# Analysis of Location Prediction Performance of LZ Algorithms Using GSM Cell-based Location Data

Alicia Rodriguez-Carrion, Carlos Garcia-Rubio, Celeste Campo, Alberto Cortés-Martín, Estrella Garcia-Lozano, Patricia Noriega-Vivas
Department of Telematic Engineering
University Carlos III of Madrid
Leganés (Madrid), Spain
e-mail: {arcarrio, cgr, celeste, alcortes, emglozan, pnoriega}@it.uc3m.es

***Abstract*—Predictions about users' next locations allow bringing forward their future context, thus having additional time to react. To make such predictions, algorithms capable of learning mobility patterns and estimating the next location are needed. This work is focused on making the predictions on mobile terminals, thus resource consumption being an important constraint. Among the predictors with low resource consumption, the family of LZ algorithms has been chosen to study their performance, analyzing the results drawn from processing location records of 95 users. The main contribution is to divide the algorithms into two phases, thus being possible to use the best combination to obtain better prediction accuracy or lower resource consumption.**

***Keywords-location; prediction; LZ;***

## I. INTRODUCTION

Many applications are based on users' current context, but sometimes this is not enough. If the application reacts when the current context changes and the task to carry out due to the change takes some time to complete, then the result of this reaction may not come in time. Even more, reacting too late to context changes may make the user experience worse by showing information out of context or performing unexpected actions for the user. To solve these situations we could add information about the most probable next context, so that those applications would have more time to make the necessary adjustments and be ready when the user's context actually changes.

This work is focused on a particular aspect of users' context, their location, thus aiming to offer services based on users' future destinations. More precisely, we are going to study some tools for estimating those future locations: the so-called location prediction algorithms.

Location predictions may be an interesting improvement for ubiquitous computing applications, such as Location Based Services (LBSs). The prediction of user's next location would allow providing services related not only to the user's current location, but also to her future destinations. This way the user could be aware about information of a certain place (restaurant, museum) and decide whether to stop by that place or not right before getting there. The mobile phone itself may also be aware of the user's future location, thus being able interact with that location (e.g. an office or home) so it is prepared somehow when the user gets there (computer, lights or heat turned on).

We are also interested in learning and predicting using the mobile terminal itself because of several reasons, namely: (i) the advantages drawn from the fact that each user (terminal) learns and predicts her location, thus making the process distributed (with respect to the option of the network doing all the work); (ii) the improvement in privacy, since there is no need for sending location data through the network (the device obtains that information and process it); and (iii) the possibility of choosing the preferred technology for location tracking among the many ones integrated in mobile devices (GPS, WiFi or GSM/MTS…).

Taking into account the limited resources of mobile devices and the goal of increasing the number of correct predictions, our work is focused on the LZ family, a set of three compression algorithms: LZ [1], LeZi Update [2] and Active LeZi [3]. The interest in these algorithms is due to their ability to make real time predictions without consuming many resources, thus being good candidates to be executed on mobile devices. In addition to that, they are able to adapt to routine changes, which is an interesting feature taking into account the variability of user's behaviors.

One of the main contributions of our research is the new approach followed when analyzing these algorithms. Instead of considering them as a block process, we split each one into two independent phases: tree updating scheme and probability calculation method. This approach allows studying which instance of each phase is the best for reducing error rate and achieving the lowest resource consumption. We discuss the working principles of these predictors and how to make this separation in section II.

In section III we present the results obtained after evaluating the combination of different instances of each phase, regarding both error rates and resource consumption. The analysis is based on GSM location records, but there are similar analysis using Wi-Fi data [4]. In a previous study [5] we showed preliminary results obtained after processing 10 traces randomly chosen from a set of 95 users. The contributions of the current work over [5] are: (i) the analysis of the results obtained after processing of 95 users' traces using the prediction algorithms to validate the performance evaluation results shown in the previous work; (ii) the analysis of the results drawn from processing some mobility

traces we have recorded for comparing them with those of the anonymous users; and (iii) the explanation of certain unexpected results related to Active LeZi algorithm.

