ANALYSIS, RANKING AND PREDICTION IN PERVASIVE COMPUTING TRAILS

D. Papadogkonas, G. Roussos, M. Levene

School of Computer Science and Information Systems,
Birkbeck College, University of London,
London WC1E 7HX, U.K.
Email: {dikaios,gr,mark}@dcs.bbk.ac.uk

Keywords: pattern recognition, prediction, ranking

Abstract

Many pervasive computing applications involve the recording of user interaction with physical and digital resources in the environment. Such records can be used to establish context histories that can be subsequently used for user behaviour analysis, pattern recognition, prediction, and the provision of context aware services. In this paper we use trails as the principal data processing primitive for analysis and prediction. We define a trail as the sequence of recorded interactions with the pervasive computing space. Trails contain patterns of space usage and they can be used for the provision of different services, space usage analysis or sociological information of people using the environment simultaneously. Trail analysis requires considerable storage and computational resources to discover such patterns. Moreover no single method exists that identifies significant trails based on different metrics for a variety of different pervasive computing application. In this paper, we introduce a trail based analysis approach, an associated model for the representation of trails and trail aggregates, and suitable data structures for efficient storage, filtering and retrieval. Also, we propose several related algorithms and associated metrics for ranking and identifying significant trails. We use these techniques in 2 different case studies to extract valuable information about the pervasive system environment usage and evaluate the summarizability and the predictive power of our model.

1 Introduction

Pervasive computing systems often provide facilities to record interactions between different devices or physical objects and users. In some cases, such recording is purposeful in that the aim of the research is to identify and analyze human social behaviour or understand utilization of the particular environment monitored. For example, Reality Mining [3] and Wireless Rope [12] attempt to understand interactions between users and fixed locations. Other projects, for instance Senseable City [17], use these interactions to analyze and describe the way we use cities. Moreover, in some cases the aim is not only to analyze but rather to affect user behaviour, for example in Cityware [13] researchers try to develop tools for deploying pervasive computing systems based on the relationship between space and user behaviour by recording Bluetooth devices at specific locations. All projects face common problems in understanding the captured data and analyzing these series of interactions in order to understand the systems usage or provide context aware services. Even though these and other projects work on the analysis of series of interaction in pervasive computing space no unified approach exist that allows system usage analysis, pattern recognition and prediction for a plethora of different applications under one probabilistic model.

In this paper, we address the need for such a unified approach which allows the analysis of different pervasive system datasets. We propose a trail based probabilistic data model and a collection of algorithms which can be used to understand the use of space in a pervasive computing environment identify patterns and make predictions. A unique feature of our model is that it can extract patterns based on different metrics. The model can consider and weight different metrics that can be recorded from the pervasive system environment. These metrics can be time of interaction or any other kind of sensor reading. The techniques proposed are general and can be employed at any level of abstraction and incorporate whatever types of user or service interactions are deemed appropriate.

We develop our approach based on the notion of landmark which we take to be the position of a significant entity within a landscape or any type of wireless resource a user can interact with. We capture interactions between users and landmarks by observing wireless communication between a user device and a device embedded in a landmark, although our methods do not depend on the specifics of the technology used and can cater for RFID, Wi-Fi, Bluetooth, or any other type of wireless sensor network technology.

We organize series of interactions with landmarks into *trails* which contain both spatial and temporal information. In particular, we record the duration of each interactive session for each user and each landmark, the distance between user and landmark and possibly the orientation if this is supported by the technology. All of which we subsequently use to calculate some of our proposed outputs for analysis, ranking and prediction.

Our selection of trail records is not coincidental since trails have been used as the basis for coordination between humans for centuries in different forms. For example, navigation trails provide route information and record information about paths to potential destinations. Aggregating multiple trails acquired over time across a particular environment is the technique humans often use to develop complete maps of a particular landscape and subsequently assist navigation, especially in the context of exploration [5]. Oral trails are also quite common in human coordination and are best represented as narratives which are replayed and recast repeatedly to incorporate new knowledge [11].

We process trails to extract common patterns of behaviour using our software engine which implements our unified model and associated data structures. The logical flow of our systems is presented in Figure 1). The engine consists of a parser and a sectioning tool for reading data collected from the pervasive computing environment and fragmenting them into a suitable format that represent trails; a probabilistic tree data structure which represent the recorded trails. The query engine is a framework for the definition of metrics used to calculate the best route under different circumstances during navigation; and associated mechanisms that can calculate such routes efficiently. Finally the results can be viewed by using an appropriate interface.

In section 2 we present related work, section 3 presents our trail representation model, in section 4 the query engine we use to query model and in sections 5 and 6 we discuss the experiments we conducted.

