykora et al., 2013) and assess the predictive validity of perceived psychological stress.

2.1. Detection of Stress from Tweets

There are numerous systems for effective, efficient and accurate sentiment and emotion detection from language. A broader overview of the various approaches is available in Thelwall et al. (2012). One of the popular techniques is based on the use of words and phrase dictionaries with known associated sentiment polarities or emotion categories; however, these dictionaries, although sometimes combined and semi-automatically generated for better cross-domain performance, are relatively flat and lack semantic expressivity. Even more recently Eichstaedt et al. (2015) still used a combination of simple dictionaries to perform their automated tweet analysis.

In this work we employ an ontology based approach, which is essentially a map of words and phrases with a much richer semantic representation than simple dictionaries. The system we will use is called EMOTIVE and is based on (1) a custom Natural Language Processing (NLP) pipeline, which parses tweets and classifies parts-of-speech tags, and (2) an ontology, in which emotions, related phrases and terms (including a wide set of intensifiers, conjunctions, negators, interjections), and linguistic analysis rules are represented and matched against (Sykora et al., 2013). EMOTIVE automatically detects expressions of eight well recognised and fine-grained emotions in sparse texts (e.g. Tweets). The system discovers the following range of emotions; anger, disgust, fear, happiness, sadness, surprise (also known as Ekman's basic emotions - Ekman and Davidson, (1994)), and confusion and shame, but at the same time differentiates emotions by strength (also known as activation level, e.g. fear - 'uneasy', 'fearful', 'petrified'). An evaluation of the system against other benchmarks performed in Sykora et al. (2013) showed excellent results, with a very high f-measure of .962. Given the rich representation of emotions and the ontology this is based on, we will link and extended this system into representing stress in its various shapes and forms, with the intention to validate this system against real experiences of stress (see next section for details on this validation study).

3. Methodology and Study Design

Emotional stress has been conceptualized in different ways, including in terms of negative affect and as a state of distress. Two criteria will be utilized as criteria for validation in this study, including;

- The single-item distress thermometer, which is a simple Likert scale shaped like a vertical thermometer that asks the subject to select a number corresponding to their level of distress (Zwahlen et al., 2008).

- The 10-item negative affect scale from the expanded version of the Positive and Negative Affect Schedule (PANAS-X) will also be used (Watson & Clark, 1994).

These measures will be framed using moment instructions, i.e., we will ask whether participants have experienced distress/negative affect "right now", that is, at the present moment. The aim will be to collect at least 10 measures of each during the two week follow-up (see section on overall design). An algorithm will be used to scan a series of discrete stress-related terms (still being compiled) in real-time for all participants and randomly trigger the study follow-up survey via SMS / text message. This survey will be triggered roughly at evenly-spaced time points across the two-week follow-up period, based on the rate of tweets by the study participants. In order to understand the context of Tweets, a series of questions will also be asked, to assess what activity mode the participant was in (e.g., work, play, commute, domestic, study) at the time of the Tweet, and whether and how the surrounding environment influenced the Tweet in any way. Participants will also be requested to automatically geo-tag their Tweets by default, for the duration of the study follow-up.

3.1. Recruitment

The study population will include long-term (>3 months at study entry) active (>4 posts per week) Twitter users who are free from anxiety disorders. The planned sample size is 140 participants, which was calculated based on Bland and Altman (1986) and scaled up by 40% for anticipated drop-outs. A pool of potential study participants (i.e. long term, active Twitter users) will be identified from a database of several million collected Tweets, geo-tagged in the Toronto area. The study will also be limited to participants who live and work in the greater Toronto area.

3.2. Overall Design

The study can be broken-up into three phases:

- a. Running enrollment of study participants (1 month, begins mid-May 2015)
- b. Follow-up period (2 months) each participant 2 weeks
- c. Study exit and hair sampling (running in parallel to b), and ultimately study takedown by mid-September.

Data about participants will be collected at study entry, specifically socio-demographic and psychological information. Information about the experience of emotional stress and relationship with place will be collected at approximately 10 time points during the follow-up period. Subsequently in the study exit participants will be asked to complete a checklist of potentially stressful events, in order to understand the influence of major life events during the follow-up period.

3.3. Assessment of Chronic Psychological Stress

The research team has also secured additional funding to augment this summer's study with a collection of hair samples for cortisol analysis in order to examine how our devised measure of stress predicts chronic activation and allostatic load (i.e. physiological dysfunction). Participants who agreed to this will provide a 0.95 cm hair sample (from the root) at *study exit*, which will be analysed using immunoassay analysis following a validated protocol (Gow et al., 2010). Because hair grows at a rate of approximately 1.25 cm per month, cortisol embedded in this sample length will reflect a retrospective record of approximately three prior weeks. Hair cortisol level will be considered an outcome in regression models from our measure of stress.

4. Conclusion and Future Work

There are several key benefits of our study. First, a place-based measure of physiological stress will significantly broaden the potential for research to examine how the neighbourhood environment affects human health and well-being. This could lead to studies that inform the design of neighbourhoods that facilitate stronger prevention and management of stress-related illnesses. Second, the final predictive validation model will create empirical evidence of the inter-relationship amongst emotional and psychological stress, endocrine activation and a range of demographic and psychological traits. The study described in this paper will be repeated, with lessons learned, next year in the city of London. It is hoped that this will strengthen the model and validation, and will also provide a culturally different built environment, with its own characteristics. We hope this will lend itself to some interesting analyses.

Acknowledgments

We are grateful for this work to be supported by an SSHRC (Social Sciences and Humanities Research Council) Partnership Development grant and partly by an internal Wilfrid Laurier University grant.

References

Abel, F., Hauff, C., Houben, G., Stronkman, R., and Tao, K. Semantics + filtering + search = twitcident exploring information in social web streams. In *Proceedings of the 23rd ACM International Conference on Hypertext and Social Media*, Milwaukee, USA, 2012.

Baum, A. and Grunberg, N. E. Gender, stress, and health. *Health Psychology*, 10(2):80–85, 1991.

Bland, J. M. and Altman, D. G. Statistical methods for assessing agreement between two methods of clinical measurement. *Lancet*, 327(8476):307–310, 1986.

Chang, E. C. Optimismpessimism and stress appraisal: Testing a cognitive interactive model of psychological adjustment in adults. *Cognitive Therapy Research*, 26 (5):675–690, 2002.

Chew, C. and Eysenbach, G. Pandemics in the age of twitter: content analysis of tweets during the 2009 h1n1 outbreak. *PLOS One*, 5(11):e14118, 2010.

Choudhury, M. De. Role of social media in tackling challenges in mental health. In *Proceedings of the 2nd International Workshop on Socially-aware Multimedia*, pp. 49–52, New York, USA., 2013.

Coppersmith, G., Harman, C., and Dredze, M. Measuring post traumatic stress disorder in twitter. In *Proceedings of the 8th International AAAI Conference on Weblogs and Social Media*, Ann Arbor, USA, 2014.

Daar, A. S., Singer, P. A., Persad, D. L., Pramming, S. K., Matthews, D. R., Beaglehole, R., ..., and Bell. Grand challenges in chronic non-communicable diseases. *Nature*, 450(7169):494–496, 2007.

Dumont, M. and Provost, M. A. Resilience in adolescents: Protective role of social support, coping strategies, self-esteem, and social activities on experience of stress and depression. *Journal of Youth Adolescence*, 28(3):343–363, 1999.

Eichstaedt, J. C., Schwartz, H. A., Kern, M. L., Park, G., Labarthe, D. R., Merchant, R. M., ..., and Seligman, M. E. Psychological language on twitter predicts county-level heart disease mortality. *Psychological Science*, 26 (2):159–169, 2015.

Ekman, P. and Davidson, R. J. (eds.). *The Nature of Emotion: Fundamental Questions*. Affective Science. Oxford University Press, 1994.

Gow, R., Thomson, S., Rieder, M., Uum, S. Van, and Koren, G. An assessment of cortisol analysis in hair and its clinical applications. *Forensic Science International*, 196(1):32–37, 2010.

Miller, G. Social scientists wade into the tweet stream. *Science*, 333(6051):1814–1815, 2011.

O'Connor, B., Balasubramanyan, R., Routledge, B., and Smith, N. From tweets to polls: Linking text sentiment to public opinion time series. In *Proceedings of the 4th International AAAI Conference on Weblogs and Social Media*, Washington D.C., USA, 2010.

Pearlin, L. I. *The stress process revisited: Reflections on concepts and their interelationships*. Handbook on The Sociology of Mental Health. Plenum Press, New York, 1999.

Schwartz, R. and Germaine, R. H. The spatial self: Location-based identity performance on social media. *New Media & Society*, 2014. doi: 10.1177/1461444814531364.

Shankardass, K. Place-based stress and chronic disease: A systems view of environmental determinants. In *Rethinking Social Epidemiology*, pp. 113–136. Springer, Netherlands, 2012.

Shankardass, K., McConnell, R., Jerrett, M., Milam, J., Richardson, J., and Berhane, K. Parental stress increases the effect of traffic-related air pollution on childhood asthma incidence. In *Proceedings of the National Academy of Sciences*, volume 106, pp. 12406–12411, USA, 2009.

Shankardass, K., McConnell, R., Jerrett, M., Lam, C., Wolch, J., Milam, J., Gilliland, F., and Berhane, K. 2014. parental stress increases body mass index trajectory in preadolescents. *Pediatric Obesity*, 9(6):435–442, 2014.

Sykora, M., Jackson, T. W., O'Brien, A., and Elayan, S. Emotive ontology: Extracting fine-grained emotions from terse, informal messages. *IADIS International Journal on Computer Science and Information Systems*, 8(2):106–118, 2013.

Thelwall, M., Buckley, K., and Paltoglou, G. Sentiment strength detection for the social web. *Journal of the American Society for Information Science and Technology*, 63(1):163–173, 2012.

Tumasjan, A., Sprenger, T. O., and Welpe, I. M. Predicting elections with twitter: What 140 characters reveal about political sentiment. In *Proceedings of the 4th International AAAI Conference on Weblogs and Social Media*, Washington D.C., USA, 2010.

Watson, D. and Clark, L. A. The panas-x. manual for the positive and negative affect schedule: expanded form. Technical report, University of Iowa, Iowa City, IA, USA, 1994. URL `http://www2.psychology.uiowa.edu/faculty/clark/panas-x.pdf`.

Zwahlen, D., Hagenbuch, N., Carley, M., Recklitis, C., and Buchi, S. Screening cancer patients' families with the distress thermometer (dt): a validation study. *Psycho-Oncology*, 17(10):959–966, 2008.

Car-traffic forecasting: A representation learning approach

Ali Ziat, Gabriella Contardo, Nicolas Baskiotis, Ludovic Denoyer FIRSTNAME.LASTNAME@LIP6.FR
Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France

Abstract

We address the problem of learning over multiple inter-dependent temporal sequences where dependencies are modeled by a graph. We propose a model that is able to simultaneously fill in missing values and predict future ones. This approach is based on representation learning techniques, where temporal data are represented in a latent vector space. Information completion (missing values) and prediction are then performed on this latent representation. In particular, the model allows us to perform both tasks using a unique formalism, whereas most often they are addressed separately using different methods. The model has been tested for a concrete application: car-traffic forecasting where each time series characterizes a particular road and where the graph structure corresponds to the road map of the city.

1. Introduction

Traffic data has particular characteristics that can not be fully handled by classical sequential and temporal models: they contain multiple missing values, and one has to consider simultaneously multiple sources that can be somehow related, by spatial proximity for example.