To finish the paper we summarize the main conclusions along with some future research lines in section IV.

## II. PREDICTION ALGORITHMS

In this section we study the working principles of the three LZ family algorithms: LZ, LeZi Update and Active LeZi. These algorithms are domain independent, meaning that they consider each location as a different symbol without taking into account any other information about that location (as opposite to domain dependent algorithms, which make their predictions based on location context information such as coordinates or place function). They process a symbol string, known as movement history or trace ($L$) that represents the locations visited by the user. The predictions made by these algorithms are based on two main hypotheses: (i) user's mobility patterns are repetitive, thus movement history being a stationary process; and (ii) user's movement follows a probabilistic model, and therefore $L$ is also a stochastic process.

There are two main reasons for considering these algorithms: (i) they do not need many resources, thus being possible to execute them on mobile devices; and (ii) they take into account changes in user's behavior, therefore if a user usually visits certain places and at some point starts visiting other locations, the algorithm will realize this change and make the predictions according to the new routine.

Along this section we describe the aforementioned algorithms, highlighting the possibility of splitting each one into two independent phases as described later in the section.

### A. LZ algorithm

This is the base algorithm and works as follows [1]. Let $\gamma$ be the empty string and $L$ the input movement history. LZ algorithm takes $L$ and splits it into substrings $s_0 s_1 \ldots s_m$ such that $s_0 = \gamma$ and for all $j \geqslant 1$ the prefix of substring $s_j$ (i.e. all but the last character of $s_j$) is equal to some previous $s_i$, for all $i<j$. The division is made sequentially, so that when each $s_i$ is parsed, then the algorithm considers only the remaining trace. In order to store these substrings (patterns) in an efficient way, LZ algorithm builds the so called LZ tree, each node of which representing a pattern and storing the number of times that substring appears among the parsed patterns.

After updating the tree, the next step is to calculate the probability for each known symbol to be the corresponding to the next location. In order to do that, LZ algorithm uses an approach proposed by Vitter [6]. Finally, LZ algorithm chooses the symbol with the highest probability of being the corresponding to the next location.

LZ algorithm has three main drawbacks: (i) patterns between two parsed substrings are lost; (ii) patterns contained within substrings parsed by LZ scheme are also lost; and (iii) Vitter method cannot make any prediction when a pattern is detected for the first time, since it has not enough information. The two next algorithms try to overcome these limitations.

### B. LeZi Update algorithm

Bhattacharya and Das [2] propose to make the same parsing of LZ algorithm, but instead of adding just the substrings resulting from this parsing, LeZi Update adds also all the suffixes of each substring to the LZU tree. Therefore patterns within substrings are also taken into account.

Regarding probability calculation method, LeZi Update algorithm uses PPM (Prediction by Partial Matching [7]) and applies what is known as exclusion technique [2]. This algorithm solves the problems posed by Vitter approach, and the probability estimations are based on more information.

### C. Active LeZi algorithm

The algorithm proposed by Gopalratnam [3] is intended to consider the substrings among consecutive parsed patterns when building the so called ALZ tree, thus solving the remaining problem of LZ algorithm. In order to achieve this, Active LeZi uses a window of variable length, which is determined by the longest pattern parsed by LZ algorithm at each step. Once the length of the window is updated (if needed) and the new symbol is added to it, all the suffixes of the window are added to the tree.

The probability calculation process is based on PPM algorithm as before, but in this case exclusion method is not applied. With Active LeZi algorithm all the initial problems are solved at the expense of increasing the information stored, and therefore memory and time resources required, as we will see in the next section.

### D. Our proposal

After describing each algorithm, we may realize that they share a common structure. Every algorithm takes each new symbol, processes it to update the corresponding tree and finally calculates some probabilities. Therefore, we can distinguish two stages: (i) **tree updating scheme,** which processes each new symbol and updates the corresponding tree, which is in charge of storing user's mobility patterns; and (ii) **probability calculation method,** which takes the data of the updated tree to estimate the probability of each known symbol to be the corresponding to the next location. Once all the probabilities are calculated, the prediction would be the symbol with the highest probability.