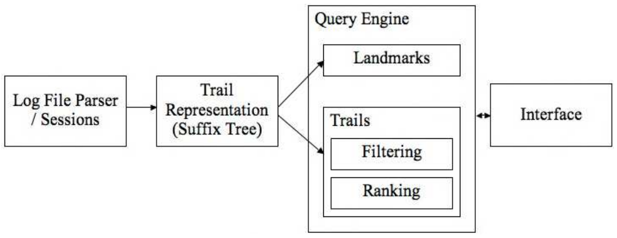

Log File Parser / Sessions
Trail Representation (Suffix Tree)
Query Engine
Landmarks
Trails
Filtering
Ranking
Interface

Figure 1) Architecture of the Ranking Engine

2 Related Work

There are a number of pervasive computing systems which allow data analysis, pattern recognition and prediction. A project closer related to our work is the MIT Reality Mining project [3]. In reality mining notion of *eigenbehaviours* is introduced. Eigenbehaviours use data from many subjects for a series of days. From these recordings a set of vectors is calculated that describes human behaviour between a group of people and it is also possible to construct behaviours of individuals based on these eigenbehaviors. With these techniques their subsequent behaviours could be predicted with up to 90% accuracy based on a weighted sum of users primary eigenbehaviors. Even though this is a highly accurate prediction the experiment was conducted from a small number of different states of behaviour, *{Home, Elsewhere, Work, No Signal, Off}*.

The ideas described in this paper are highly influenced from the area of web navigation. Trails in web navigation have been used to provide user specific applications (e.g. advertising) and an in depth view of a web page's usage. The same principles apply to pervasive computing systems where the interaction of users with objects is interpreted the same way as users following web page links. Many researchers have focused on different data models to represent the sequence of http requests in efficient data models. For this type of data mining problem the main techniques used are different types of Markov models, clustering and association rules or these techniques combined.

As proven by different real word tests in [16] [7] Markov chain structures have predictive capabilities and are suited for prefetching pages, targeting ads and personalization but the accuracy is limited to sort patterns. In [1] a system is described where a Hypertext Probabilistic Grammar is used where the prediction is made by the n previously visited pages. Other researchers use clustering techniques to simplify and enhance the accuracy of their models. Pitkow et al. [15] infers a Markov model from a collection of longest repeating sub-sequences, another system [2] uses different browsing sessions which are categorized and then each cluster is a state in the Markov chain model. Lan and al. [9] propose a system where association rules are used to identify dependencies between pairs of documents. Another work that is using association rules is described in [9] similar to [10] but it is related to web server caching.

3 Trail Representation

To store this interaction history between pervasive system objects we need to introduce a directed graph where vertices represent sensor network nodes and edges represent paths between these nodes. Two vertices (nodes) are said to be connected when there is a corresponding interaction record that indicates that the two landmarks have been visited in sequence by at least one user. For the remainder of the discussion we will use the terms landmarks, nodes and vertices interchangeably. Similarly we will use the terms edges, paths, links or trails to represent the connection between landmarks.

The links between landmarks are always directed and the landmarks are weighted with different usage metadata. Example of the metadata fields can include a unique id for the user, a timestamp representing when the user came into range or interacted with an object, a timestamp representing when the user went out of range, a positive integer representing frequency of visitation, the distance between the user and landmark during their interactive session, the orientation of the user in relation to the sensor node, etc. Higher order metadata can also be used, for instance the compound probability that a link will be followed given the user has arrived to a specific node following a particular path of fixed length within the network.

It should be noted that not all landmarks will be capable of providing all this information, nor does it make sense to store all the possible metadata fields for all types of landmarks. In any case, calculating the weights from the raw system logs requires considerable computational effort and poses several challenges in reconciling and ordering the log records. An example of the challenges involved is the fact that a sensor network is likely to be heterogeneous and with only approximate time synchronization.

Our probabilistic data structure differs from a Markov chain from the fact that probabilities are calculated based on a number of previous interactions and this does not comply with the Markov property. In terms of this graph, which is central to our approach trails, are represented as sequence of nodes. For efficiency the graph is stored as a probabilistic suffix tree [18] enhanced with metadata needed to encapsulate different information and metrics relevant to each interaction [14]. An example of such a representation is shown in Figure 2. The choice of this data structure has been guided by our desire to develop a system that can maintain all captured information while being able to rapidly respond to a great variety of queries, virtually being capable of responding to requests about any number of possible times, space and semantics related criteria.

4 Query Engine

The query engine is responsible for extracting patterns based on user defined metrics from the probabilistic data representation. The data related information is based into two major criteria. Queries related to the usage of Landmarks and queries based on the interaction trails.

4.1 Landmarks

An advantage of the probabilistic suffix tree data structure is that from the first level nodes we can extract all the available information that is related on specific landmarks. Because of the suffix tree properties, the root node will have as children one instantiation of each landmark and each of these nodes will contain the entire history of the landmark usage. This property exists due to the incremental addition of trails during the tree update process. Taking this suffix tree property as an advantage we can extract information related to a specific user's interaction with the landmark or overall statistics based on different metadata stored in the nodes.