We propose a novel method that aims at integrating these aspects in one single model. The proposed approach is based on representation learning techniques aiming at projecting the observations in a continuous latent space, each sequence being modeled at each time-step by a point in this space. It has many advantages w.r.t existing techniques: it is able to simultaneously learn how to fill missing values and to predict the future of the observed temporal data, avoiding to use two different models, and it naturally allows one to deal with information sources that are organized among a graph structure. Moreover, the model is based on continuous optimization schemes, allowing a fast optimization over large scale datasets.

Proceedings of the 2^{nd} International Workshop on Mining Urban Data, Lille, France, 2015.

2. Context and Model

2.1. Notations and Tasks

Let us consider a set of n temporal sequences $\mathbf{x_1}, ..\mathbf{x_n}$ such that $x_i^{(t)} \in \mathcal{X}$ is the value of the i-th sequence at time t defined by $\mathbf{x_i} = (x_i^{(1)}, .., x_i^{(T)})$. In the case where $\mathcal{X}$ is $\mathbb{R}^m$, the context corresponds to multiple multivariate time series. The sequences contain missing values so we also define a mask $m_i^{(t)}$ such that $m_i^{(t)} = 1$ if value $x_i^{(t)}$ is observed - and thus available for training the system - and $m_i^{(t)} = 0$ if $x_i^{(t)}$ is missing - and thus has to be predicted by the model. In addition, we consider that there exists a set of relations between the sequences which correspond to an external information, like spatial proximity for example when $\mathcal{X}$ is discrete. The sequences are thus organized in a graph $\mathcal{G} = \{e_{i,j}\}$ such that $e_{i,j} = 1$ means that $\mathbf{x_i}$ and $\mathbf{x_j}$ are related, and $e_{i,j} = 0$ elsewhere.

2.2. Model

The *RepresentAtIoN-baSed TempOral Relational Model* (RAINSTORM) is a loss-based model which is described through a continuous derivable loss function that will be optimized using classical optimization techniques.

Let us define $\mathcal{L}(\theta, \gamma, \mathbf{z})$ the loss function to minimize where $\mathbf{z}$ is the set of all the vectors $z_i^{(t)}$ for $i \in [1..n]$ and $t \in [1..T]$, T being the size of the observed time windows i.e. the history of the time series. We define $\mathcal{L}$ as:

$$\begin{aligned}\mathcal{L}(\theta, \gamma, \mathbf{z}) = &\frac{1}{O}\sum_{i=1}^{n}\sum_{t=1}^{T} m_i^{(t)} \Delta(f_\theta(z_i^{(t)}), x_i^{(t)}) \text{ (term 1)} \\ &+ \lambda_{dyn}\sum_{i=1}^{n}\sum_{t=1}^{T-1} ||z_i^{(t+1)} - h_\gamma(z_i^{(t)})||^2 \text{ (term 2)} \\ &+ \lambda_{struct}\sum_{i,j\in[1..N]^2}\sum_{t=1}^{T} e_{i,j}||z_i^{(t)} - z_j^{(t)}||^2 \text{ (term 3)}\end{aligned} \tag{1}$$

where O is the number of observed values i.e. values such

that $m_i^{(t)} = 1$.

This loss function contains three terms, each one associated with one of the constraints that have been presented previously:

- Term 1 aims at simultaneously learn $\mathbf{z}$ and a function f_θ - called **decoding function** - such that, from $z_i^{(t)}$, f_θ can be used to predict the value $x_i^{(t)}$. The function $f_\theta(z_i^{(t)})$ is defined as $f_\theta : \mathbb{R}^N \to \mathcal{X}$. Δ is used to measure the error between predicting $f_\theta(z_i^{(t)})$ instead of $x_i^{(t)}$, $m_i^{(t)}$ playing the role of a mask restricting to compute this function only on the observed values.
- Term 2 aims at finding values $z_i^{(.)}$ and a dynamic model h_γ such that, when applied to $z_i^{(t)}$, h_γ allows us to predict the representation of the next state of time series i i.e. $z_i^{(t+1)}$. h_γ is the **dynamic function** which models the dynamicity of each series directly in the latent space: $h_\gamma : \mathbb{R}^N \to \mathbb{R}^N$. The parameters γ will be learned to minimize the mean square error between the prediction $h_\gamma(z_i^{(t)})$ and $z_i^{(t+1)}$.
- At last, term 3 corresponds to a **structural regularity** over the graph structure that encourages the model to learn closer representations for time series that are related. This will force the model to learn representations that reflect the structure of the considered graph.

λ_{dyn} and λ_{struct} are manually defined coefficients that weight the importance of the different elements in the loss function.

The learning problem aims at minimizing the loss function $\mathcal{L}(\theta, \gamma, \mathbf{z})$ simultaneously on θ, γ and $\mathbf{z}$. By restricting the f_θ and h_γ to be continuous derivable functions, we can use gradient-descent based optimization approaches.

3. Traffic Forecasting and Experiments

Experiences have been made on two datasets from the cities of Beijing and Warsaw. The dataset are provided by (Zheng, 2011) and (ICDM) and are not described here for sake of space.

3.1. Models

We propose to compare the RAINSTORM approach to the following baseline models, some baselines being used for data completion, and some others for prediction. For the completion problem we consider the following models:
MF: This correspond to the classical matrix factorization framework for data completion.
MF-with geographic context: This method is the one named TSE (traffic speed estimation) in (Shang et al., 2014).
For the prediction task, we consider:
NeuralNetwork: This is the classical baseline method used in traffic forecasting based on a neural network architecture, described for instance in (Dougherty & Cobbett, 1997).
SAE: This is the method described in (Lv et al., 2014).

We also compare RAINSTORM with a model based on a heuristic able to perform both completion and prediction that we call **RoadMean** and can be described as follow: this model predicts and fills missing value with the mean of observed values on the sequence.

3.2. Experiments and Results

We compare our model with baselines approach for the two tasks of completion and prediction. Results are reported in Table 1. and Table 2.

Table 1. Prediction at $T+1$, comparison between described baselines models and the RAINSTORM model for different size of latent space N with a root mean square error (RMSE)

N	Model/Dataset	Beijing	Warsaw	
		Volume	Volume	Speed
	RoadMean	5.51	5.09	11.02
	NeuralNetwork	4.77	4.27	8.05
	SAE	4.75	4.27	7.85
5	RAINSTORM	4.82	4.28	7.74
10	RAINSTORM	4.78	**4.20**	7.21
20	RAINSTORM	**4.54**	4.21	**7.19**
50	RAINSTORM	4.66	**4.20**	7.60

Table 2. Completion for 50% missing data, comparison between described baselines models and the RAINSTORM model for different sizes N of the latent space with a root mean square error (RMSE)

N	Model/Dataset	Beijing	Warsaw	
		Volume	Volume	Speed
	RoadMean	5.55	5.00	11.10
	MF	3.58	3.16	6.80
	MF-Geo	3.24	2.99	6.49
5	RAINSTORM	2.99	3.12	6.49
10	RAINSTORM	3.03	3.00	6.24
20	RAINSTORM	3.22	2.94	**6.23**
50	RAINSTORM	**2.97**	**2.93**	6.70

4. Conclusion

We have presented a new way to learn over incomplete multiple sources of temporal relational data sources. The RAINSTORM approach is based on representation learning techniques and aims at integrating in a latent space the observed information, the dynamicity of the sequences of

data, and their relations. In comparison to baselines models that have been developed for prediction only or completion only, our approach shows interesting performance and is able to simultaneously complete missing values and predict the future evolution of the data.

References

Dougherty, Mark S and Cobbett, Mark R. Short-term inter-urban traffic forecasts using neural networks. *International journal of forecasting*, 13(1):21–31, 1997.

ICDM. I.E.E.E icdm contest: Tomtom traffic prediction for intelligent gps navigation.

Lv, Yisheng, Duan, Yanjie, Kang, Wenwen, Li, Zhengxi, and Wang, F-Y. Traffic flow prediction with big data: A deep learning approach. 2014.

Shang, Jingbo, Zheng, Yu, Tong, Wenzhu, Chang, Eric, and Yu, Yong. Inferring gas consumption and pollution emission of vehicles throughout a city. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014.

Zheng, Yu. T-drive trajectory data sample, August 2011. URL http://research.microsoft.com/apps/pubs/default.aspx?id=152883.

On Predicting Traveling Times in Scheduled Transportation (Abstract)

Avigdor Gal AVIGAL@IE.TECHNION.AC.IL
Avishai Mandelbaum AVIM@IE.TECHNION.AC.IL
Francois Schnitzler FRANCOIS@EE.TECHNION.AC.IL
Arik Senderovich SARIKS@TX.TECHNION.AC.IL
Technion - Israel Institute of Technology, Haifa, Israel

Matthias Weidlich WEIDLIMA@INFORMATIK.HU-BERLIN.DE
Humboldt-Universität zu Berlin, Berlin, Germany

Traveling time prediction

Urban mobility impacts urban life to a great extent. People, living in cities, plan their daily schedule around anticipated traffic patterns. Some wake-up early to "beat" rush hour. Others stay at home and work during days when a convention comes to town.

To enhance urban mobility, much research was invested in traveling time prediction, see (Wu et al., 2004). That is, given an origin and destination, provide a passenger with an accurate estimation of how long a journey lasts. In particular, the ability to predict traveling time in scheduled transportation, *e.g.,* buses, was shown to be feasible (Chien et al., 2002).

In this work, we address the problem of *online travel time prediction* in the context of a bus journey. That is, a journey may be ongoing in the sense that journey events already indicated the progress of the bus on its route. For such an ongoing journey, we are interested in the current prediction of the traveling time from the current bus stop to some destination via a particular sequence of stops, which is defined by the respective journey pattern.

Prediction Approach

To address the problem of online travel time prediction, we investigate a novel use of methods from Queueing Theory and Machine Learning in the prediction process. We propose a prediction engine that, given a scheduled bus journey (route) and a 'source/destination' pair, provides an estimate for the traveling time, while considering both historical data and real-time streams of information that are transmitted by buses. To do so, we model buses as clients that go through a journey of segments that are interpreted as a network of queues. We propose a model that uses natural segmentation of the data according to bus stops and a set of predictors, some use learning while others are learning-free, to estimate traveling time.

The model of journey segments. As the foundation of our approach, we propose to model each bus trip by using a segmentation model as follows. A trip between two stops consists of segments, with each segment being represented by a 'start' stop and an 'end' stop, see Figure 1. Given the first stop of a trip ω_1 and the last stop of a trip ω_n, the intermediate stops are known in advance since each bus follows a predefined journey pattern. Therefore, a trip can be described by segments that are characterized by a pair of stops of the form $\langle \omega_i, \omega_{i+1} \rangle$ (Figure 1). This segmented model, in turn, allows for fine-granular grounding of the prediction of traveling time $T(\langle \omega_1, \ldots, \omega_n \rangle, t_{\omega_1})$ for a sequence of stops $\langle \omega_1, \ldots, \omega_n \rangle$ when departing at time t_{ω_1}: instead of considering only journeys that follow the same sequence of stops $\langle \omega_1, \ldots, \omega_n \rangle$, all journeys that share some segments can be used for prediction.

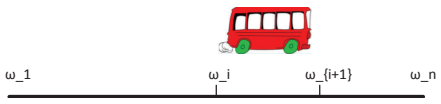


Figure 1. A segmented model of traveling times

Using information on bus stops, the prediction of the journey traveling time $T(\langle \omega_1, \ldots, \omega_n \rangle, t_{\omega_1})$ is traced back to the sum of traveling times per segment. The traveling time per segment is assumed to be independent of a specific journey pattern and, thus, also independent of a specific journey:

$$T(\langle \omega_1, \ldots, \omega_n \rangle, t_{\omega_1}) = \sum_{i=1}^{n-1} T(\langle \omega_i, \omega_{i+1} \rangle, t_{\omega_i}),$$

where $t_{\omega_{n-1}} = t_{\omega_1} + T(\langle \omega_1, \omega_{n-1} \rangle, t_{\omega_1})$.

