Available online at www.sciencedirect.com

ScienceDirect

Transportation Research Procedia 31 (2018) 67–73

Transportation Research Procedia

www.elsevier.com/locate/procedia

45th European Transport Conference 2017, ETC 2017

Automatic Transport Network Matching Using Deep Learning

Manuel Martin Salvador[a,b], Marcin Budka[a,*], Tom Quay[b]

[a]*Bournemouth University, Bournemouth, United Kingdom*
[b]*We Are Base, Bournemouth, United Kingdom*

Abstract

Public transport users are increasingly expecting better service and up to date information, in pursuit of a seamless journey experience. In order to meet these expectations, many transport operators are already offering free mobile apps to help customers better plan their journeys and access real-time travel information. Leveraging the spatio-temporal data that such apps can produce at scale (i.e. timestamped GPS traces), opens an opportunity to bridge the gap between passenger expectations and capabilities of the operators by providing a real-time 360-degree view of the transport network based on the 'Apps as infrastructure' paradigm. The first step towards fulfilling this vision is to understand which routes and services the passengers are travelling on at any given time. Mapping a GPS trace onto a particular transport network is known as 'network matching'. In this paper, the problem is formulated as a supervised sequence classification task, where sequences are made of geographic coordinates, time, and line and direction of travel as the label. We present and compare two data-driven approaches to this problem: (i) a heuristic algorithm, which looks for nearby stops and makes an estimation based on their timetables — used as a baseline — and (ii) a deep learning approach using a recurrent neural network (RNN). Since RNNs require considerable amounts of data to train a good model, and collecting and labelling this data from real users is a challenging task, one of our contributions is a synthetic journey data generator. The datasets that we generated have been made as realistic as possible by querying real timetables and adding position and temporal noise to simulate variable GPS accuracy and vehicle delays, sampled from empirical distributions estimated using thousands of real location reports. To validate our approach we have used a separate dataset made of hundreds of real user journeys provided by a UK-based bus operator. Our experimental results are very promising and our next step is to deploy the solution in a production environment. From the operator's point of view, this will enable multiple smart applications like account-based ticketing, identification of disruptions, real-time passenger counting, and network analysis.

Keywords: network matching; deep learning; machine learning

* Corresponding author
E-mail address: mbudka@bournemouth.ac.uk

10.1016/j.trpro.2018.09.053

1. Introduction

Public transport users are increasingly expecting better service and up to date information, in pursuit of a seamless journey experience. In order to meet these expectations, many transport operators are already offering free mobile apps to help customers better plan their journeys and access real-time travel information. Leveraging the spatio-temporal data that such apps can produce at scale (i.e. timestamped GPS traces), opens an opportunity to bridge the gap between passenger expectations and capabilities of the operators by providing a real-time view of the transport network based on the 'Apps as infrastructure' paradigm. The first step towards this vision is to understand which routes and services the passengers are travelling on at any given time.

Mapping a timestamped GPS trace onto a particular transport network is known as the 'network matching' problem (Zheng et al. (2008)). This is related to the 'map matching' problem (also known as 'snap to road'), where the aim is to fit a sequence of GPS coordinates to a network of roads (Bernstein and Kornhauser (1996)). In this paper, we define the network matching problem as supervised sequence classification (Graves (2012)), where a machine learning model is trained with labelled sequences of elements. These elements represent geographic coordinates, date and time, while a combination of line and direction of travel forms the label.

We present and compare two data-driven approaches to this problem: (i) a heuristic algorithm, which looks for nearby stops and makes an estimation based on their timetables, and is used as a baseline, and (ii) a deep learning approach using a recurrent neural network (RNN), which to the best of our knowledge, is a novel approach to the network matching problem.

RNNs require considerable amounts of data to train a good model for complex problems (Sutskever (2013)). Collecting and labelling this data from real users is a challenging task (e.g. asking too often can be overwhelming, there are privacy concerns related to collecting GPS location (Stopher (2008)), the labels can be unreliable due to mistakes or misuse). One of our contributions is a synthetic journey data generator. The datasets that we generated have been made as realistic as possible by querying real timetables and adding spatial and temporal noise to simulate variable GPS accuracy and vehicle delays, sampled from empirical distributions estimated using thousands of real location reports.