4.2 Trail Ranking

A trail represented by a sequence of interactions between nodes is said to be significant if it satisfies one or more of the following criteria:

- It is one of the top n trails in respect of trail popularity.
- It is one of the top n trails in respect of average time (or some other temporal statistical measure) spent interacting with the landmarks in the trail.
- It is one of the top n trails in respect of the relevance of the landmarks to some chosen semantics (for example related to a specific spatial sub-area of the physical space that carries some possibly arbitrary user-defined significance).
- It is one of the top n trails in respect of one of the above criteria in a chosen time period
- It is one of the top n trails in respect of one of the above criteria for a chosen team sub-grouping.

These criteria request different subsets of the dataset so different filters are used to extract a subtree of the overall data structure which is later ranked.

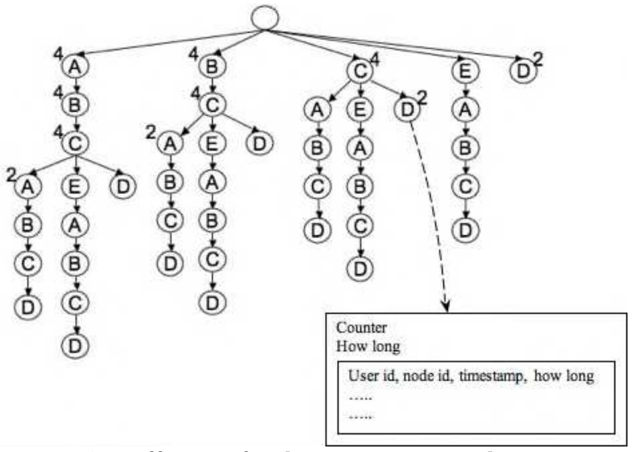


Figure 2) Suffix tree for the ABCABCD and BCEABCD trails with an example of the information associated with a node. The numbers next to the nodes represent the counters. Each node is also associated with additional metadata used to answer and rank significant trail queries

Trail Frequency: The probability of a trail T = is given by $T = n_1, n_2, ..., n_{t-1}, n_t$ is given by:

$$P(T) = \frac{count(T)}{count(root(T))}$$

Where $count(T)$ denoted the number of times that the trail T was followed, i.e. its count in the suffix tree, and $root(T)$ is the root node of the suffix tree. Note that $count(root(T))$ is the total count of all landmarks in the tree, i.e. the sum of counts of the first level nodes in the tree. Because there are nodes that finish in the middle of other trails we have to add dummy nodes with counter equal to the counter of the parent node minus the sum of the counters of the children nodes, in order to get the probability of the sum of the leaf nodes of the suffix tree equal to one.

The *pruned suffix tree* is defined to be a sub-tree of the suffix tree, whose root is the same as $root(T)$, the root of the original suffix tree. Let S be a pruned suffix tree, and let $\{T_1, T_2, ..., T_m\}$ be the set of all trails in S such that each trail T_i corresponds to a trail in S starting from its root and ending at one of its m leaf nodes. The probability of S is thus

$$P(S) = \sum_{i=1}^{m} P(T_i) \quad (1)$$

Where $P(T_i)$ is given according to the counts of the original suffix tree.

Now, let $\lambda, 0 < \lambda \le 1$ be a parameter, denoting the probability mass we require in the pruned suffix tree. In the extreme case when $\lambda = 1$, the pruned suffix tree will be equal to the original suffix tree, but when $\lambda < 1$ it becomes more selective in the trails that it contains. We now give the pseudo-code of an algorithm, designated POPULAR TRAILS (T, λ) that returns a pruned suffix tree, S, whose probability equal to or just above λ, where T is the suffix tree built from the log file.

```
[POPULAR TRAILS(T , $\lambda$)]
1. **begin**
2. S = root(T );
3. probs := 0;
4. **mark** root(S) as visited;
5. **while** probs < $\lambda$ do
6.      **let** n_i be a node that has not yet been visited in
7.           T and such that the trail T_i in T starting from
8.           root(T ) and ending at n_i has the highest
9.           probability $P(T_j)$ according to (1) among
10.          all trails $T_j$ starting from root(T ) and ending
11.          at a node $n_i$ in T that has not been visited;
12.     **mark** $n_i$ in T as visited;
13.     **add** $n_i$ as a leaf node to S;
14.     probs := P(S) according to (1);
15. **end while**
16. **return** S;
17. **end**.
```

Time: Like the frequency trail algorithm we define the normalized time of a trail $T = n_1, n_2, ..., n_{t-1}, n_t$ by

$$N(T) = \frac{Time(T)}{Sum(Times(LeafNodes))}$$

where $Times(T)$ denotes the time spent on trail T and $Sum(Times(LeafNodes))$ is the sum of time spent stored in the leaf nodes which is the total amount of time users spent interacting with the landmarks. Again for $\lambda, 0 < \lambda \le 1$ denoting the mass, we can use a modified version of the Popular Trails algorithm to calculate best trails by the time spent on them.

