A Context Prediction Code and Data Base

Rene Mayrhofer, Harald Radi and Alois Ferscha

Institut für Pervasive Computing, Johannes Kepler University Linz
Altenberger Str. 69, A-4040 Linz, Austria
{mayrhofer,radi,ferscha}@soft.uni-linz.ac.at

Abstract. Many of the currently available sensors do not provide simple, numerical values but more complex data like a list of other devices in range. Although these sensors can, in the general case, not be transformed to numerical values, they nonetheless provide valuable information about the device or user context. For exploiting all available context information, it is thus important to also regard ordinal and nominal sensor values. In this paper, we propose to jointly develop a meta data format for the evaluation and assessment of context recognition and prediction methods.

1 Introduction

The goal is to derive current and future high level context information from low level sensor data by following a four-step approach: data acquisition, feature extraction, classification and prediction [1]. Starting from low level sensor data (e.g. Bluetooth, WLAN, RFID), appropriate features (e.g. SNR, MAC addresses in proximity, access points in range, ESSID) are extracted and classified to yield high level user context information (e.g. busy, traveling, in a meeting, in the office, at lunch, at home, telephoning, etc.).

However, these sensors do not yield single, numerical values but data with a more complex structure. Some important information that can be extracted from these sensors is categorical (nominal) and non-atomic, e.g. the list of MAC addresses which are currently in communication range. Nonetheless, it could provide useful information for determining the current user context via an automatic classification of all available sensor signals. If non-atomic values should be used as input to a classification algorithm which can only work with numerical inputs, they need to be coded. The standard procedure for dealing with categorical data is to code each possible sensor value as binary input to the classification algorithm. This has been applied successfully to categorical data with a bounded set of values (e.g. department), but is problematic when the set of possible values is not known in advance or is too large to be coded with binary inputs (e.g. WLAN MAC addresses would need 2^{48} input dimensions to cover all possible values). Therefore, coding categorical data as numerical inputs does not seem to be the best solution and a better method should be found. We have presented a method to classify heterogeneous feature vectors in [2]; it is based on two operations, which have to be defined for each dimension of the feature space,

i.e. for each feature. With these two methods, it is possible to use most common classification and clustering algorithms on the heterogeneous input space.

Although this method helps in coping with different sensor data types, a major problem with the current state of data bases for context recognition and prediction can be identified: many groups work on context recognition and prediction (e.g. [3–18]) and consequently use many different sensor technologies and many different sensor data types combined with many different recognition and prediction methods. Due to the lack of a common data format, the input sensor data sets and results are incompatible and incomparable. An approach to solve this major problem is to use an unified context data representation, which ensures compatibility, ease of data exchange and compliance with software tools. We propose to jointly define a meta data format, realized as Document Type Definition (DTD) or XML schema, to host context data sets and to host methods for context recognition and prediction on these data sets.

In this paper, we do a first step towards this goal by presenting a preliminary format to log time series of heterogeneous feature vectors efficiently w.r.t to storage space and to parse it easily for processing and evaluation. This log format is fully ASCII-based and could be a base for the common data format that still needs to be developed. We also offer a small C++ library for reading and writing log files in this format and a selection of methods for recognizing and predicting context which have already been adapted to this format.

2 Log Format

We would like to point out that the presented log format is preliminary; it would be desirable to develop a DTD or XML schema for a common format suitable for storing arbitrary sensor logs for context recognition and prediction. The main advantages of a XML-based format are that it can be processed with most software packages and that it is extensible without breaking compatibility to older versions. We intend to work towards such a goal and convert our current data sets to the common format as soon as it has been defined.