To validate our approach we have used a separate dataset containing hundreds of real user journeys provided by a UK-based bus operator. Our experimental results are very promising and our next step is to deploy the solution in the production environment. From the operator's point of view, this will enable multiple smart applications like account-based ticketing, identification of disruptions, real-time passenger counting, and network analysis. Passengers will, therefore, benefit from a better service and an increase in the quality of information due to leveraging such big data processing.

The organisation of the paper is as follows. Problem description and related work are presented in Section 2; the heuristic algorithm used as baseline is presented in Section 3; Section 4 describes the deep learning approach we implemented; the data generator for training our RNN models is explained in Section 5; the experimental setup is described in Section 6 and results are in Section 7; finally, the paper concludes in Section 8.

2. Related work

The network matching problem has been addressed before in Zheng et al. (2008) and Bolbol and Cheng (2013) but they only consider distinguishing between different modes of transport (bus, car, cycle, train, tube). Moreover, those studies do not take actual time into account but only an ordered sequence of locations. In our study, we investigate the problem of matching a timestamped GPS trace to a particular route of a transport network.

We have defined the problem as supervised sequence classification (Graves (2012)). Let S_{train} be a set of training examples made of pairs ($\mathbf{x}$, c), where $\mathbf{x} = (x_1, x_2, \ldots, x_T)$ is a finite sequence of real-valued vectors and c is the label of that sequence. The task is to train a sequence classification algorithm $\boldsymbol{h}: \boldsymbol{X} \rightarrow \boldsymbol{C}$ to label the sequences in a test set S_{test} disjoint from S_{train}. In our context, $\mathbf{x}$ is a sequence made of timestamped GPS coordinates while c is the combination of line name and direction of travel (e.g. '2-inbound').

3. Heuristic algorithm

A single bus stop can be served by one or more lines at different times of the day. Bus departure and arrival information are provided in the form of timetables. Our heuristic approach creates a probability table by inspecting the timetables of the nearest bus stops of a given GPS trace. Algorithm 1 contains the pseudocode for estimating the line number and direction of a bus using a sequence of GPS coordinates. This algorithm does not require any training as it is able to infer the label by inspecting the timetables.

A known drawback of this approach is the way the stops are selected. The first GPS observation is likely to be in the vicinity of a bus stop since the passenger whose mobile phone is providing the location data is boarding the bus. However, the rest of the observations are taken at periodic intervals, so it is less likely that they are in the close vicinity of some bus stop.

This method tends to be slow as it has to query the timetables of each candidate line. Therefore, the execution time grows linearly with the number of candidate lines. We use this method as a baseline.

Input:

X := Sequence of coordinates {lat, long} (size of **X** is in range [1, 60])

T := Sequence of datetimes for each pair of coordinates

Constant input:

K := Lines of the network

D := Directions of travel (i.e. inbound and outbound)

Algorithm 1:

1. **N** := List of nearest bus stops for each pair of coordinates from **X**. That is, **N**[i] contains the nearest bus stop to **X**[i].
2. **P** := Prior probabilities for each line of the network. That is, **P**[k] = number of stops in **N** served by line k divided by the size of **N**.
3. **L** := Candidate lines from **P** selected as follows:
    1. The line(s) in **P** with the highest probability.
    2. Any line in **P** with probability over 20% (this threshold was selected experimentally to maximise the accuracy).
4. **M** := Matrix of size |**L**|x|**D**| in which each cell contains the number of bus stops from **N** matching a possible journey in the timetable of each candidate line and direction of travel.
5. **U** := Matrix of size |**L**|x|**D**| in which each cell contains the number of journeys within the timeframe **T** in the timetable of each candidate line and direction of travel.

Output:

Pair {line, direction} that maximise the formula $(\omega + \mu)*\rho$ where ω is the number of stops in **M**, μ is number of candidate journeys in **U**, and ρ is the prior probability in **P**. More than one pair is output in case of ties.