5 Model Evaluation

From our experimental evaluation we present a series of different outputs that give an understanding of the pervasive system environment usage. Moreover we present the summarizability of the model, the significance of patterns based on different metrics and their consistency in a dataset of considerable size. Finally we want to evaluate the predictive power of our model.

5.1 Summarizability

To measure the sumarisability of our model we use two different techniques, the spearman footrule [4] with a distance parameter, which measures the distance between two ranked lists and the overlap between two lists. For our experiments we will extract the top-m trails, of a given length sorted by a trail weight or other query constraints and each trail is represented as a list of landmarks where each position in the list represents the place of the landmark in the trail.

The footrule metric is defined as follows. For two lists of traiks L_1 and L_2 both with *m* number of elements and L is the union of this list. We have functions $f(i)$ and $g(i)$, where $i \in L$ that return the ranking of a trail i in the list and if, for example, i does not exist in L_1 then $f(i) = m + 1$ The footrule metric now is

$$F(L_1, L_2) = 1 - \frac{\sum_{i \in L} |f(i) - g(i)|}{MAX}$$

5.2 Predictive Power

To investigate the predictive power of the model we have divided the data into a training set and a testing set. We try to predict the last landmark of each trail in the testing set from the trails collected in the training set. Based on the information stored in the probabilistic suffix tree data structure we are able to predict the next landmark after a series of landmark interactions. In order to predict the next landmark using the visit frequencies, we present the user with a ranked list of landmarks according to their probability in the suffix tree, combined with the pattern matching method of [6]. Suppose that we have a trail, $T = t_1, t_2 ... t_{n-1}, t_n$ from our test set of trails, where t_i, $1 \le i \le n$, are landmarks, and we wish to predict the next landmark in the sequence, i.e. t_{n+1}. Then Algorithm called Predict, based on [6], is given below, where S is the suffix tree built from the training set of trails.

```
[Predict(T,S)]
1. **begin**
2. **let** s be the longest suffix of T in S;
3. **if** s is empty
        %i.e. no historical information is available on s;
4       **return** the top-10 most popular landmarks;
        %these will be children of the root of the tree.
5. **else** % s is non-empty;
```

```
6.      return a ranked list of the most popular
7.              landmarks directly reachable from s ;
8.end if
9.end
```

The first metric we use for prediction score is Hit and Miss (H&M) rule where the prediction result is the link with the highest probability. The scoring of this metric counts the number of correct predictions, i.e. it scores 1 for a correct prediction and 0 for a false prediction. The second metric is Mean Absolute Error (MAE) which returns a ranked list 1-r where r is the actual link that was followed and the MAE is calculated by r-1. Hit and Miss is a strict metric that predicts based on the highest probability when MAE returns the position of your prediction in a ranked list of possible predictions which have the top-n probabilities.

6 Experimental Testing

We are going to investigate two datasets collected from the Dartmouth campus wireless network [8] and data collected from the Reality Mining [3] project. We use one year worth of Dartmouth data where the user connection with the wireless network has been recorded and the Reality Mining project recorded users' mobile phone connection with cell towers for more than a year. In order to interpret the pervasive environment log files into trails we need to divide them into sessions. A session is defined as a finite sequence of landmark visits $u = P_1, P_2, ..., P_m$ with $P_i \in Landmarks$ and represents the experience of the user in the pervasive computing environment. How the data are divided into sessions depends on the type of data we are dealing with and the two main factors are location and time. For both datasets we have set a time limit of 5 hours of inactivity and a trail length limit 10. Table 1) Summarizes the characteristics of the collected data sets. From the collection of these sessions we create the probabilistic suffix tree discussed in section 3. From our experimental testing we want to present some pervasive system analysis information, investigate the presence of patterns between different parts of the dataset and we are going to present the constant appearance of patterns based both on the metric of time and trail frequency. Finally we are going to present the results from the prediction experiment we conducted.

Dataset	Interactions	Users	Landmarks
Dartmouth	1782931	4745	623
Reality Mining	2536034	89	32628

Table 1) Dataset Characteristics

6.1 Overall Analysis

In this subsection we present a number of different outputs and information we can extract by using the introduced trailed based analysis of pervasive computing trails. We want to get an overall idea of the landmark usage, how the use of the environment change over time, what are the most popular trails followed and how user behaviour might change over time. We provide this kind of query results to present the wide range of different queries our engine can execute and the different levels of abstraction and detail. We want to produce an overall view of the different kind of queries and information we extract from the probabilistic suffix tree.

We start our experimental testing for the Dartmouth dataset by conceptually categorising the different buildings by type (academic, administration, athletic, library, residence, and sociology). From the first level nodes we can identify the type of building and we can add up the landmark usage to provide information of the type of buildings with most user usage of the wireless network (Figure 3).