The main log data is currently stored as a semicolon separated list with one data vector per line; meta data about the features, including minimum and maximum values as well as code books, is stored in an XML file. It would be possible to store the meta data as a header in front of the log data and we actually recommend a common format to allow for such meta data headers. For putting log data in a public repository, it is simpler to have all necessary data stored in a single file. In Fig. 1, the first three lines of a sample log file are shown. During initialization, a header line starting with the keyword `[init]` indicates the used features and their order in the log file; this information should be embedded in the header/meta data section of a XML-based file. As can be seen, nearly all possibly types are used in this log file: numerical discrete (e.g. `Time.Timestamp`, `Audio.Peaks`), numerical continuous (e.g. `Audio.Mean`, `Wlan.ActiveSignalLevel`), nominal (e.g. `ActiveWindow.ActiveWindow`, `Wlan.ActiveEssid`, `GSM.CellID`) and a special case thereof, binary (`Power.Plugged`).

The only other keyword currently defined for this file format is `[missing]`, which indicates that the respective feature value could not be sampled at that time and which is obviously different from the empty value indicated by two adjacent semicolons `;;` (e.g. a list value like `Wlan.Peers` can be empty if no element of the list is present). When processing the log data, special care has to be taken for missing values – in our current implementation we assign them a weight of zero. Feature names in the header are derived from our sensors and the extracted features, e.g. the currently set ESSID of the WLAN sensor, but are transparent to the log file format. Another possible formatting of the data from Fig. 1, embedded in XML, is shown in Fig. 3. Due to the use of XML attributes for feature IDs, missing values do not have to be listed explicitly.

An example of the additional XML meta data file can be seen in Fig. 2. For each of the features that need persistent storage (i.e. data which is not contained in the log part but needs to be preserved across invocations), an element with the feature id as attribute and the storage values as sub elements is added. This format is general and allows each attribute to store and interpret arbitrary values, as long as they are coded as XML elements. Other features like the binary `Power.Plugged` feature – indicating if the laptop is plugged into its charger – do not need any persistent storage and are consequently not listed here. This XML tree could be used directly in the header section of a combined log file.

Lessons Learned

The definition and usage of this preliminary log format led to a few insights on context recognition data sets: the format should be

simple to use Experience shows that one of the most important properties is the ability to import the data sets into various software packages for data processing. Although our current semicolon separated format was adequate for first tests, XML is more appropriate for public data sets and benchmarks due to better extensibility and tool support. Powerful tools like XSLT processors allow simple transformations of the log data in the case that a specific tool can not directly import it.

self contained Using different files for storing the actual time series and the meta data about features also turned out to be disadvantageous. Even if the handling of a combined data format (with meta data in a header and time series in a log section) leads to more complex writer and parser code, this is compensated by the clearly simpler administrative handling of a single file for each data set instead of two files belonging together. However, in a few scenarios it was necessary to update the meta data file immediately after writing each log line because a clean application shutdown could not always be guaranteed due to low battery failures. To prevent a corruption of the whole data set with invalid meta data, as it had happened before implementing the synchronous update of the meta data file, it must be guaranteed that the meta data section is always up to date. With a combined XML file, we do currently not have a solution for that problem; the whole file would need

to be re-written for each new log entry because the meta data header might change. But this would lead to a significant performance degradation. With the current format, it is sufficient to append a line to the log file and update the (small) meta data file.

open During modification and extension of our logging framework over a period of a few months, the definition of an abstract persistent storage area allowed great flexibility in adding new types of features without needing to adapt the log format; we definitely recommend to let each feature (i.e. each dimension of the feature vector) read and write its own persistent storage in the form of arbitrary XML elements. The ability to add additional elements in future versions without breaking older data sets makes XML generally more future-proof.

3 Hosting Public Code and Data Sets

Since we expect that for most data sets for context recognition and prediction, additional information will be necessary, uploading the compressed data sets to a publicly accessible FTP directory might not be sufficient. Instead, we propose to use a Wiki [19] as a common repository, as has proven successful for the Portland Pattern Repository [20]. A description of the common context data format and methods for processing can be stored along with public data sets and can be easily modified by the community. We have set up a Wiki for storing all available data sets along with a description of the data formats and already entered our own data sets and this paper in an electronic version. It is publicly accessible and freely modifiable at
`http://pervasive.soft.uni-linz.ac.at/context-database/` .