4. Deep learning approach

Traditional machine learning algorithms for classification like Logistic Regression or Support Vector Machines have been designed to handle fixed-sized vector inputs. Although such an approach is possible in the network matching problem by always using a fixed number q of most recent GPS coordinates (e.g. the last 10 locations), it has a number of drawbacks. On one hand, such models are unable to classify sequences shorter than q, and on the other, they are ignoring potentially useful information beyond this fixed horizon. Selecting the value q is hence a trade-off between how flexible the model will be and how well it will perform.

For these reasons, we have decided to look at approaches specifically designed for handling variable-sized sequential input data. Recurrent neural networks (RNNs) are probably the best-known models of this type (Graves et al. (2006)), and although they have been around for quite some time (Hochreiter and Schmidhuber (1997)), they have only gained momentum in recent years due to a number of successes in applications like speech recognition (Graves et al. (2013)), natural language processing (Sak et al. (2014)) or machine translation (Cho et al. (2014)).

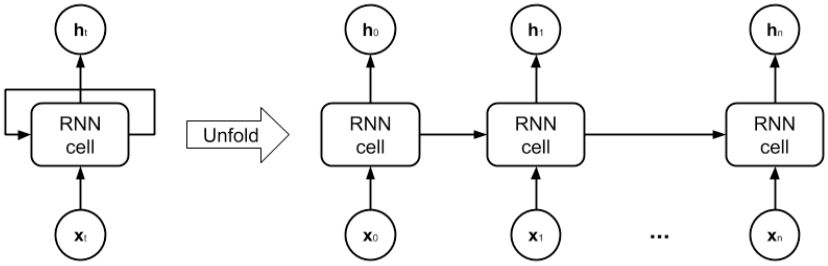

Figure 1. RNN unfolding in time

The basic idea behind RNNs is very simple - repeatedly apply the same transformation to each new datapoint x_t in a loop and output a corresponding hidden state h_t, which is then passed on as an additional input to the next iteration. This can also be thought of as "unfolding" the transformation in time, depicted in Figure 1.

Depending on the inner workings of the RNN cell there exist a number of RNN models, with the two most popular being GRU (Gated Recurrent Unit) (Chung et al. (2014)) and LSTM (Long Short Term Memory) (Hochreiter and Schmidhuber (1997)). In order to improve accuracy, it is common to stack multiple cells together to form a so called multicell, and to interweave those with dropout layers (Srivastava et al. (2014)) for improved generalisation (i.e. accuracy on new unseen data), as shown in Figure 2.

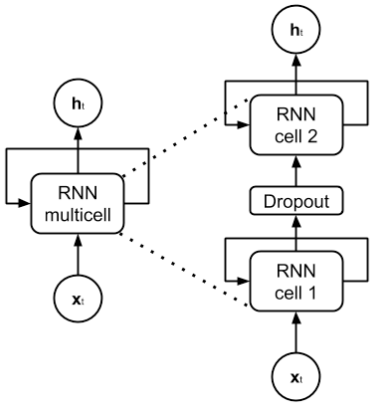

Figure 2. Multicell RNN

5. Synthetic data generator

A limitation of deep neural networks including RNNs is that they require a large amount of labelled data to train a good model. Unfortunately, collecting labelled real journeys is rather difficult since it requires users' permission to track their smartphones and also to manually enter what line they are on.

In order to generate data, we first create a sampling space of all possible journeys within the network. Then, we randomly select a set of journeys from that space. For each journey, we generate a sequence of timestamped GPS coordinates within the path made between bus stops in that journey.

To make the sequence more realistic, we add random normal noise to the locations — based on an empirical distribution estimated from thousands of real location reports — and a random bus delay of up to 15 minutes.

This generator allows us to create datasets of any size, and can also be used as an API to get an infinite stream of sequences on demand.

6. Experimental setup

The experiments have been carried out with data from a public transport operator in the UK. The network is made of about 140 buses serving 23 lines. The total number of classes is 46 (23 lines x 2 directions).