Prediction based on the snapshot principle. A first set of predictors is grounded in heavy-traffic approximations

Proceedings of the 2^{nd} International Workshop on Mining Urban Data, Lille, France, 2015.

in Queueing Theory. It is non-learning, in the sense that it does not generalize prediction from historical events, but rather uses recent events to predict future traveling times. Applied to our context, the main idea is that a bus that passes through a segment, will experience the same traveling time as another bus that has just passed through that segment (not necessarily of the same type, line, *etc*). Following this line, we define a single-segment snapshot predictor, called Last-Bus-to-Travel-Segment (LBTS).

To use this predictor to address the online travel time prediction problem, it needs to be lifted to a network setting. To this end, we exploit the fact that the snapshot principle holds for networks of queues, when the routing through this network is known in advance (Reiman & Simon, 1990). Clearly, in scheduled transportation, this is the case, so that we define a multi-segment (network) snapshot predictor, called Last-Bus-to-Travel-Network. It is derived by summing up the LBTS predictions for the segments of the respective journey pattern of the bus for which the prediction is made.

Prediction using Machine Learning methods. A second set of predictors comes from Machine Learning and is based on regression trees. They exploit past journey logs to learn a prediction model, and then use this model to make a prediction on new instances of the problem, in our case, traveling times as part of current journeys.

As a first step, we formalize the traveling times prediction problem as a regression problem. Features considered for the regression include the travel time of the last bus that used that segment (LBTS, as introduced above); the interval between the time the last bus left the segment and the estimated time to enter the segment; the day of the week; and the time of the day. For the resulting regression model, various generic algorithms are applied to derive an ensemble of regression trees that is then used to solve the prediction problem. Specifically, random forests, extremely randomized forests, AdaBoost, and gradient tree boosting are leveraged.

In a final step, the above methods originating from Queueing Theory and Machine Learning are combined. That is, we rely on the boosting algorithms and modify them such that the first model considered in the boosting is the snapshot predictor model.

Evaluation

To demonstrate the value of our approach, we tested the proposed predictors using bus data that comes from the bus network in the city of Dublin.[1] The data includes location of buses that is sampled in intervals of 5 to 300 seconds, depending on the current location of the bus.

Using this data, we empirically evaluated the prediction accuracy of the presented methods. The main results of our experiments have been:

- Prediction methods that combine the snapshot principle and Machine Learning techniques are superior in quality of prediction to both snapshot predictors and Machine Learning methods (that do not include the snapshot predictor).
- The prediction error increases with the number of bus stops per journey. However, when considering the relative error, it is stable for all trip lengths, i.e. the predictors do not deteriorate proportionally to the length of the journey (in stops).
- Surprisingly, the snapshot predictor does not deteriorate for longer trips, therefore contradicting the hypothesis that the snapshot predictor would be more precise for journeys with higher temporal proximity to the current journey.

Conclusion

In this work, we presented a novel approach towards predicting travel time in urban public transportation. It is grounded in a partitioning of the travel time into stop-based segments, and combines the use of Machine Learning and Queueing Theory predictors to model traveling time in each segment. Our empirical evaluations confirmed that the combination of methods indeed improves performance. Moreover, we observed that the snapshot predictor is, counter-intuitively, unaffected by the length of a journey. This leads to positive evidence in favor of applying mixed Queue and Machine Learning predictors in similar settings.

This work was supported by the EU INSIGHT project (FP7-ICT 318225).

References

Chien, Steven I-Jy, Ding, Yuqing, and Wei, Chienhung. Dynamic bus arrival time prediction with artificial neural networks. *Journal of Transportation Engineering*, 128 (5):429–438, 2002.

Reiman, Martin I and Simon, Burton. A network of priority queues in heavy traffic: One bottleneck station. *Queueing Systems*, 6(1):33–57, 1990.

Wu, Chun-Hsin, Ho, Jan-Ming, and Lee, Der-Tsai. Travel-time prediction with support vector regression. *Intelligent Transportation Systems, IEEE Transactions on*, 5(4):276–281, 2004.

[1] See also http://www.dublinked.ie/ and http://www.insight-ict.eu/

Report from Dagstuhl: SocioPaths - Multimodal Door-to-Door Route planning via Social Paths

Thomas Liebig THOMAS.LIEBIG@TU-DORTMUND.DE
University of Dortmund, 44221 Dortmund, Germany

Sabine Storandt STORANDT@INFORMATIK.UNI-FREIBURG.DE
University of Freiburg, 79110 Freiburg, Germany

Peter Sanders SANDERS@KIT.EDU
University of Karlsruhe, 76128 Karlsruhe, Germany

Walied Othman WALIED.OTHMAN@GMAIL.COM
Triade systems

Stefan Funke FUNKE@FMI.UNI-STUTTGART.DE
University of Stuttgart, 70569 Stuttgart, Germany

Abstract

Individual multi-modal trip planning is a major task in transportation science. With increasing availability of new means of transportation personal constraints (e.g. elevator phobia or fear of flying) and preferences (e.g. train over bus) gain higher impact. Existing trip planners are mostly based on static time-tables and road-network data. Furthermore an objective function that covers individual constraints and preferences on route choice is hard to find for existing trip planners.

In this position paper we present an approach that incorporates the 'wisdom of the crowd' by construction of a transfer graph based on previously successfully performed trips of other persons. By this approach personal constraints and preferences may easily be taken under consideration by filtering those routes which were performed by people with similar restrictions. Also regular congestions may be taken into consideration as these are already in the data. In case of hazards or blockages corresponding connections can be removed in the transfer graph and alternatives are provided. With a sufficiently large set of initial routes, we expect the method to produce reasonable route suggestions.

Proceedings of the 2^{nd} International Workshop on Mining Urban Data, Lille, France, 2015.

1. Introduction

The upcoming means of transportation (e.g. autonomous or flying cars) enable a more flexible individual mobility. Moreover some of these transportation means can be carried within the other (as we do nowadays with bikes in buses or trains on ferries). This leads to novel options when travelling to some location. At the same time personal constraints (e.g. elevator phobia, fear of flying) have a stronger impact on personal route choices. The task to plan a route from one start location to a target location is called trip planning, when multiple means of transportation (also called 'travel modes') are involved this becomes multi-modal trip planning.

Existing trip planning algorithms operate on a graph representation of the road network the so-called traffic network G consisting of vertices V and connecting edges E: Every edge $e \in E$ of the traffic network represents a segment (e.g. a street, a flight corridor, or the connection among subsequent bus stops) The vertices V represent junctions between segments and therefore locations where decisions on travel directions can be made. A cost function maps each edge to a positive number that denotes how much it would 'cost' to travel the corresponding segment. The cost function needs to be consistent throughout the traffic network, but can be defined in several ways, such that it holds the most important aspects: for example length of the segment, travel time, or comfortableness. With a given start and end location in the traffic network, trip planning searches the path that connects start and goal and minimizes the cost.

Many trip planning algorithms exist in literature, for a brief overview we point the reader to (Bast et al., 2015) and (Delling et al., 2009). To summarize, the shortcomings of existing route planners are:

- They are mostly based on static time-table and road network data,
- The objective functions to produce routes people actually use are surprisingly hard to find. Current solutions either list all pareto-optimal routes, which is time consuming and results in a too large solution space for the end user, or, use an ad-hoc restriction to some routes without any validation via user experience.
- Not all public transit timetable information is available,
- modelling transfer buffers or walking times is hard even with complete timetable information,
- routes people prefer are often also determined by unknown factors like traffic congestion and overcrowding

In contrast, the hereby presented approach bases on the main idea to stitch real, recorded (historic) travel segments of other travelers together into a travel plan. This stitching approach poses the following challenges that are detailed in this paper. Initially, historical data is needed for bootstrapping. The approach has to stitch together a travel plan at query time that also reflects the user's preferences. In addition the practicality of this plan has to be validated. Historical, real-time and predicted traffic knowledge (e.g. blocked roads, need to be incorporated). While we identify these challenges, and provide solution sketches for some challenges in this position paper, we leave solving of a few points for future work.

Our approach constructs a transfer graph from given routes and filters the connections in case of constraints. The resulting transfer graph can be used for routing. This procedure is carried out in four steps: a) Sourcing routes, b) Constructing the transfer graph, c) Adjust in case a departure time is specified, d) Adjust in case current traffic conditions make a transfer impossible.

This paper is a position paper that presents an outline of our approach and is an introduction to our current work on route computations. Application of this algorithm to real routes is in preparation but this description is not included in this paper.

This position paper is structured as follows. Section 2 starts with a brief primer on trip planning and highlights current literature in multi-modal trip planning. Section 3 provides details on our approach, followed by Section 4 on a discussion and future research directions.

2. Related Work

A popular algorithm for trip computation is A^* (Hart et al., 1968), this method searches the minimal connecting path iteratively, beginning at the goal. Not traversing all possible detours, A^* tests the most promising ones first, based on a lower-bound heuristic on the cost function that estimates the minimal travel costs between any two locations. An example for such a heuristic is the geographical distance, which is always lower than the road based distance and therefore a suitable heuristic in case path length is the cost function. In multi-modal trip planning multiple of these traffic networks (one for each mode) are linked together at locations where it is possible to switch from one mode to another (transfer vertices). Multi-modal trip planning requires a consistent cost function which is applicable to all parts of the traffic network and thus to all modes of transportation.

Let us briefly highlight two currently very popular speed-up techniques for queries in road networks as well as public transportation networks. For a road network with static cost functions, contraction hierarchies (Geisberger et al., 2008) are a speed-up scheme that improves considerably upon the A^* algorithm and enables trip calculation with guaranteed optimality in large traffic networks at European scale within few milliseconds. By augmenting the original road network with so-called shortcuts in a preprocessing phase, the search space is restricted to a tiny fraction of the whole network, hence improving the query times by several orders of magnitudes compared to Dijkstra or any A^* variant. For public transportation networks, a very popular and powerful technique is that of so-called transfer patterns (Bast et al., 2010). In a preprocessing step, all possible sequences of transfers on optimal routes are precomputed and based on that a condensed graph structure is created which allows for the almost instant answering of source-target queries.

A comprehensive comparison of existing trip planning methods is provided in (Bast et al., 2015). Recent work incorporates user constraints in multi-modal trip planning (Dibbelt et al., 2015), in addition to their approach our method incorporates knowledge on regularly occurring congestions. The approaches in (Niu et al., 2015) and (Liebig et al., 2014) utilize predictions to avoid upcoming traffic hazards, but their method has no incorporation of user preferences nor multi-modality. In (Bast & Storandt, 2014) trip guidebooks are created which are suitable for a long period of time, e.g., "Take Bus 10 to the main station, from there take Tram 11 or 13 (whichever comes next) to your target station. Trip duration: 30 minutes. Frequency: every 20 minutes.'