4 Conclusions

Heterogeneous sensor data can provide valuable information for context recognition and prediction and should thus not be neglected. A number of sensors which describe certain aspects of the device or user context, e.g. WLAN or Bluetooth devices in range, are categorical and thus not covered by standard, numerically oriented log formats. We have presented a preliminary log format that allows to efficiently store highly heterogeneous sensor data in an ASCII format and can be easily embedded in a common, XML-based format that still needs to be defined. Meta data like the mapping of sets of categorical sensor values or minimum and maximum values of numerical sensor data is currently stored in an accompanying XML file that could act as a header for a common file. It would be desirable to jointly define a common DTD or XML schema for publishing currently available data sets for context recognition and prediction in an extensible, future-proof format.

References

1. Mayrhofer, R., Radi, H., Ferscha, A.: Recognizing and predicting context by learning from user behavior. In G. Kotsis, A. Ferscha, W.S., Ibrahim, K., eds.: The International Conference On Advances in Mobile Multimedia (MoMM2003). Volume 171., Austrian Computer Society (OCG) (2003) 25–35
2. Mayrhofer, R., Radi, H., Ferscha, A.: Feature extraction in wireless personal and local area networks. In Agha, K.A., Omidyar, C.G., eds.: The Proceedings of The Fifth IFIP-TC6 International Conference on Mobile and Wireless Communications Networks (MWCN 2003), World Scientific (2003) 195–198
3. Gellersen, H., Schmidt, A., Beigl, M.: Multi-sensor context-awareness in mobile devices and smart artefacts. Mobile Networks and Applications **7** (2002) 341–351
4. Schmidt, A.: Ubiquitous Computing – Computing in Context. PhD thesis, Lancaster University (2002)
5. Salber, D., Dey, A.K., Abowd, G.D.: he context toolkit: Aiding the development of context-enabled applications. In: Proceedings of the 1999 Conference on Human Factors in Computing Systems (CHI '99). (1999) 434–441
6. Cook, D.J., Youngblood, M., E. O. Heierman, I., Gopalratnam, K., Rao, S., Litvin, A., Khawaja, F.: Mavhome: An agent-based smart home. In: First IEEE International Conference on Pervasive Computing and Communications (PerCom'03), IEEE Computer Society Press (2003) 521–524
7. Clarkson, B., Mase, K., Pentland, A.: Recognizing user context via wearable sensors. In: ISWC. (2000) 69–76
8. Beigl, M., Krohn, A., Zimmer, T., Decker, C., Robinson, P.: Awarecon: Situation aware context communication. In Dey, A., Schmidt, A., J.McCarthy, eds.: Proceedings of the Fifth International Conference on Ubiquitous Computing (UbiComp'03). Volume 2864 of Lecture Notes in Computer Science., Seattle, WA, USA, Springer (2003) 132–139
9. Cakmakci, O., Coutaz, J., Laerhoven, K.V., Gellersen, H.W.: (Context awareness in systems with limited resources)
10. Gross, T., Specht, M.: Awareness in context-aware information systems. In: Proc. Mensch & Computer 01, Bonn 2001. (2001)
11. : Deliverable D05: 1st year progress report of the Oresteia project. Technical report (2002) available at http://manolito.image.ece.ntua.gr/oresteia/htmldocs/deliverables.htm.
12. Kidd, C.D., Orr, R., Abowd, G.D., Atkeson, C.G., Essa, I.A., MacIntyre, B., Mynatt, E.D., Starner, T., Newstetter, W.: The aware home: A living laboratory for ubiquitous computing research. In: Proceedings of the Cooperative Buildings, Integrating Information, Organization, and Architecture, Second International Workshop, CoBuild'99. Volume 1670 of Lecture Notes in Computer Science., Springer (1999) 191–198
13. Headon, R.: Movement awareness for a sentient environment. In: First IEEE International Conference on Pervasive Computing and Communications (PerCom'03), IEEE Computer Society Press (2003) 99–106
14. Judd, G., Steenkiste, P.: Providing contextual information to pervasive computing applications. In: First IEEE International Conference on Pervasive Computing and Communications (PerCom'03), IEEE Computer Society Press (2003) 133–142
15. Mäntyjärvi, J., Himberg, J., Huuskonen, P.: Collaborative context recognition for handheld devices. In: First IEEE International Conference on Pervasive Computing and Communications (PerCom'03), IEEE Computer Society Press (2003) 161–168