This division allows studying each step separately and determining its impact on the performance. Some results derived from processing several traces with all the possible combinations will be shown in the next section.

## III. PERFORMANCE EVALUATION

In this section we show some results obtained from processing mobility traces with the algorithms described in section II. The analysis will be focused on hit rate as well as the memory usage and processing time. But before starting with the performance analysis, a description of the mobility data used is provided.

### A. Data collection analysis

The set of algorithms explained in section II deals with symbols representing the locations visited by the user. In order to obtain those symbols, two steps are needed. The first

step is to gather location-related information, using any of the several technologies integrated in almost every mobile device nowadays and which retrieves this information. We have chosen location data based on GSM network information. Devices can record the base station (BS) to which they are connected every time. The BS a user is connected to changes as she moves so that the movement can be followed by tracking the BS series the user has been connected to. Although it is the option with worst location accuracy (with respect to Wi-Fi or GPS), GSM network provides global coverage even in indoor environments, and implies the lowest power consumption. During the second step the location information extracted from the GSM network is translated into symbols in the following way. The network splits the space into cells, each one identified by a Location Area Code (LAC) and a cell identifier. These two parameters can be translated into a unique symbol that represents the zone covered by the cell. Therefore each time the terminal changes from one cell to another, the device records the new location represented by its corresponding symbol.

In order to evaluate the predictors' performance we have analyzed two different datasets made up of GSM-based location data. The first one comprises the trace we have recorded, which stores the movements of a person who makes a regular routine (going from home to work, then going to a restaurant for having lunch, afterwards returning to work and finally going back home) during four days, generating a trace that gathers 2897 cell changes among 33 different cells. However, since we want to study the behavior of the algorithms in general scenarios, we have considered the Reality Mining Project dataset [8], which recorded the movement history of 95 different anonymous users during the 2004-2005 academic year.

### *B. Hit rate analysis*

We are going to use the 9 combinations that can be made with the three instances of each phase described in section II-D to process the traces described above. Figure 1 represents the percentage of traces (users) that attain, at least, the corresponding averaged hit rate (number of predicted next cell and actual next cell matches divided by the total number of cell changes). For example, in the case of Active LeZi (ALZ) algorithm combined with Vitter or PPM without exclusion, 50% of users achieve, at least, an averaged hit rate around 60%.

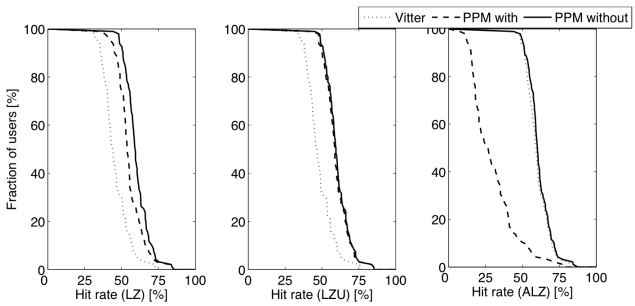


Figure 1. Comparison of hit rate attained when fixing the tree updating scheme and varying the probability calculation method

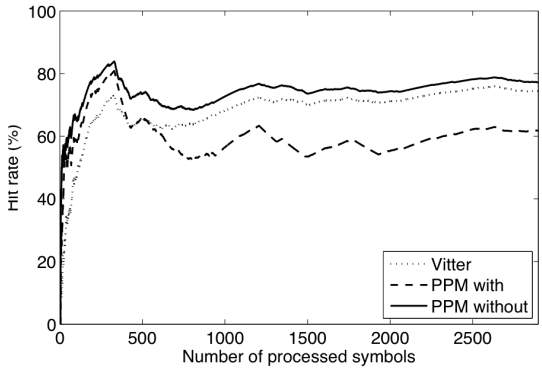


Figure 2. Hit rate evolution when processing the 4 days trace with Active LeZi updating scheme combined with each probability calculation method