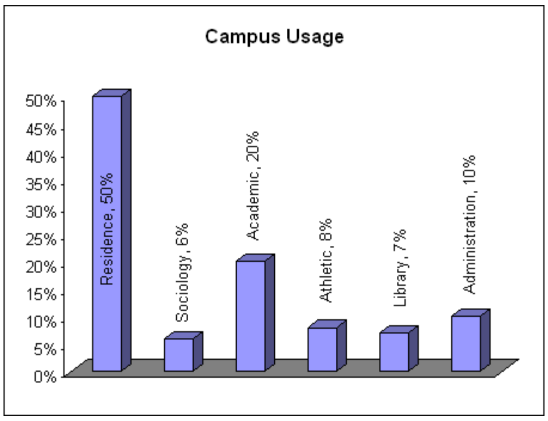


Figure 3) Landmark category usage in Dartmouth

The next experiment (Figure 4) investigates the number of overall users during different hours of a day in the Dartmouth dataset. Again this query is answered from first level nodes and by using the time information per user access the engine can calculate the number of visitors per landmark during a certain time period.

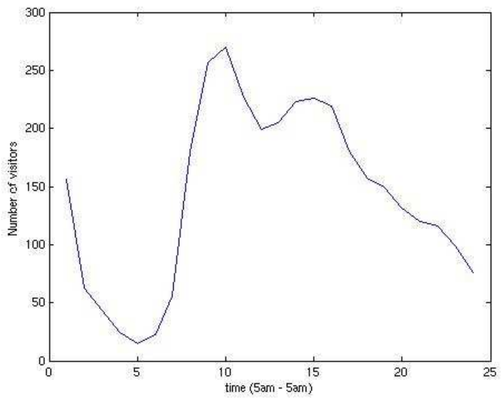


Figure 4) Number of overall visitors during each hour of the day in Dartmouth

From our query engine we present one of the ranking engine's output trail set, from the Dartmouth dataset, in Figure 5). The best trails based on their popularity are presented. We filtered the data based on their length and trails of length 10 where only considered. We represent trails by using different colours per trail and landmarks are placed based on their original location, to present user movement in the university campus. From the available output we can identify as most popular trails students moving between Residence, Academic and Library buildings

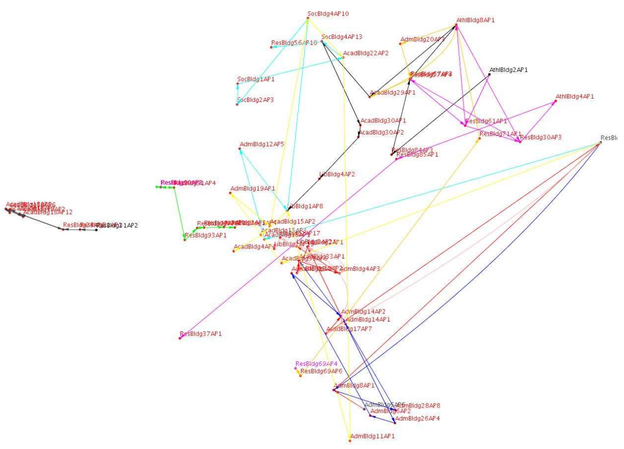
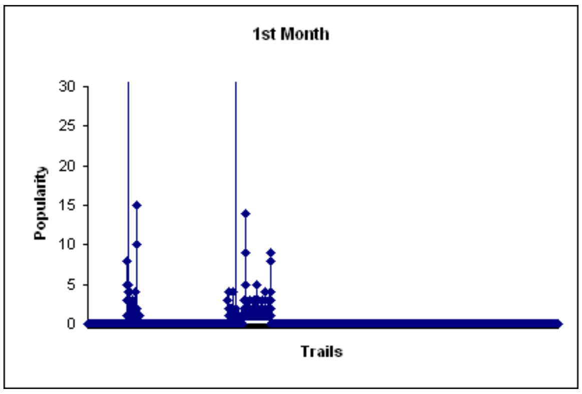
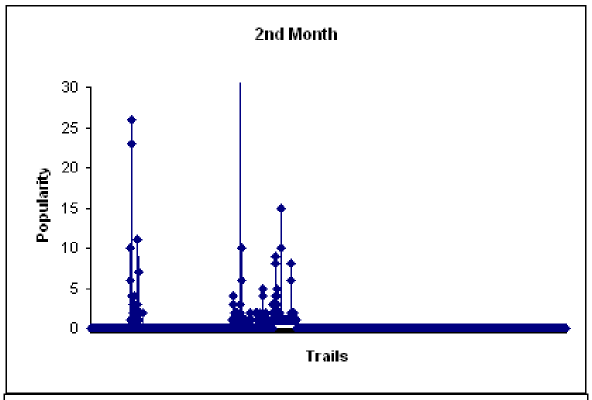


Figure 5) Query Engine result for one day data from the Dartmouth dataset. The 10 best trails of length 10 where extracted

In figure 6) we present an analysis of user behaviour change. We provide three consecutive months of trail distribution that a user followed. The tree for each month can be constructed by extracting from the overall tree nodes where the user id number containing in the nodes is equal to the id number of the user and the timestamps indicate that the interaction occurred during the specified time period.