The heuristic approach doesn't require any learning phase. On the other hand, RNN is trained with over 15 million sequences of synthetic journeys, with the length varying between 5 and 60. Every two consecutive points of the sequence are one minute apart. The RNN has been implemented using Google's TensorFlow 1.3 (Abadi et al. (2015)) and trained on NVIDIA GeForce 1080 and Titan X GPUs. We have tested a range of hyperparameter values, including: (1) the RNN cell type (GRU and LSTM), (2) the number of cells in a multicell (i.e. the number of layers, between 1 and 5), (3) the cell size (256, 512 and 768), and (4) various encodings of the time information (as minute of the week and as embeddings for day of the week, hour and minute). We have used softmax as the top layer of the network to obtain a probability distribution over all possible line/direction pairs, cross-entropy loss, and the Adam optimiser (Kingma and Ba (2015)) with a range of learning rates and the remaining settings left to their default values.

As this is a classification problem, the measure we are trying to minimise is the classification error on a test set. Overlapping of lines could mislead this measure as there are some situations in which it is equally probable that a sequence belongs to multiple lines. In fact, for this particular network, 37% of stop to stop segments are shared by at least two lines. Thus, we also consider the classification error on the second and third best prediction.

Both approaches have been tested with the same dataset made of 164 real journeys with length varying between 6 and 56 observations (see histogram in Figure 3). Some of these observations represent walking or even standing periods since tracking lasts up to 1 hour from the beginning of the journey. We haven't filtered those out to make the data more realistic.

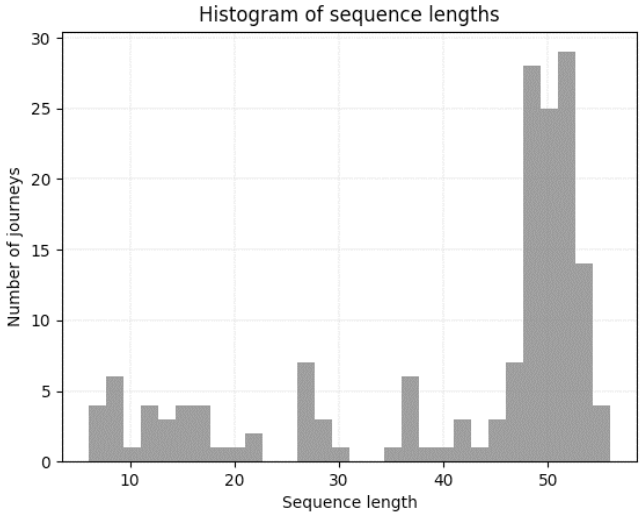

Figure 3. Histogram of sequence lengths from 164 real journeys

7. Results

Figure 4 shows the classification accuracy on successive test sets from the data generator over the training period of RNN models. The three plots represent the fraction of times the actual line was in top 1 (accuracy), top 2 and top 3 most probable lines according to our model. Models trained with LSTM cells (blue lines) show more stable behaviour over GRU cells (red lines). Some GRU-based models show a drop of performance at certain iterations, which makes GRU a less robust choice for this problem. We don't yet understand the reason for this behaviour but we plan to investigate it in a follow-up study.