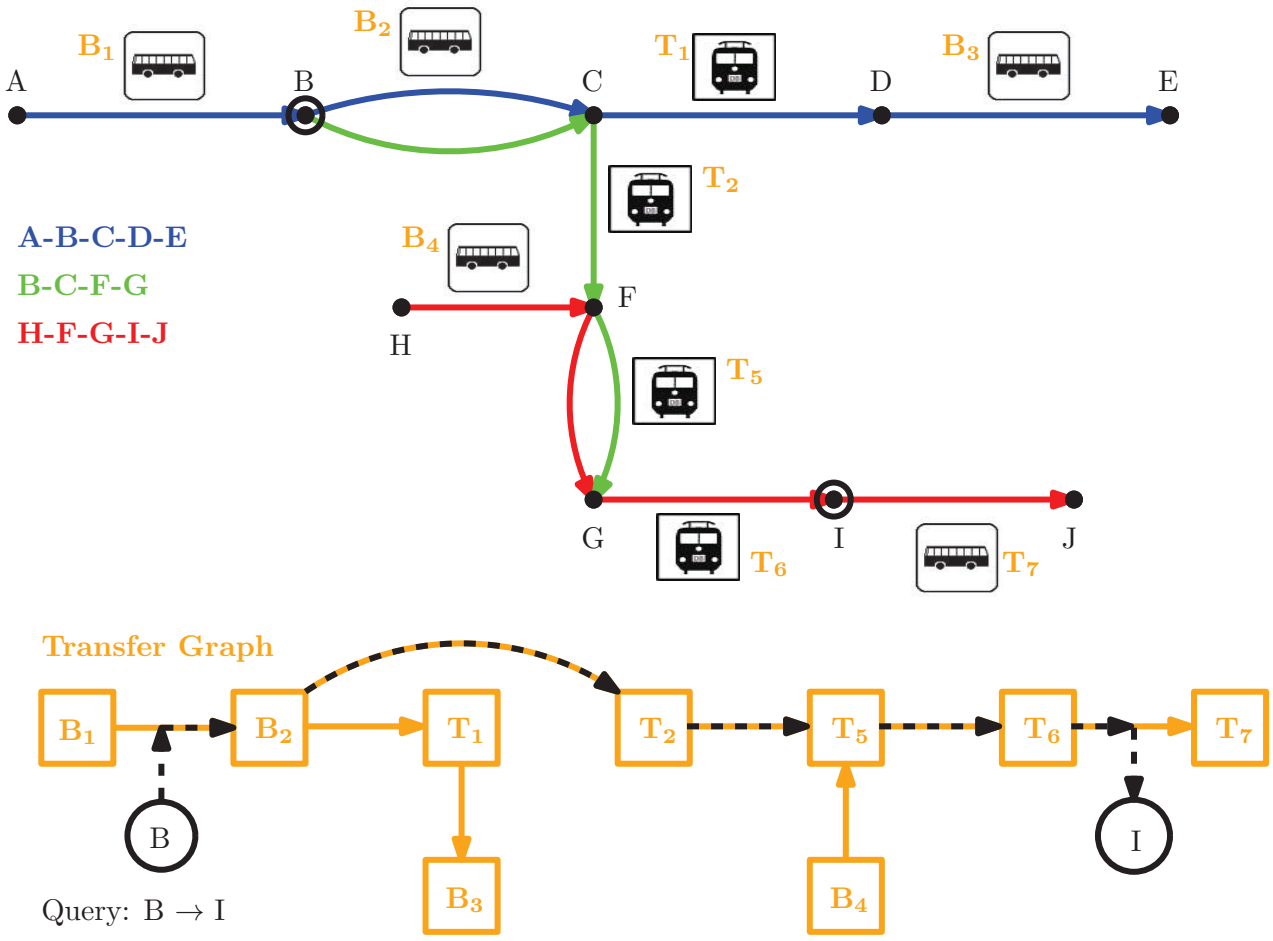


Figure 1. Stitch Network for multi-modal trip planning. Based on sourced trips (blue,green,red) nodes are travel legs and edges are uni-modal routes. An edge exists when a transfer between legs has actually been executed (and its popularity is counted). We query this graph to obtain a travel plan from B to I (black dotted line).

3. Socio-Paths routing method

In previous section we provided an introduction to trip planning and highlighted latest research for multi-modal and large-scale trip computation. But, as previously stated in the introduction, most of these approaches have the following shortcomings: (1) The computation is mostly based on static time-table and road network data. (2) It is hard to find the objective functions to produce routes people actually use. Current solutions either list all pareto-optimal routes, which is time consuming and results in a too large solution space for the end user, or, use an ad-hoc restriction to some routes without any validation via user experience. (3) Public transit timetable information is incomplete. (4) Even with complete timetable information, modelling transfer buffers or walking times is hard. (5) Often, also unknown factors like traffic congestion and overcrowding determine routes that people prefer. In practice these limitations anticipate delivery of route suggestions that fit to personal preferences (e.g. preference of train over bus) and constraints (e.g. seasickness) upon a trip calculation request.

To overcome these limitations, we propose a novel four-step method for stitching travel plans from previously recorded and bootstrapped routes. By utilization of these heterogeneous data sources the wisdom of multiple oracles (trip planners and prediction models) and local experts (e.g. via crowdsourcing) can be considered.

The four steps our method comprises are (1) Sourcing routes, (2) Constructing the transfer graph, (3) Adjust in case a departure time is specified, and (4) Adjust in case current traffic conditions make a transfer impossible are summarized in Algorithm 1. In the following we explain each step in more detail.

Algorithm 1 SocioPath Algorithm

Input: D = Source_Routes
Input: $hazards$,
Input: query: $(start, goal, constraints)$,
G_{source} =Construct_Transfer_Graph(D)
$G_{current}$ = Adjust_Transfer_Graph($G_{source}, hazards$)
G = Filter_Transfer_Graph($G_{current}, constraints$)
Compute_Route($G, start, goal$)

3.1. Sourcing routes

Our approach bases on some initially sourced routes. These routes are ideally real travelled routes that represent the expert knowledge of local experts, e.g., shortcuts that avoid congestions or possible connections among several means of public transport that are not stored in schedules and existing trip planners. This real-world data can be obtained in three ways: (1) via an active participation app for a persons smartphone (crowdsourcing), via a passive tracking system (e.g. cellular phone networks (Andrienko et al., 2013)) or via questionnaires (Janssens et al., 2012). Obviously this step processes sensitive data, as personal travel plans easily reveal individual habits and preferences of the person. Therefore these methods have to be designed such that re-identification is prohibited and no vulnerable data can be accessed by the system. Possible approaches for protection of individual data in this setting are (Boutsis & Kalogeraki, 2013) and (Liebig, 2015).

In case no real routes are available, or they do not provide sufficient coverage of the traffic network, routes can also be retrieved from existing route planners. This allows joining the information of various special-purpose or incomplete trip planners in a single system.

3.2. Constructing the transfer graph

Based on previously sourced routes a transfer graph is constructed that represents travel alternatives and possible changes of transport mode. The transfer graph G consists of edges E and nodes V. The nodes are travel legs, i.e. uni-modal routes. An edge $e \in E \subset V \times V$ among two nodes (v_i, v_j) exists when a transfer from the leg v_i to v_j has actually been executed (and its popularity is counted).

This transfer graph is queried to obtain a travel plan from location A to B. An example for the transfer graph construction is provided in Figure 1. In the figure the sourced trips are depicted in blue, green, and red. The corresponding multi-modal traffic network is in the upper part of the figure. At the edge the travel mode is depicted by small pictograms. Based on these routes and the corresponding traffic network the transfer graph is constructed as described above. The resulting transfer graph is shown in the lower part of Figure 1. Nodes of this graph are travel legs and edges are uni-modal routes. An edge exists iff a transfer between legs has actually been executed. Finally, we query this transfer graph to obtain a travel plan from location B to I in Figure 1. The resulting trip is marked by the black dotted line.

Possible additions to this process is, similar to transfer patterns (Bast et al., 2010), the annotation with concrete time information to provide time depending trip calculations. It is also easily possible to extend the criteria for path selection in the transfer graph according to the user preferences (including length or duration, number of transfers, price, robustness, waiting times, and, popularity) once these features were annotated at the nodes during previous sourcing of the routes.

3.3. Adjustments

The transfer graph, constructed in previous section, provides a useful data structure for trip computations from a set of initially given routes. Based on this graph a route can be stitched together from the information other people provided. The resulting route can be adjusted, once the exact travel time is given. This comprises two cases (1) if possible, validate all transfers in the route with the data (Find a route where that transfer was possible), and, (2) if no evidence is found that this transfer is possible, validate the transfers with timetable and road network data. This step leads to more data that may be added to the transfer graph, in a similar way as the initially sourced routes.

In case current traffic conditions make a transfer impossible, we temporarily "disable" that specific transfer node in the graph. In case of frequent problems, good alternatives should already be in the data and generated routes will avoid the regularly occurring transfer problem.

4. Conclusion and Future Work

In this position paper we sketched a novel idea for route planning based on routes people really used. The method can be bootstrapped using routes from ordinary route planners. We expect our approach to be particularly useful for route planning with special needs (e.g. disabled persons, bikers).

One could remark that the provided routes have no optimality guarantee and detours might be provided. However, if the graph construction was initially bootstrapped with ordinary trip planners or large sets of recorded routes this limitation will diminish. An open issue is that the proposed trip planner is deterministic and provides same output with same queries. Though this approach provides user-centric trip queries including individual preferences and constraints, guiding all persons selfishly to travel via some leg with limited capacity (e.g. a bus or a narrow

street) could lead to congestions (Roughgarden & Tardos, 2002). Future work therefore has to study how load balancing can be included directly in trip planning without causing too long detours for individuals, we will study usage of auction models (Dütting et al., 2012) for this problem.

Acknowledgments

This work results from a group discussion at Dagstuhl Seminar 13512. The authors received funding from the European Union's Seventh Framework Programme under grant agreement number FP7-318225, INSIGHT.

References

Andrienko, Gennady, Gkoulalas-Divanis, Aris, Gruteser, Marco, Kopp, Christine, Liebig, Thomas, and Rechert, Klaus. Report from dagstuhl: the liberation of mobile location data and its implications for privacy research. *ACM SIGMOBILE Mobile Computing and Communications Review*, 17(2):7–18, 2013.

Bast, H., Delling, D., Goldberg, A., Müller-Hannemann, M., Pajor, T., Sanders, P., Wagner, D., and Werneck, R. F. Route Planning in Transportation Networks. *ArXiv e-prints*, April 2015. URL http://arxiv.org/abs/1504.05140.

Bast, Hannah and Storandt, Sabine. Flow-based guidebook routing. In McGeoch, Catherine C. and Meyer, Ulrich (eds.), *2014 Proceedings of the Sixteenth Workshop on Algorithm Engineering and Experiments, ALENEX 2014, Portland, Oregon, USA, January 5, 2014*, pp. 155–165. SIAM, 2014. ISBN 978-1-61197-319-8. doi: 10.1137/1.9781611973198.15.

Bast, Hannah, Carlsson, Erik, Eigenwillig, Arno, Geisberger, Robert, Harrelson, Chris, Raychev, Veselin, and Viger, Fabien. Fast routing in very large public transportation networks using transfer patterns. In *Algorithms - ESA 2010, 18th Annual European Symposium. Proceedings, Part I*, pp. 290–301, 2010.

Boutsis, Ioannis and Kalogeraki, Vana. Privacy preservation for participatory sensing data. In *2013 IEEE International Conference on Pervasive Computing and Communications, PerCom 2013, San Diego, CA, USA, March 18-22, 2013*, pp. 103–113. IEEE Computer Society, 2013.

Delling, Daniel, Sanders, Peter, Schultes, Dominik, and Wagner, Dorothea. Engineering route planning algorithms. In *Algorithmics of Large and Complex Networks - Design, Analysis, and Simulation [DFG priority program 1126]*, pp. 117–139, 2009.