From the produced output we can identify trails which appear throughout the three months and we can identify change in the users behaviour in the third months where new trails are introduced.





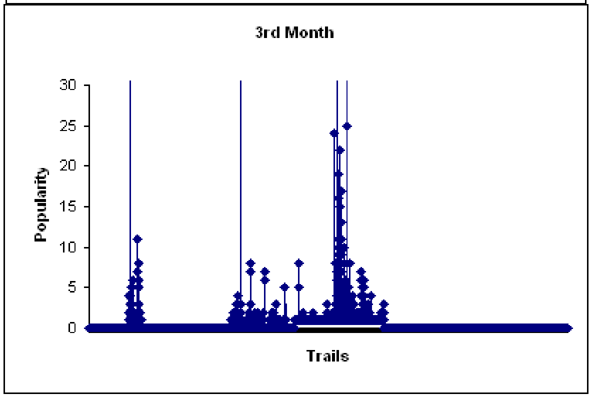


Figure 6) User behaviour change in monthly bases

6.2 Summarizability of the model

As mentioned before in order to compare the ranking and evaluate the summarization ability of our model we are going to use 2 metrics: the spearman footrule and the overlap metric. Overlap measures the percentage of trails returned from the ranking that exists in both ranks, when the spearman footrule takes into account the ranking of the trails. For this experiment we take into account all returned trails and we set as parameter to return trails only of a certain length and we retrieve the 10 best trails based on landmark popularity. We run the query twice to rank the results based on frequency of trails and time spent on the trails. We want to set such a non strict query definition to have an overall evaluation view of the datasets. Because both dataset are about one year worth of data, we divided the datasets into training and test trail sets. Test trail sets consist of the final month and the train set of the rest of the dataset. In figure 7) and 8) we present the overlap and spearman footrule results of our dataset based on different trail lengths.

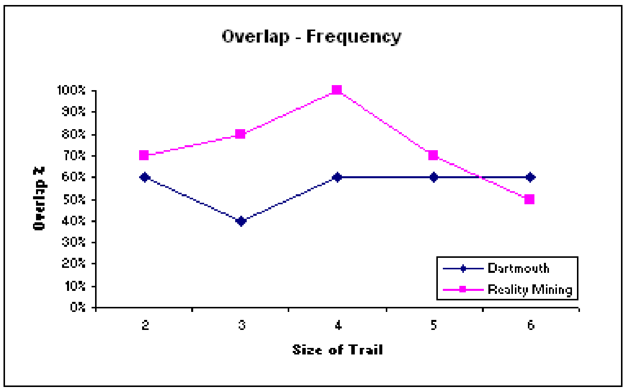


Figure 7) Overlap Trails based on frequency

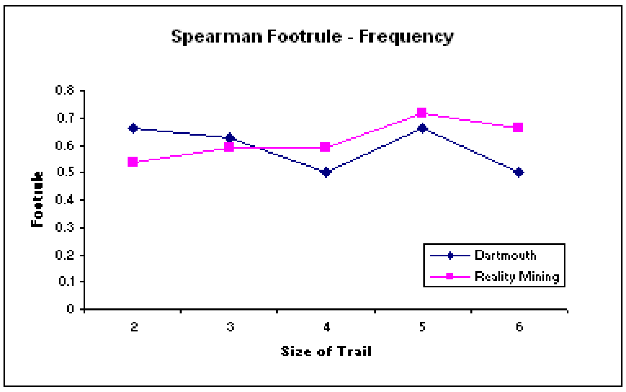


Figure 8) Spearman footrule based on frequency

In figure 7) we present the overlap for the frequency ranking query and in figure 4) the time ranking query. In figure 7), for the Dartmouth dataset we see a consistency of 6/10 trails to be present for all trail length except length two, the footrule metric is similar to the overlap and it shows some difference between the ranking of the discovered trails.

We also experimented with time based queries for the Dartmouth dataset (Figures 9 and 10) and even though we see at the overlap that the percentage of trail results present on both testing and training dataset are few, around 20% and 50%. From the spearman footrule we see that these trails are returned on the same ranking for training and testing datasets ability of our model by using the metrics discussed at the previous section.