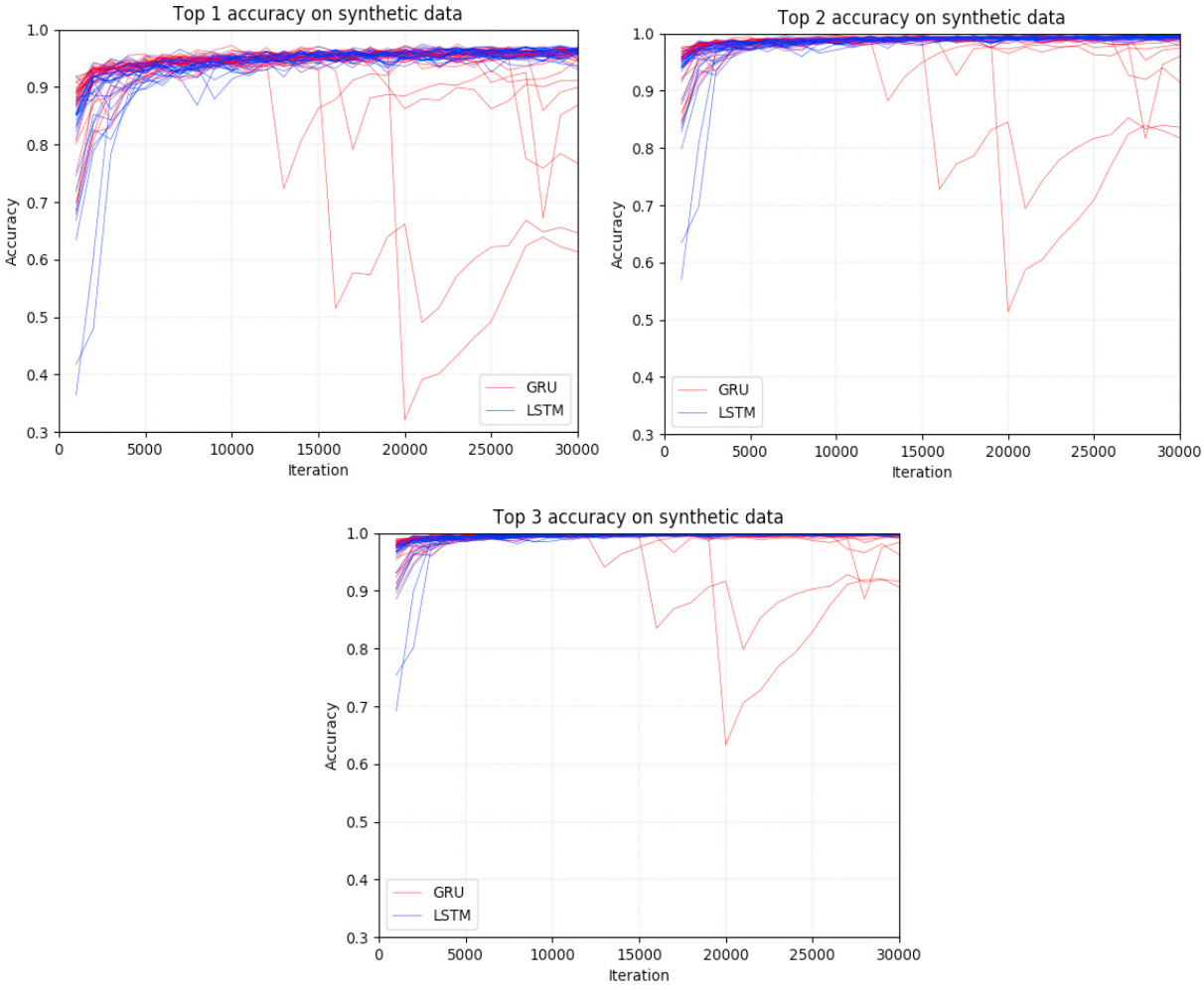

Figure 4. Prequential test accuracy on synthetic data: Top 1 (left); Top 2 (right); Top 3 (bottom)

Results of both heuristic algorithm and best RNN models with LSTM and GRU cells are shown in Table 1. We vary the maximum sequence length (τ) to find approximately the minimum number of points needed to make acceptable predictions. The highest accuracy we achieved was 67.68% from the GRU model when τ = 30. Best top 2 and top 3 scores were 87.80% and 92.68% from LSTM model also when τ = 30. In all cases, RNN models significantly outperform the heuristic baseline.

Table 1. Classification results on the real test set.
τ = max sequence length (can be seen as minutes). δ = improvement ratio with respect to heuristic * The largest sequence has 56 observations.