For this first comparison we have fixed the updating scheme, represented in each of the three subfigures, and applied each probability calculation method. We can see that PPM without exclusion method is the best one in all cases. Vitter method results are very close to those of PPM without exclusion, even being a much simpler calculation method and thus consuming fewer resources, as we will see later. Hit rate achieved by Active LeZi combined with PPM with exclusion method is much lower than in the rest of the cases. This same behavior can be also observed in our 4 days trace. Figure 2 shows the evolution of hit rate along the 4 days, considering Active LeZi updating scheme combined with the three probability calculation methods. As it can be noticed, hit rate achieved by Vitter method is very close to that achieved by PPM without exclusion, whilst PPM with exclusion hit rate remains much lower.

This last fact may be surprising, but if we take a deeper look into the working principles of PPM with exclusion method, the reason becomes clear. This method is focused in assigning probabilities to complete patterns (set of consecutive symbols), as we can see in [2], instead of assigning probabilities to symbols (as done by PPM without exclusion [3]). Active LeZi adds always the content of the window (i.e. complete patterns) to the its tree whilst LeZi Update or LZ adds incremental patterns that starts with one-symbol patterns and increment their length as the trace is analyzed. Therefore, whereas LeZi Update or LZ tree has one-symbol patterns that can be evaluated by PPM with exclusion, Active LeZi does not. Since we are only interested, by now, in the next location, we only want the probabilities of the next one-symbol patterns. Consequently, the combination of LeZi Update or LZ and PPM with exclusion provides good results, but Active LeZi does not provide enough information to PPM with exclusion to make correct predictions.

With respect to the comparison of updating schemes (i.e. if we fix the probability calculating method and apply different updating schemes), we can see in figure 3 that Active LeZi (ALZ) is the best choice when working with Vitter and even with PPM without exclusion, although the differences in the last case are very small. LeZi Update works better with PPM with exclusion because of what we

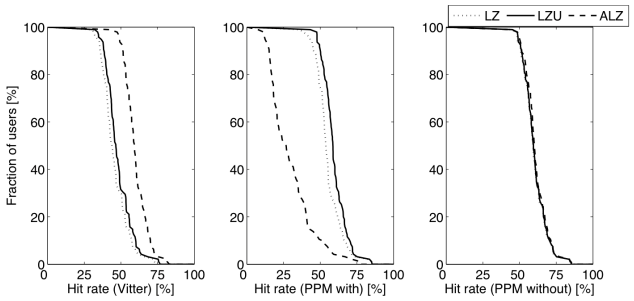


Figure 3. Comparison of hit rate attained when fixing the probability calculation method and varying the tree updating scheme

have previously discussed. Without taking into account the combination of Active LeZi with PPM with exclusion method, the results are consistent with those shown in [4]. This conclusion could be foreseen since information gathered by ALZ tree is greater with respect to LZU tree, and the same applies to LZU tree with respect to LZ tree.

### *C. Resource consumption*

Each of the two phases explained in section II-D has very different effects related to resource consumption. Tree updating scheme takes care of building the pattern tree, thus being tightly coupled with memory consumption, whilst the probability calculation is related to the processing time and depends on the complexity of the method used.

Figure 4 represents the node count evolution of each tree as each symbol of a single trace is processed. ALZ tree grows much faster, whereas LZ and LZU tree sizes also starts growing quickly but stops increasing so fast at a lower level, achieving a size two orders of magnitude lower than ALZ tree in the end. This shows that Active LeZi scheme achieves the best hit rate in most cases at the expense of much higher memory consumption, which may be unacceptable for some applications.

With respect to processing time, Vitter method needs much less time than PPM methods, which spend an accumulated time (after processing an entire trace) even two orders of magnitude larger than Vitter method. Therefore, if applications using these algorithms are time sensitive, Vitter method would be more convenient even having lower hit rate.