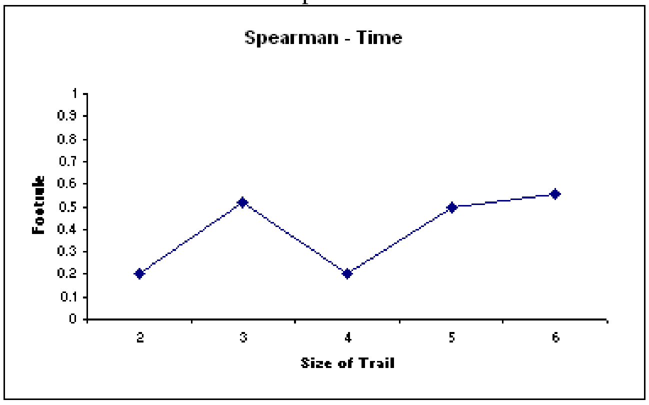


Figure 9) Spearman footrule based on Time query

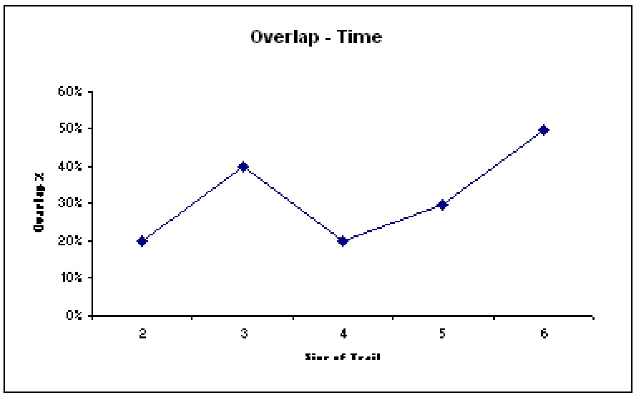


Figure 10) Overlap Trails based on Time query

6.3 Predictive Power of the model

To investigate the predictive power of our model we will use cross validation to compare the prediction accuracy based on the metrics discussed in section 5. Figures 11) and 12) present the results produced from hit and miss and Figure 13) the MAE testing of the data. All graphs present that prediction gets better when we take into account previous history of the user movement. For H&M we see that predicting the 4th and 5th step produces more than 70% prediction accuracy from the Dartmouth dataset. Even though the Reality Mining dataset produces poor results we discarded prediction improves by removing unexpected events. We define unexpected events as trails with probability lower than a certain threshold. In Figure 12) we have removed trails based on different trail probability. By removing trails with probability less than 10% the prediction accuracy rises from 50% to 70%.

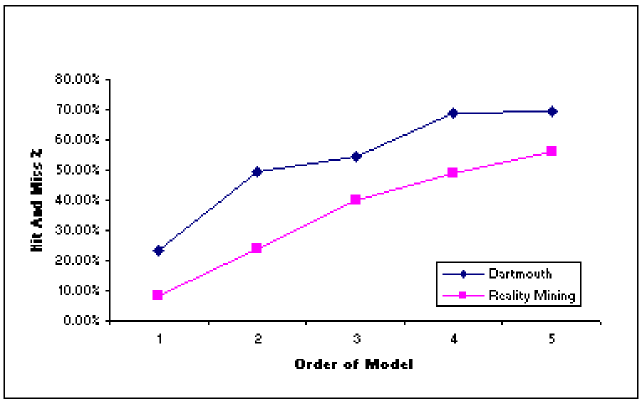


Figure 11) Hit and Miss for different model orders

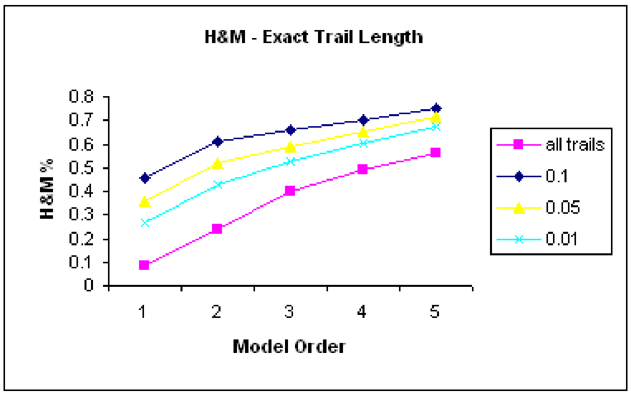


Figure 12) H&M for Reality Mining data after removing unexpected events

Even though for small trails H&M produces unreliable prediction results with MAE prediction is accurate for smaller trails and for both datasets. For Dartmouth we can predict trails in a list of top-3 for first order models and for longer trails the correct result always appear in the list of top-2 results. For Reality Mining first order models is inaccurate and the correct result is in the top-15 but again for longer trails the result is in the top-3 for order 2 and 3 and after that the correct prediction exists in the top-2 list.