	Classification accuracy (%)														
	Heuristic			**RNN**											
				LSTM						**GRU**					
τ	**Top 1**	**Top 2**	**Top 3**	**Top 1**	δ	**Top 2**	δ	**Top 3**	δ	**Top 1**	δ	**Top 2**	δ	**Top 3**	δ
5	27.69	27.69	36.92	39.51	1.43	61.11	2.21	74.07	2.01	38.89	1.40	62.35	2.25	75.31	2.04
10	33.55	33.55	54.84	56.44	1.68	79.14	2.36	88.34	1.61	53.99	1.61	76.07	2.27	85.89	1.57
20	39.35	39.35	58.71	65.85	1.67	84.15	2.14	92.07	1.57	65.85	1.67	84.76	2.15	90.85	1.55
30	44.81	44.81	62.99	66.46	1.48	**87.80**	1.96	**92.68**	1.47	**67.68**	1.51	85.98	1.92	92.07	1.46
*56	38.56	38.56	56.21	62.20	1.61	84.76	2.20	90.24	1.61	65.85	1.71	84.76	2.20	90.24	1.61

8. Conclusion

This paper has presented two data-driven approaches to automate the transport network matching problem. Taking into account the peculiarities of the network and quality of the collected data, the classification results are very promising (up to 92.68% on top 3 predictions).

The lack of timetables reliability (Martin-Salvador et al. (2017)) suggests that our models can be further improved by incorporating historical records of real-time information during the training process. That additional information coming directly from the vehicles would help to enhance our training data with (i) GPS points following the roads between two bus stops (instead of a straight line) and (ii) buses running earlier than scheduled.

From the user data collection point of view, sampling every minute may not offer enough resolution for distinguishing between some buses running closely. We would like to experiment with different sampling rates to increase the journey resolution without compromising the phone battery life.

Acknowledgments

This work has been developed as part of Innovate UK partnership project KTP010097 between Bournemouth University and We Are Base. We would also like to thank NVIDIA Academic Program for providing the GPU.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mane, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viegas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015) "TensorFlow: Large-scale machine learning on heterogeneous systems". Software available from tensorflow.org

Bernstein D. and Kornhauser A. (1996) "An Introduction to Map Matching for Personal Navigation Assistants". Princeton University, pp. 1-17

Bolbol, A. and Cheng, T. (2013) "Matching GPS Data to Transport Networks". 21st GIS Research UK, 44, pp. 1-11

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014) "Learning phrase representations using RNN encoder-decoder for statistical machine translation". EMNLP 2014

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014) "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". NIPS 2014 Workshop on Deep Learning

Graves, A. (2012) "Supervised Sequence Labelling with Recurrent Neural Networks". Studies in Computational Intelligence 385, pp. 1-131

Graves, A., Fernandez, S., Faustino, G., and Schmidhuber, J. (2006) "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks". International Conference on Machine Learning, pp. 369-376

Graves, A., Mohamed, A., and Hinton, G. (2013) "Speech recognition with deep recurrent neural networks". ICASSP 2013, pp. 6645-6649

Hochreiter, S. and Schmidhuber, J. (1997) "Long Short-Term Memory". Neural Computation, 9 (8), pp. 1735-1780

Kingma, D.P., and Ba, J. (2015) "Adam: A Method for Stochastic Optimization". ICLR 2015

Martin-Salvador, M., Budka, M., Quay, T. (2017) "How accurate is real-time passenger information?". Passenger Technology Group, pp. 1-10

Quddus, M.A., Ochieng, W.Y., Zhao, L., and Noland, R.B. (2003). "A general map matching algorithm for transport telematics applications". GPS Solutions, 7(3), pp. 157-167.

Sak, H., Senior, A.W., and Beaufays, F. (2014) "Long short-term memory recurrent neural network architectures for large scale acoustic modeling". INTERSPEECH 2014, pp. 338-342

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., (2014) "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". Journal of Machine Learning Research, 15, pp. 1929-1958

Stopher, P.R. (2008). "Collecting and Processing Data from Mobile Technologies". ISCTSC.

Stopher, P.R. and Greaves, S.P., (2007). "Household travel surveys: Where are we going?". Transportation Research Part A: Policy and Practice, 41(5), p. 367–381.

Sutskever I. (2013) "Training Recurrent Neural Networks". University of Toronto, pp. 1-101

Zheng, Y., Liu, L., Wang, L., and Xie, X. (2008) "Learning Transportation Mode from Raw GPS Data for Geographic Applications on the Web". 17th international conference on World Wide Web, pp 247-256