Dibbelt, Julian, Pajor, Thomas, and Wagner, Dorothea. User-constrained multimodal route planning. *J. Exp. Algorithmics*, 19:3.2:1.1–3.2:1.19, April 2015. ISSN 1084-6654. doi: 10.1145/2699886.

Dütting, Paul, Henzinger, Monika, and Weber, Ingmar. Maximizing revenue from strategic recommendations under decaying trust. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pp. 2283–2286. ACM, 2012.

Geisberger, Robert, Sanders, Peter, Schultes, Dominik, and Delling, Daniel. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In *Proceedings of the 7th International Conference on Experimental Algorithms*, WEA'08, pp. 319–333, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 3-540-68548-0, 978-3-540-68548-7.

Hart, Peter E., Nilsson, Nils J., and Raphael, Bertram. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems, Science, and Cybernetics*, SSC-4(2):100–107, 1968.

Janssens, Davy, Knapen, Luk, Körner, Christine, Monreale, Anna, Panis, Luc Int, Rinzivillo, Salvatore, Schulz, Daniel, Simini, Filippo, Ardanuy, Jesùs Fraille, Dhulst, Reinhilde, Pelekis, Nikos, and Theodoridis, Yannis. D1.1 report on big data available and privacy aspects. Technical Report D1.1, Universiteit Hasselt Transportation Research Institute, 8 2012.

Liebig, Thomas. Privacy preserving centralized counting of moving objects. In Bacao, Fernando, Santos, Maribel Yasmina, and Painho, Marco (eds.), *AGILE 2015*, Lecture Notes in Geoinformation and Cartography, pp. 91–103. Springer International Publishing, 2015. ISBN 978-3-319-16786-2.

Liebig, Thomas, Piatkowski, Nico, Bockermann, Christian, and Morik, Katharina. Route planning with real-time traffic predictions. In *Proceedings of the 16th LWA Workshops: KDML, IR and FGWM*, pp. 83–94, 2014.

Niu, Xiaoguang, Zhu, Ying, Cao, Qingqing, Zhang, Xining, Xie, Wei, and Zheng, Kun. An online-traffic-prediction based route finding mechanism for smart city. *International Journal of Distributed Sensor Networks*, 501:970256, 2015.

Roughgarden, Tim and Tardos, Éva. How bad is selfish routing? *Journal of the ACM (JACM)*, 49(2):236–259, 2002.

Modelling Time and Location in Topic Models

Christian Pölitz CHRISTIAN.POELITZ@TU-DORTMUND.DE
TU Dortmund University, Artificial Intelligence Group,
Otto Hahn Str. 12, 44227 Dortmund, Germany

Abstract

Many text collections like news paper or social media blog archives contain texts that often refer to special dates and/or locations. These information can be valuable to investigate topics in certain regions and time spans. We use topic models that integrate time and geographical information extracted from the texts to find such topics. In this extended abstract, we motivate our approach and shortly describe the method. Experimental evaluations and detailed description are in preparation to a full paper.

1. Introduction

Topic models (see for instance (Blei et al., 2003)) have been used extensively to summarize text collections into semantic clusters. Such text collections can contain for instance news paper articles, Blog entries, tweets or any written social media content. The documents in these collections contain often information about locations and dates. These information are valuable for extracting topics for certain regions or time spans. In order to integrate temporal and positional information, we need the corresponding time and location information for our text corpus. Previous approaches assumed that we either directly have information about time and position for each document in the corpus or that a named entity recognition tool finds geographic locations. We propose a hybrid approach that extends standard Latent Dirichlet Allocation (LDA (Blei et al., 2003)) topic models. We assume that the documents can have multiple dates and positions. In order to integrate multiple information for single documents, we propose to extract parts (or chunks) in the documents that contain only information about one location and one date. For example, a document might contain the sentence: "The weather was nice in Berlin last Sunday, but next day at home in Cologne it was cloudy.". In order to reflect all temporal and positional information, NLP tools would divide the sentence in: "The weather was nice in Berlin last Sunday" and "but next day at home in Cologne it was cloudy" and label the words in the first chunk with the time stamp of that corresponding Sunday and the geographical location of Berlin. The words in the second chunk are labeled with the time stamp of that corresponding Monday and the geographical location of Cologne.

Proceedings of the 2^{nd} International Workshop on Mining Urban Data, Lille, France, 2015.

2. Related Work

There are several previous approaches that integrate temporal and positional information into topic models. In (Yin et al., 2011) Yin et al. discuss methods to find and compare topics in documents that have associated GPS coordinate. Speriosu et al. propose in (Speriosu et al., 2010) to use topic models that use non-overlapping regions as latent topics. By this, they model each document as distribution over these regions. Further approaches use topic models with geographic information on social media data to extract activity patterns of users. Hasan and Ukkusuri for instance use in (Hasan & Ukkusuri, 2014) topic models that integrate sequences of activities rather than documents. In (Hong et al., 2012) Hong et al. introduce a sparse generative topic model and in (Kurashima et al., 2013) Kurashima et al. propose a geographic topic model that use Twitter tweets to extract user activities in terms of movement and interests.

3. Method

While the standard topic models group only words and documents in semantically related topics, we are further interested in the distribution of the topics over time and geographic position. In order to extract the distribution of word senses over time and positions, we use topic models that consider temporal information about the documents as well as locations in form of numerical vectors that represent the geographic position. This means, each document has at least one time stamp and one geographic position. The time stamps are assumed to be Beta distributed and the position Normally distributed. The two distributions are simply in-

tegrated in an LDA topic model under the assumption that given the latent topics, the words, the time stamps and geographic positions are independent. We combine the methods by Wang and McCallum (Wang & McCallum, 2006) introduced as topics over time and supervised LDA introduced by Blei et al. (Blei & McAuliffe, 2007).

The generative process of the words, dates and location is:

1. For each topic t:
   (a) Draw $\theta_t \sim Dir(\beta)$
2. For each document d:
   (a) Draw $\phi_d \sim Dir(\alpha)$
   (b) For each chunk $c(d)$:
      i. For each word i in $c(d)$:
         A. Draw $z_i \sim Mult(\phi_d)$
         B. Draw $w_i \sim Mult(\theta_{t_i})$
         C. Draw $t_c \sim Beta(\psi_{z_i})$
         D. Draw $l_c \sim N(\eta' \hat{z_{c(d)}}, \rho^2)$

Assuming a number of topics we draw for each of them a Multinominal distribution over words in this topic from a Dirichlet distribution $Dir(\beta)$ with metaparameter β. For each document we draw a Multinominal distribution of the topics in this document from a Dirichlet distribution $Dir(\alpha)$ with metaparameter α. For each word in the document we draw a topic with respect to the topic distribution in the document and a word based on the word distribution for the drawn topic. Additionally, we draw a time stamp $t_i \sim Beta(\psi_{z_i})$ with $\psi_{z_i} = (a, b)$ the shape parameters of the Beta distribution and the location $l_i \sim N(\eta' \hat{z_{c(d)}}, \rho^2)$ with $\hat{z_{c(d)}}'$ the empirical topic frequencies for document d. The shape parameters ψ are estimated by the method of moments. For each topic z we estimate the mean $\hat{m}$ and sample variance s^2 of all time stamps from the documents that have been assigned this topic. We set $a = \hat{m} \cdot (\frac{\hat{m} \cdot (1-\hat{m})}{s^2} - 1)$ and $b = (1 - \hat{m}) \cdot (\frac{\hat{m} \cdot (1-\hat{m})}{s^2} - 1)$ for each topic. Finally, for the Normal distribution, η is estimated via EM methods, that minimize the likelihood during the estimation of the topic model: $L(\eta) = -\frac{1}{2\rho} \sum_d (y_d - \eta' \hat{z_d})^2 - -\frac{1}{2\sigma} \sum_k \eta_k^2$

Integrating the time stamp as Beta distributed random variable and the geographic location as Normal distributed random variable, we get for the probability of a topic z_i, given a word w in a chunk $c(d)$ with time stamp t and location l and all other topic assignments:

$$
\begin{aligned}
& p(z_i | w, t, l, z_1, \cdots z_{i-1}, z_{i+1}, \cdots z_T) \\
& \propto \frac{N_{w,z_i} - 1 + \beta}{N_{z_i} - 1 + W \cdot \beta} \cdot (N_{d,z_i} + \alpha) \cdot \\
& \frac{(1 - t_{c(d)})^{a-1} \cdot t_{c(d)}^{b-1}}{Beta(a, b)} \cdot exp(-\frac{\|l_{c(d)} - \mu_{w,d}\|^2}{2\rho}) \qquad (1)
\end{aligned}
$$

Using this, we can estimate the topic model via Gibbs sampling.

References

Blei, David M. and McAuliffe, Jon D. Supervised topic models. In *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pp. 121–128, 2007.

Blei, David M., Ng, Andrew Y., and Jordan, Michael I. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003. ISSN 1532-4435.

Hasan, Samiul and Ukkusuri, Satish V. Urban activity pattern classification using topic models from online geolocation data. *Transportation Research Part C: Emerging Technologies*, 44(0):363 – 381, 2014. ISSN 0968-090X. doi: http://dx.doi.org/10.1016/j.trc.2014.04.003.

Hong, Liangjie, Ahmed, Amr, Gurumurthy, Siva, Smola, Alexander J., and Tsioutsiouliklis, Kostas. Discovering geographical topics in the twitter stream. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pp. 769–778, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1229-5. doi: 10.1145/2187836.2187940.

Kurashima, Takeshi, Iwata, Tomoharu, Hoshide, Takahide, Takaya, Noriko, and Fujimura, Ko. Geo topic model: Joint modeling of user's activity area and interests for location recommendation. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pp. 375–384, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1869-3. doi: 10.1145/2433396.2433444.

Speriosu, M., Brown, T., Moon, T., Baldridge, J., and Erk, K. Connecting Language and Geography with Region-Topic Models. 2010.

Wang, Xuerui and McCallum, Andrew. Topics over time: A non-markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pp. 424–433, New York, NY, USA, 2006. ACM. ISBN 1-59593-339-5. doi: 10.1145/1150402.1150450.

Yin, Zhijun, Cao, Liangliang, Han, Jiawei, Zhai, Chengxiang, and Huang, Thomas. Geographical topic discovery and comparison. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pp. 247–256, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0632-4. doi: 10.1145/1963405.1963443.

Evaluating distance measures for trajectories in the mobile setting

Nikolaos Larios NLARIOS@DI.UOA.GR
University of Athens

Christos Mitatakis CMITATAKIS@DI.UOA.GR
University of Athens

Vana Kalogeraki VANA@AUEB.GR
Athens University of Economics and Business

Dimitrios Gunopulos DG@DI.UOA.GR
University of Athens

Abstract

Mobile devices, such as smartphones allow us to use computationally expensive algorithms and techniques. In this paper, we study algorithms in order to solve the problem of finding the most similar trajectory within a number of trajectories. We built a framework that enables the user to compare a trajectory Q with trajectories that have been generated and stored on mobile devices. The system returns to the user the most similar trajectory based on the algorithm that has been selected. The algorithms for the measurement of the trajectory similarity have been implemented for mobile devices running Android OS. We evaluate our algorithms with real geospatial data.