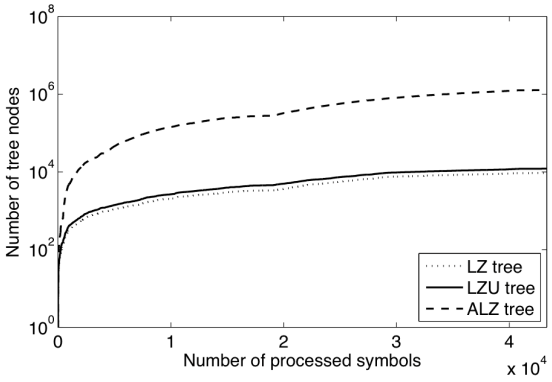


Figure 4. Node count of different trees (log scale)

## IV. CONCLUSIONS

Along this paper we have evaluated three LZ family algorithms (LZ, LeZi Update and Active LeZi), by separating them into two independent phases, and taking into account hit rate and resource consumption. The following conclusions can be drawn from this work: (i) Active LeZi updating scheme achieves the highest hit rate at the expense of being the highest memory consumer; (ii) the best probability calculation method depends on the updating scheme and the trace to be processed: PPM methods are usually the best ones, whereas Vitter method is much faster.

With respect to the analysis done in [5] we have considered an entire set of 95 anonymous users' as well as controlled traces we have recorded. Regarding Song's work [4], we have studied the algorithms as two independent phases, we have included Active LeZi algorithm, and used GSM-based traces, thus covering a countrywide area. It would be interesting to consider Markov models in future works to check if order-2 Markov model achieves better results than LZ algorithms as showed in [4] (where a campus wide network was considered) when processing country wide location data. It would also be interesting to study, among others, the following topics: (i) how to include time information in the predictions to know also when the user will move; (ii) how to filter cell changes when user does not move; and (iii) to study the energy consumption associated to the execution of the algorithms on the terminals.

## ACKNOWLEDGMENT

This work has been partially supported by the project España Virtual, led by DEIMOS Space and funded by CDTI as part of the Ingenio 2010 program.It has been also partially supported by the Spanish Ministry of Science and Innovation through the CONSEQUENCE project (TEC2010-20572-C02-01) and partially founded by UC3M and DGUI in the framework of the project CCG10-UC3M/TIC-4992.

## REFERENCES

[1] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," IEEE Trans. Inf. Theory IT-24, 5, pp. 530–536, Sep 1978.

[2] A. Bhattacharya and S. K. Das, "LeZi-update: an information-theoretic framework for personal mobility tracking in PCS networks," ACM/Kluwer Wireless Networks J. 8, 2- 3, pp. 121–135, Mar 2002.

[3] K. Gopalratnam and D. J. Cook, "Online sequential prediction via incremental parsing: The Active LeZi algorithm," IEEE Intell. Syst. 22, 1, pp. 52–58, Jan/Feb 2007.

[4] L. Song, D. Kotz, R. Jain, and X. He, "Evaluating Next-Cell Predictors with Extensive Wi-Fi Mobility Data," IEEE Transactions on Mobile Computing 5, 12, pp. 1633– 1649, Dec 2006.

[5] A. Rodriguez-Carrion, C. Garcia-Rubio, and C. Campo, "Performance evaluation of LZ-based location prediction algo- rithms in cellular networks," Comm. Letters. 14, pp. 707– 709, August 2010.

[6] J. S. Vitter and P. Krishnan, "Optimal prefetching via data compression," Journal of the ACM 43, 5, pp. 771– 793, Sep 1996.

[7] J. G. Cleary and W. J. Teahan, "Unbounded length contexts for PPM," in The Computer Journal, pp. 52–61, 1997.

[8] N. Eagle, A. Pentland, and D. Lazer, "Inferring Social Network Structure using Mobile Phone Data," Proceedings of the National Academy of Sciences (PNAS) 106, 36, pp. 15 274–15 278, 2009.