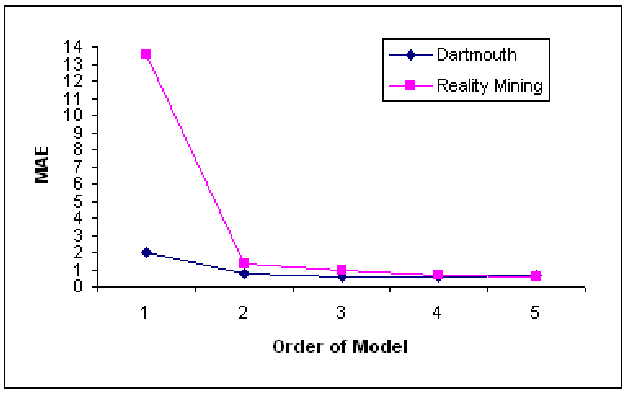


Figure 13) Mean Absolute Error for different model orders

Generally H&M is a strict metric and it requires some user interaction history before producing accurate results. On the other hand with MAE returns a list of possible predictions and we can see that even with no history at all we can get the correct result in a small list of possible predictions.

1 Conclusion and Future Work

In this paper we proposed a model for trail-based analysis for pervasive computing systems, which can identify trail patterns based on different metrics. We have presented experimental results based on two datasets and we have evaluated the summarizability and predictive power of our probabilistic suffix tree data model. Our experimental results present the flexibility of our model to produce different kind of outputs based on different levels of abstraction, presented high rate of prediction and the presence of patterns based on different metrics. For future work we are experimenting with different prediction measures and we are working on a classification algorithm for pattern matching. An application and an API are also under construction that will allow other researchers the use of our probabilistic model.

References

[1] J. Borges and M. Levene, *A fine grained heuristic to capture web navigation patterns*, SIGKDD Explorations, 2, pp. 40-50, 2000.

[2] I. Cadez, D.Heckerman, C. Meek, P. Smyth, and S. White. *Visualization of navigation patterns on a web site using model based clustering* In Proceedings of the Sixth International KDD conference pages 280-284, 2000

[3]. N. Eagle and A. Pentland, *Reality Mining: Sensing Complex Social Systems.* Personal and Ubiquitous Computing, Vol 10, 4, 2006.

[4] R. Fagin, R. Kumar, and D. Sivakumar. *Comparing top k lists* SIAM Journal of Discrete Mathematics, 17(1):134160, November 2003.

[5] R. Golledge, *Wayfinding Behavior: Cognitive Mapping and Other Spatial Processes*,The Johns Hopkins University Press, 1998.

[6]. P. Jacquet, W. Szpankowski, and I. Apostol. *A universal predictor based on pattern matching* IEEE Transactions on Information Theory, 48:1462-1472,2002.

[7] S. Jespersen, T.B. Pedersen, H. Thorhauge, *Evaluating the Markov Assumption for Web Usage Mining*, WIDM '03, November 7-8, 2003, New Orleans Louisiana, USA.

[8] D. Kotz and K. Essien, *Analysis of a campus wide wireless network, WirelessNetworks*, vol.11, pp. 115133, 2005.

[9] B. Lan, S. Bressan, B.C. Ooi and Y. Tay, *Making Web Servers Pushier* Proc.Workshop Web Usage Analysis and User Profiling (WEBKDD '99), Aug. 1999

[10] B. Lan, S. Bressan, B.C. Ooi and K. Tan, *Rule-Assisted Prefetching in Web-Server Caching*" Proc. ACM Int'l Conf. Information and Knowledge Management (ACM CIKM '00), pp. 215-228, Sept. 2000

[11] M. Mateas and P. Sengers, *Narrative Intelligence*, John Benjamins Publishing, 2003

[12] Tom Nicolai and Holger Kenn *Towards Detecting Social Situations with Bluetooth.* UbiComp 2006, Irvine, USA.

[13] O'Neill, E., Kostakos, V., Kindberg, T., Fatah gen. Schiek, A., Penn, A., Stanton Fraser, D. and Jones, T. *Instrumenting the city: developing methods for observing and understanding the digital cityscape.* UbiComp 2006, Irvine, USA.

[14]. D. Papadogkonas, G. Roussos and M. Levene. *Discovery and Ranking of Significant Trails*". 2nd Int. Workshop Expl. Context History in Smart Environments (ECHISE 2006). Irvine, CA, 17 September, 2006.

[15] J. Pitkow and P. Pirolli "*Mining longest repeating subsequences to predict world wide web surfing*" In Proc. of the Second Usenix Symposium on Internet Technologies and Systems, pages 139-150, Colorado USA, October 1999

[16] R.R. Sarukkai *Link Prediction and path analysis using markov chains*, Computer Networks and ISDN Systems, 30:457-467, 1998.

[17] Sevtsuk A., Ratti C., 2005, *iSPOT: describing the use of space on the MIT campus through the analysis of WiFi networks*, proceedings of CUPUM '05: The Ninth International Conference on Computers in Urban Planning and Urban Management, London, 29 June - 1 July 2005

[18] P. Weiner, *Linear Pattern Matching Algorithms*, Proc. 14th IEEE Annual Symp. on Switching and Automata Theory, pp1-11, 1973.