1. Introduction

The study of the similarity between trajectories is important in a plethora of application domains (e.g. Traffic management, Video analysis, Molecular Design, Similarity of Meteorological Phenomena etc.). More specifically, the trajectory similarity between moving objects is useful for applications in intelligent traffic systems, such as traffic navigation and traffic prediction, both of which require mining of past trajectories patterns, etc. The identification of similarities between moving objects is a challenging task, since not only their locations change but also because their speed and semantic features vary. Mobile devices and smartphones, in particular are rapidly emerging as a dominant computing and sensing platform. They are equipped with a variety of sensors such as GPS, cameras and accelerometers. This gives birth to several unique opportunities for data collection and analysis such as finding trajectory similarities between trajectories generated and stored on distributed smartphones. We developed a framework that allows users to find similar trajectories in a distributed environment. Our framework requires the participation of a number of users in order to find a solution of a query. In order to specify the way in which our framework works, every query is assigned by a user of our system and it is addressed to all the available users of our platform. The goal of our platform is to generate and store trajectory data on mobile devices (smartphones) and compare a query trace Q against the crowd of trajectories that is distributed on a large number of mobile devices.

Proceedings of the 2^{nd} International Workshop on Mining Urban Data, Lille, France, 2015.

1.1. Related Work

There are many methods and techniques related to searching for similar moving object trajectories. Some previous methods were based on Euclidean distance space, but this is not suitable for road network space because is difficult to apply the distance of Euclidean space to road network space. Other methods considered only spatial similarity without considering temporal similarity to search for similar moving object trajectories. The methods that interest us the most are those that are used for measurement of the spatio-temporal similarity between trajectories. Although in this work we do not consider the privacy issues that come up when data from users become available in the system, there is work in the area that can be leveraged in a real system. In "Select-Organize-Anonymize"(Giorgos Poulis, 2013), the authors find similar trajectories, by using Z-ordering and data projections on subtrajectories. Then they

organize the selected trajectories into clusters which they anonymize after that. Another work that the researchers propose a method to compare trajectories of moving object is in "Shapes based trajectory queries for moving objects".(Lin & Su, 2005) In this work, the writers introduce a new distance function and the evaluate the similarity measurement by using a grid representation to describe trajectories. The most similar work is the paper Crowdsourced Trace Similarity with Smartphones paper (Zeinalipour-Yazti et al., 2013) where the writers try to solve efficiently the problem of comparing a query trace against a number of traces generated by smartphones. They developed the SmartTrace+ framework which consists a distributed data storage model that trajectories are stored and top-K query processing algorithms that exploit trajectory similarity measures, elastic to spatial and temporal noise.

1.2. Our Contributions

In this paper, we build a distributed mobile platform that uses algorithms to search for trajectory similarity among a number of trajectories that are stored in distributed mobile devices. The trajectories which are stored have been collected by monitoring the movement of the mobile devices. The users choose whether their movement will be recorded by the application. So our platform consists of two parts, the one is where the users gather the trajectory data, so that can be created a distributed database where the trajectories are stored.

The second contribution of our platform is the mechanism that given a query trajectory Q, we want to find any trajectories of the mobile devices using our platform that follow a motion similar to Q. We initiate an experimental evaluation of three different trajectory similarity measures, namely Dynamic Time Warping (DTW), Longest Common Subsequence (LCSS), and Fréchet distance. Intuitively, Dynamic Time Warping (DTW) is the equivalent of the L_2 distance when stretching of the trajectories is allowed since DTW takes into account the individual differences of all matched points in the stretched sequences. Similarly, Longest Common SubSequence (LCSS) can be thought of as the equivalent of L_0 since it counts how many elements are the same. Completing the analogy, the Fréchet distance is equivalent to L_{inf} since it considers the maximum of the individual differences of the matches in the stretched sequences. Here we perform an evaluation on their relative accuracy and performance.

2. Problem Definition

Trajectory: Trajectory or trace of a moving object is a set of consecutive positions in space as a function of time. Due to limitations on the acquisition and storing of data, is very difficult and expensive to accurately record an entire trajectory. In spatio-temporal databases a trajectory is represented as a set of discrete samples of the positions of the moving object of the form (x, y, t), i.e. a sequence of ordered pairs (x, y), each characterized by a timestamp. As described above our goal is to develop a distributed platform for spatio-temporal similarity search between trajectories generated and stored on distributed mobile devices. Specifically, given a query trajectory Q, we want to find any trajectories of the mobile devices using our platform that follow a motion similar to Q.

The mobile application should be able to store locally the trajectories that follow the mobile device, to receive a query Q from another mobile device, through the server of the platform, to evaluate if the mobile device has followed a similar trajectory to Q and return its result back to the specific user through the server again. The platform should be able to compare a query trajectory Q that is submitted by a user, simultaneously against a number of trajectories that is stored on a large crowd of mobile devices. The most similar trajectory of every user with the query Q should be received and managed from the server application ensuring the privacy of the users of the platform. Then the server finds the most similar trajectory within the results that have been collected and sends it back to the user who submitted the query Q. Our approach exploits the fact that nowadays, mobile devices have exceptional computational power and storing capabilities. Running the algorithms in a distributed mobile environment grants our platform the necessary performance and scalability in order to support a great number of users and spatio-temporal data. Moreover, having the trajectories stored in the mobile devices the privacy of the users is protected. The definition of the term similarity between trajectories may vary between different problem cases. For example in most cases the most similar trajectory should be determined by using not only the spatial shape of the trajectories but also their evolution in time. In other cases the timing of the recording of the trajectory or the time intervals between points of interest that are defined by the query may be of greater importance.

3. System Overview

The system has two basic functionalities. The first one consists from the monitoring procedure, where the user can monitor his movement. With this procedure, the users trajectories are stored in the mobiles database creating the necessary data. The second functionality contains the query submission from one user and the search for results, using a selected algorithm, in the devices registered in the system. The architecture of our system is described by figure 1.

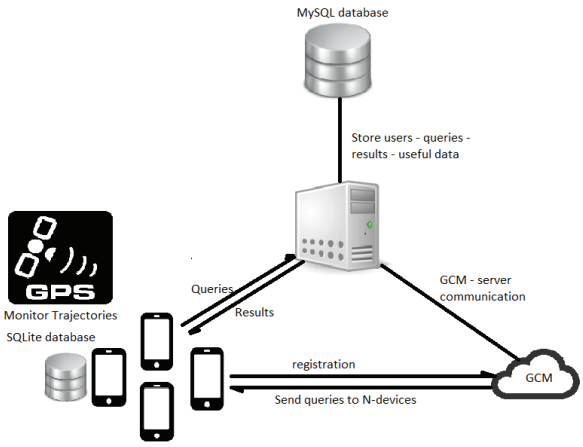


Figure 1. System's architecture

3.1. System Components

The goal of the system is to record and store efficient (in time and resources) large amounts of data from N mobile devices and support the communication between them and a main server in order to send queries on the data recorded and send their results. The way that data are stored must be defined, i.e. the structures that will be used in mobile devices and in the server for data storage. Our system consists by the following: A main application server, which is responsible for receiving and forwarding queries between mobile devices using the application. Furthermore, it is responsible for the registration of mobile devices in the system. This application is hosted on an Ubuntu server. As mentioned above in the system will be registered N mobile devices using the application's mobile platform which is responsible for data recording, sending queries, receiving and handling queries other devices and send data if the query result was successful.

3.2. Users

The system consists of N users and those users participate in it through their devices (mobile, tablets, etc.). For the efficient operation and acceptable results of the system, it requires a sufficient number of users. The more users register to our platform the precision of the system will be increased due to the available set of data for each geographic area and time. User data are generated by the sensors of the devices and are stored locally on each device. Since users are not constantly connected to the system due to intermittent connectivity for their mobile devices respectively no data will be offered to our system without their agreement.

3.3. Data Storage

Each device (mobile, tablet, etc.) can produce tuples of data through its sensors (eg GPS sensors, accelerometer, camera, microphone, etc.).The data that are stored in the mobile devices corresponds to the trajectories that have been performed by the device while the user have selected to record his movement.

The form and amount of data depend on the application of the general form to is <Id, Latitude, Longitude, Timestamp>where:

- The Id refers to the unique code that corresponds to the current trajectory that is recorded. Every time the user selects to monitor his movement, a new id is initialized which define the recording.
- The Latitude and Longitude determine the geographical location of the tuple.
- The Timestamp refers to the time of recording the current tuple.

Examples of data form is: <1, 23.32134, 37.566643, 2014-10-24 16:52:01>

Every trajectory consists of a number of tuples like the example above. Each tuple is recorded periodically every 2 seconds. This period can be changed accordingly to our preferences or the storage capabilities of the device. The trajectories that are recorded are stored in the devices internal memory. For the storage of data in the mobile devices is used SQLite which is the default SQL database engine for android powered devices.

3.4. Querying component

The Android application we have developed is responsible for the collection and storing of trajectory data by recording the movement of the mobile devices. Moreover, the algorithms that we use for the measurement of trajectory similarity have been implemented in the Android application.

A query Q when is formed by a user of our platform is forwarded by the server to other registered mobile devices. When a mobile device receives a query it runs the selected algorithm and searches the data that has been collected for the most similar trajectory, according to the selected algorithm. Let $m_1, m_2, ..., m_i$ denotes a set of i mobile devices. The server sends the query Q to this set of mobile devices where Q is a sequence of points $\{p_1, p_2, ..., p_j\}$ that describes a trajectory. The application compare the trajectory Q with each trajectory that is stored in the database of the device (e.g. $\{T_1, T_2, T_3, ..., T_k\}$), in order to find the

most similar one. All devices that have a result sends their answers to the server.

3.5. Server

The server of our platform is the middle-ware responsible for the communication between the mobile devices and stores vital data in its database. The server application forwards the queries of the users to the mobile devices which are registered to our platform. Then it receives the results of each device and the best one, according to the algorithm's results, is sent back to the user who submitted the Query.

4. Trajectory Similarity Measures

In this section we review the trajectory similarity algorithms we use in the system to find the most similar trajectories to the query. **Dynamic Time Warping:** The Dynamic Time Warping (DTW) (Donald J. Berndt, 1994) is an algorithm for measuring similarity between two temporal sequences which may vary in time or speed. The DTW is widely used in the comparison of time series, appropriately aligns the sequence of points of the two rails, so that the total distance as a sum of individual distances is minimized.

$$DTW(P_{1..n}, Q_{1..n}) =$$
$$|P_n - Q_n| + min \begin{cases} DTW(P_{1..n-1}, Q_{1..m-1}) \\ DTW(P_{1..n-1}, Q_{1..m}) \\ DTW(P_{1..n}, Q_{1..m-1}) \end{cases}$$

where P1 .. n-1 the subsequence P1 .. n that include elements (points) for time periods of 1 up to n-1.

Longest Common SubSequence: The Longest Common SubSequence (LCSS) (Michail Vlachos, 2002)problem is to find the longest subsequence common to all sequences in a set of sequences (often just two). The LCSS algorithm, using dynamic programming fits best points of the two trajectories based on a tolerance parameter time and a tolerance parameter space ε. It considers that the points do not exceed the tolerance parameters fit and attaches similarity value equal to 1. If they do not match they are assigned the value 0. Specifically, the LCSS distance between two real-valued sequences S1 and S2 of length m and n respectively is computed as follows:

$$Lcss(P_{\delta,s}(S_1, S_2)) =$$
$$\begin{cases} 0, if \quad n = 0 \quad \text{or} \quad m = 0 \\ 1 + Lcss_{\delta,s}(HEAD(S_1), HEAD(S_2)) \\ max(Lcss_{\delta,s}(HEAD(S_1), S_2), \\ Lcss_{\delta,s}(S_1, HEAD(S_2))) \\ otherwise \end{cases}$$

where HEAD(S1) is the subsequence $[S_{1,1}, S_{1,2}, ..., S_{1,m-1}]$ and δ is an integer that controls the maximum distance in the time axis between two matched elements and is a real number $0<\varepsilon<1$ that controls the maximum distance that two elements are allowed to have to be considered matched. One drawback of this measurement is that it ignores distance gaps between subsequences, that allows to obtain some points of the track when alignment. Consequently, it can lead to inaccuracies in the similarity analysis.

5. Fréchet Distance

Given two curves, A, B in a metric space, the Fréchet distance, $d_F(A, B)$ is defined as

$$d_F(A, B) = \inf_{\alpha,\beta} \max_{t\in[0,1]} \{d(A((t)), B((t)))\}$$

where , range over all monotone reparameterizations and d(,) represents the Euclidean distance, and inf is the infimum. The discrete Fréchet distance d_F between two polygonal curves $a : [0, m] \to R^k$ and $b : [0, n] \to R^k$ is defined as:

$$d_F(a, b) =$$
$$\min_{\substack{\sigma:[1:m+n]\to[0:m], \\ \beta:[1:m+n]\to[0:n]}} \max_{s\in[1:m+n]} \{d(a(\sigma(s)), b(\beta(s)))\}$$

where σ and β range over all discrete non-decreasing onto mappings of the form $\sigma : [1 : m + n] \to [0 : m], \beta : [1 : m + n] \to [0 : n]$.

We first consider the corresponding decision problem. That is, given $\delta > 0$, we wish to decide whether $\delta_F^+(A, B) \leq \delta$. Consider the matrix M as defined in the subsection 5.1. In the two-sided version of Discrete Fréchet Distance with Shortcuts (DFDS), given a reachable position (a_i, b_j) of two pointers, the A-pointer can make a skipping upward move, as in the one-sided variant, to any point $a_k, k > i$, for which $M_{k,j} = 1$. Alternatively, the B-pointer can go to any point $b_l, l > j$, for which $M_{i,l} = 1$; this is a skipping right move in M from $M_{i,j} = 1$ to $M_{i,l} = 1$, defined analogously. Determining whether $\delta_F^+(A, B) \leq \delta$ corresponds to deciding whether there exists a sparse staircase of ones in M that starts at $M_{1,1}$, ends at $M_{m,n}$, and consists of an interweaving sequence of skipping upward moves and skipping right moves (see Figure 2)..

5.1. Basic Algorithm

The implementation of the algorithm of Fréchet Distance relied on the publication "The Discrete Fréchet Distance with Shortcuts via Approximate Distance Counting and Selection" (Anne Driemel),(Rinat Ben Avraham, 2014).

Specifically, the algorithm which was implemented is the two side - DFDS who faces the problem of outliers which is sensitive the Fréchet Distance. The result of the algorithm is the lowest Fréchet distance which satisfies the two trajectories. The pseudocode of the algorithm is represented in Algorithm 1. The steps of the basic algorithm are the following:

1. Implementation of binary search and sequential executions of the algorithm of Fréchet Distance until the optimal distance e is found. The initial value which is given to the middle in the binary search is 0.025. The binary search is terminated when the distance between low and high values of the middle is smaller than 0.001.

2. Then the table is created, whose columns correspond the points (latitude, longitude) of trajectory A and the lines the points of the trajectory B. The values taken by the M matrix is either 0 or 1 depending on the distance apart of these two points together, and are given by the following formula:

$$E_{\delta}^{+} = \\ ((a_i, b_j), (a_k, b_j))|k > i, ||a_i - b_j||, ||a_k - b_j|| \leq \delta \\ \cup((a_i, b_j), (a_i, b_l))|l > j, ||a_i - b_j||, ||a_i - b_l|| \leq \delta$$

3. Forthwith after, the directed graph G is created, based on the table M which was formed above. The nodes of the graph are the positions of the table M. Correspondingly it is created an edge between two nodes if the nodes are in the same row or column of the table and their values are 1 (from the previous to the next).

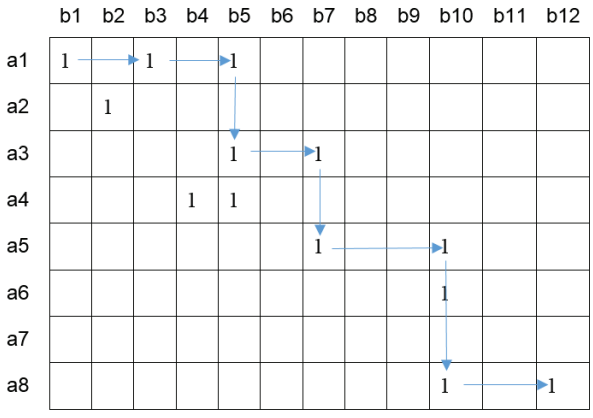


Figure 2. M table

4. In the figure 2, the algorithm Shortest Path is performed to see if there is a path from the first position of the table (0,0) to the last (N, M) wherein N and M are the dimensions of the two trajectories which were compared. If the algorithm of Shortest Path has a solution we check if the value of is between the limits we have set. In case it is the binary search terminates. Otherwise, it will perform again with latest prices. If not the execution of the algorithm terminates.

6. Evaluation

6.1. Fréchet Distance Performance

To measure the performance of the algorithm of the Fréchet Distance we performed experiments comparing time series with length ranged from 20000 to 100000 points. From the paper of Rinat Ben Avraham et all know that the complexity of the algorithm of Fréchet Distance with Shorcuts (DFDS) is $O((m^{2/3}n^{2/3} + m + n)log^3(m + n))$. The main difference with the previous version of the algorithm is that we calculate table T and simultaneously create also the graph G. In addition we set a parameter δ which represents the percentage of table points that we wish to compare. In this way, we improved the time execution of the algorithm of Fréchet Distance. Also due to the sampling of δ points in all time series, the complexity of the algorithm is reduced to linear.

The figure below shows the performance of the implementation of the Fréchet Distance algorithm compared with the length of the trajectories. In our experiment that is summarized by figure 3 we compared 50 trajectories with lengths from 20,000 points to 100,000 points. This experiment was performed 10 times and for each length of trajectories it was chosen the average execution time. Also, the chart shows the dispersion of runtime for any length of time series on error bars. From figure 3, we can conclude that the complexity of the algorithm is linear.

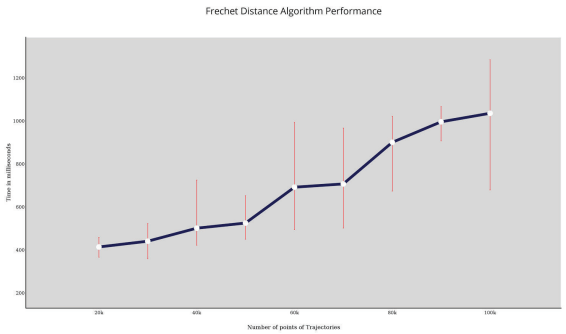


Figure 3. Frechet performance

Algorithm 1 Fréchet Distance

Input: trajectory t_i, trajectory $query$,int delta
Binary Search
repeat
  $rightTable[0..trajectory.length][0..1] = null$
  $downTable[0..trajectory.length][0..1] = null$
  for $i = 1$ **to** $t_i.length$ **do**
    $start \leftarrow i - delta$
    if $start < 0$ **then** start $\leftarrow 0$
    $stop \leftarrow i + delta$
    if $start > query.length$ **then** stop $\leftarrow$ query.length
    for $j = start$ **to** $stop$ **do**
      if $x_i > x_{i+1}$ **then**
        $M[i][j] \leftarrow 1$
        if firstExecution **then**
          if M[0][0] == 0 **then**
            $return$
          else
            $graph.addVertex(0, 0)$
          end if
          $firstExecution \leftarrow$ false
        end if
        if rightTable[i][0] != ll and rightTable[i][1] != null **then**
          $graph.addVertex((i, j))$
          $graph.addVertex((rightTable[i][0], rightTable[i][1]))$
          $graph.addEdge((i, j), (rightTable[i][0], rightTable[i][1]))$
          $rightTable[i][0] \leftarrow i$
          $rightTable[i][1] \leftarrow j$
        else
          $rightTable[i][0] \leftarrow i$
          $rightTable[i][1] \leftarrow j$
        end if
        if downTable[i][0] != null and downTable[i][1] != null **then**
          $graph.addEdge((i, j), (downTable[i][0], downTable[i][1]))$
          $downTable[i][0] \leftarrow i$
          $downTable[i][1] \leftarrow j$
        else
          $downTable[i][0] \leftarrow i$
          $downTable[i][1] \leftarrow j$
        end if
      else
        $M[i][j] \leftarrow 0$
      end if
    end for
  end for
  // Find the shortest path if exists on graph from the
  // first to the last position of table M
  $graph.findShortestPath()$
until $shortestPath$ is $false$ and $frechetDistance \in$ BinarySearchRange

6.2. Algorithm Comparison

In this section we show preliminary experimental results that compare the three different trajectory similarity methods we have described. The focus of our evaluation is to show that all methods can be used in the limited resources environment of a smartphone. For this reason we also compare the run time results with the simpler (easier to implement and efficient to run) Euclidean distance that serves as a basic comparison point. It is difficult to compare the three methods because they have been implemented with different optimizations, and also our Fréchet distance implementation computes only a bound, and necessitates the use of several runs to compute the exact Fréchet distance. We use two different datasets for the comparisons. We also show a sample result of a query and the most similar answers that we get using the three methods.

The datasets which were used in the experiments are from the chorochronos.org and Dublin Bus GPS sample data from Dublin City Council (Insight Project). The parameters with which we will compare the algorithms performance are their time execution and their results. The name of the dataset from chorochronos.org is Trucks and contains 50 trajectories. The name of the dataset from Insight project is Siri and contains 30 bus trajectories. For the experiments, the first trajectory of the dataset was used as a query and compared with the remaining 49 trajectories. The experiments were performed for trajectories with length 2048 points each.

At the end of each experiment the program returns the graphs of execution time of each algorithm and the most similar trajectory chosen by each algorithm. The algorithms we compare are:

- Dynamic Time Warping (DTW)
- Longest Common Subsequence (LCSS)
- Fréchet Distance: DFDS to find the smaller Fréchet distance. In order for DFDS to find the smaller Fréchet distance, the algorithm should be executed multiple times for each trajectory that is compared as the algorithm described above suggests. (DFDS)
- Fréchet Distance - one execution of the algorithm (DFDS one time)

Bellow we present the results from the data provided by Insight Project. This dataset containts trajectories recorded from Dublin buses across Dublin City. In comparison with 2048 points per trajectory, the execution time of the algorithms is presented in **figure 4**.

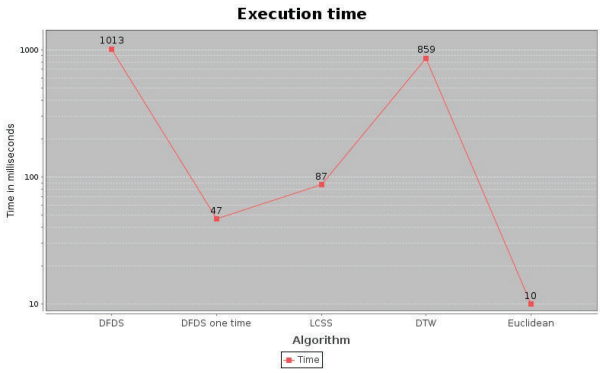


Figure 4. Algorithms execution time for 2048 points (Dublind Bus GPS sample)

The results of each algorithm that correspond to the most similar trajectory are listed below.

In **figure 5** is the most similar trajectory according to Discrete Fréchet Distance algorithm.

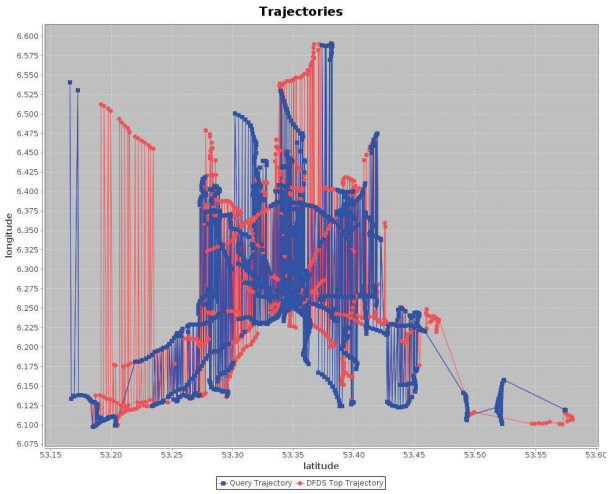


Figure 5. DFDS most similar trajectory for 2048 points (Dublind Bus GPS sample)

In **figure 6** is the most similar trajectory according to LCSS algorithm.

In **figure 7** is the most similar trajectory according to DTW algorithm.

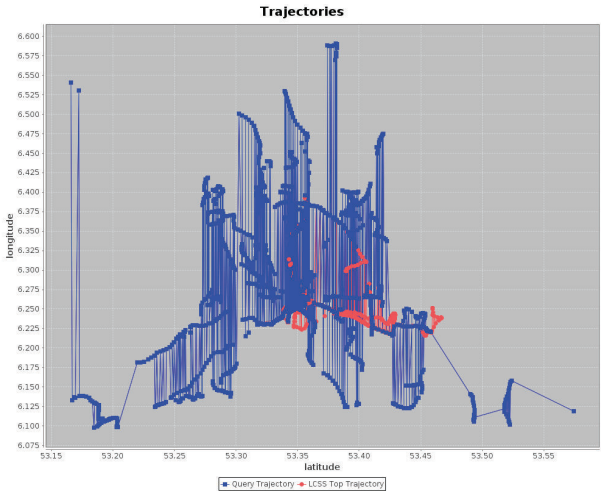


Figure 6. LCSS most similar trajectory for 2048 points (Dublind Bus GPS sample)

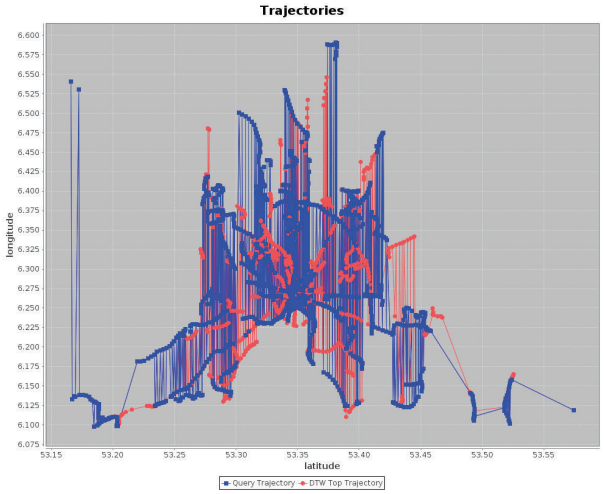


Figure 7. DTW most similar trajectory for 2048 points (Dublind Bus GPS sample)

We conducted the same experiments with the dataset provided by chorocronos.org. Specifically with trajectories that contain 2048 points. In comparison with 2048 points per trajectory the execution time of the algorithms are presented in **figure 8**.

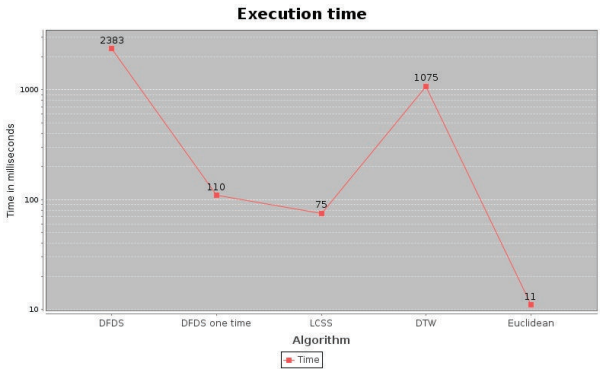


Figure 8. Algorithms execution time for 2048 points (chorochronos.org)

The results of each algorithm that correspond to the most similar trajectory are listed below.

In **figure 9** is the most similar trajectory according to Discrete Fréchet Distance algorithm

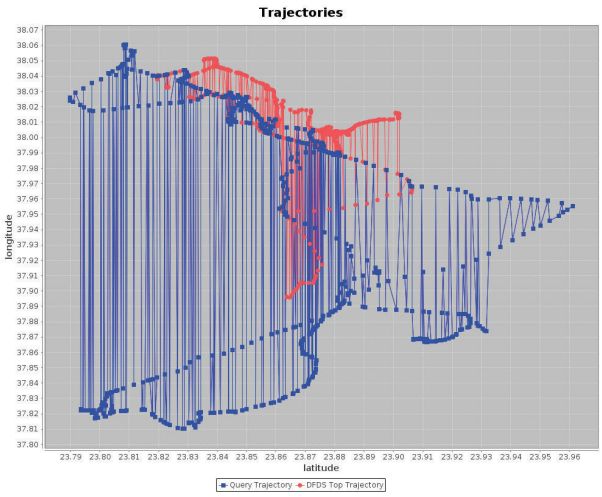


Figure 9. DFDS most similar trajectory for 2048 points (chorochronos.org)

In **figure 10** is the most similar trajectory according to LCSS algorithm

In **figure 11** is the most similar trajectory according to DTW algorithm

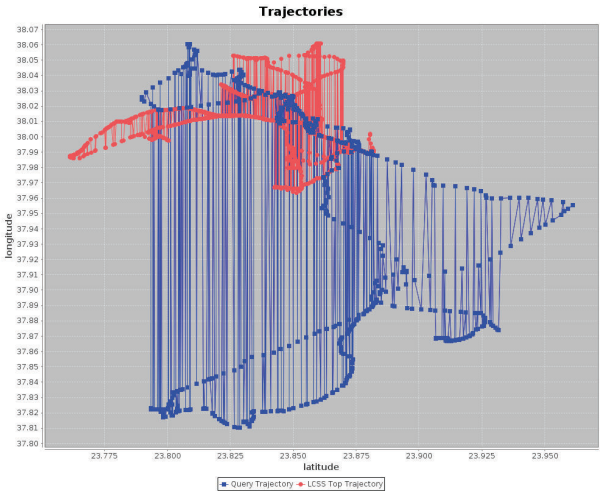


Figure 10. LCSS most similar trajectory for 2048 points (chorochronos.org)

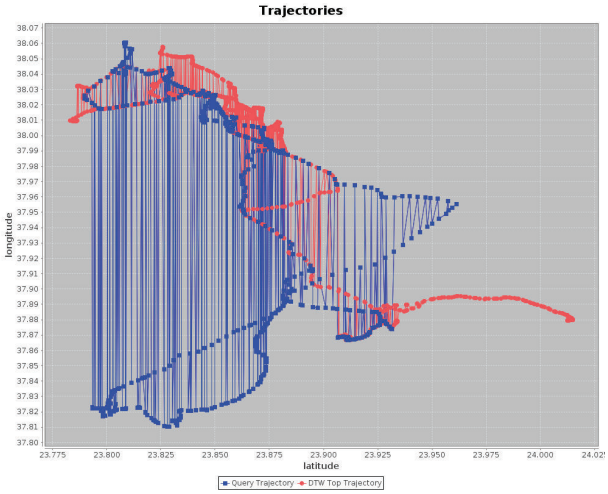


Figure 11. DTW most similar trajectory for 2048 points (chorochronos.org)

We observe that the distance measurement that have been implemented (Fréchet Distance) have similar execution time with the implementation of the LCSS algorithm. However, the final results of the algorithm differ. The execution time of a single run of Fréchet Distance algorithm is similar to the LCSS's algorithm execution time. The results of Fréchet Distance algorithm are based on the similarity of the shape of each trajectory in contrast with the other algorithm which are restricted to specific points comparison. Furthermore, we avoid the problem of outliers that Fréchet distance is sensitive to, thereby improving signif-

icantly the final results. Thus, we see that in comparison that we are interested more at having a similar shape rather than smaller distance of each point from the points of the query trajectory, Fréchet Distance produces better results than the other algorithms. Hence, Fréchet Distance is ideal for similarity measurement of the shape of trajectories This conclusion can be valuable for later implementations and research.

7. Conclusion

In conclusion, the trajectory similarity search problem have a plethora of applications, fact that gives the platform described in this work great potentials. Our system allows the user to send a query that contains a trajectory in order to receive a number of the most similar trajectories that have been recorded by other mobile devices that are registered to our system. We described the algorithms that the mobile application uses in order to measure the similarity between trajectory and finally search for the most similar ones. Moreover, the distributed architecture of our system grants it great capabilities such as scalability. We have evaluated our platform and the implemented algorithms using real spatio-temporal data. Our experiments prove the satisfying performance of our implementation. We compared each algorithm that we implemented and we evaluate both their performance and their final results according to the similarity between the selected trajectory and the query. We extracted useful conclusions about each algorithm's functionality and results, such as the different cases where the Fréchet distance algorithm is more suitable than the other distance measures. To sum up the final system that was implemented gives a number of final responses to queries submitted by its users, according to the selected algorithm in a satisfying time. A future goal is to extend the number of distance measurements that are implemented in order for the system to have more reliable and accurate final results.

8. Acknowledgements

We would like to thank Demetris Zeinalipour-Yazti, "University of Cyprus" for his assistance by providing us the source code of the implementation of LCSS algorithm. This research was supported by the ARISTEIA MMD, FP7 INSIGHT, ERC IDEAS NGHCS and THALIS GeomComp projects.

References

Anne Driemel, Sariel Har-Peled, Carola Wenk. *Approximating the Frchet Distance for Realistic Curves in Near Linear Time*.

Donald J. Berndt, James Clifford. Using dynamic time warping to find patterns in time series. In *KDD Workshop*, pp. 359–370. AAAI Press, 1994.

Giorgos Poulis, Spiros Skiadopoulos, Grigorios Loukides Aris Gkoulalas-Divanis. Select-organize-anonymize: A framework for trajectory data anonymization. In *Data Mining Workshops (ICDMW), 2013 IEEE 13th International Conference on*, pp. 867 – 874, Dallas, TX, 2013. IEEE.

Lin, Bin and Su, Jianwen. Shapes based trajectory queries for moving objects. In *Proceedings of ACM GIS*, pp. 21–30. IEEE, 2005.

Michail Vlachos, Dimitrios Gunopulos, George Kollios. Discovering similar multidimensional trajectories. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pp. 673 – 684, San Jose, CA, 2002. IEEE.

Rinat Ben Avraham, Omrit Filtser, Haim Kaplan Matthew J. Katz Micha Sharir. The discrete frchet distance with shortcuts via approximate distance counting and selection. In *SOCG'14 Proceedings of the thirtieth annual symposium on Computational geometry*, pp. 377, New York, NY, USA, 2014. ACM.

Zeinalipour-Yazti, Demetrios, Laoudias, Christos, Costa, Constandinos, Vlachos, Michail, Andreou, Maria I., and Gunopulos, Dimitrios. Crowdsourced trace similarity with smartphones. In *EEE Transactions on Knowledge and Data Engineering (TKDE '13), IEEE Computer Society*, pp. 1240–1253, Los Alamitos, CA, USA, 2013. IEEE.