16. Henricksen, K., Indulska, J., Rakotonirainy, A.: Modeling context information in pervasive computing systems. In: Pervasive Computing, First International Conference, Pervasive 2002, Zürich, Switzerland, August 26-28, 2002, Proceedings. Volume 2414 of Lecture Notes in Computer Science., Springer (2002) 167–180
17. Bauer, M., Becker, C., Rothermel, K.: Location models from the perspective of context-aware applications mobile ad hoc networks. Personal and Ubiquitous Computing **6** (2002) 322–328 Sonderforschungsbereich SFB 627 (Nexus: Umgebungsmodelle für mobile kontextbezogene Systeme).
18. Bauer, M., Rothermel, K.: Towards the observation of spatial events in distributed location-aware systems. In Wagner, R., ed.: Proceedings of the 22nd International Conference on Distributed Computing Systems Workshops (ICDCS 2002), Los Alamitos, California, USA, Universität Stuttgart, IEEE Computer Society (2002) 581–582 Sonderforschungsbereich SFB 627 (Nexus: Umgebungsmodelle für mobile kontextbezogene Systeme).
19. Leuf, B., Cunningham, W.: The Wiki Way. Addison-Wesley (2001)
20. : Portland pattern repository. (http://c2.com/ppr/)

```
[init]Time.Timestamp;ActiveWindow.ActiveWindow;Audio.Mean;Audio.Peaks;Audio.Band.0;Audio.Band.1;
   Power.Plugged;Wlan.ActiveEssid;Wlan.ActiveMode;Wlan.ActiveSignalLevel;Wlan.ActiveMacAddress;
   Wlan.Peers;Wlan.NumPeers;GSM.CellID;
1068993793;0;35.12744140624988600;0.00000000000000000;115.05078125000006000;112.98437500000001000;
   1;0;2;0.62000000000000000;0;100;1.00000000000000000;[missing];
1068993824;1;0.08825171921780387;249.00000000000000000;-1.51129771706587120;-0.69374532185629911;
   1;0;2;0.63000000000000000;0;1000;1.00000000000000000;[missing];
```

Fig. 1. Header and two lines of an example log file with highly heterogeneous data vectors

```
<persistent>
   <feature id="ActiveWindow.ActiveWindow">
      <element id="0"><![CDATA[cmd.exe]]></element>
      <element id="1"><![CDATA[trillian.exe]]></element>
      <element id="10"><![CDATA[EXCEL.EXE]]></element>
      <element id="93"><![CDATA[uninst.exe]]></element>
   </feature>
   <feature id="Audio.Mean">
      <element id="maxval"><![CDATA[153.30539772726931000]]></element>
      <element id="minval"><![CDATA[0.00000000000000000]]></element>
   </feature>
   <feature id="Audio.Peaks">
      <element id="maxval"><![CDATA[856]]></element>
      <element id="minval"><![CDATA[0]]></element>
   </feature>
   <feature id="Wlan.ActiveEssid">
      <element id="0"><![CDATA[nme]]></element>
      <element id="1"><![CDATA[]]></element>
      <element id="10"><[CDATA[]]></element>
      <element id="2"><![CDATA[universe]]></element>
      <element id="3"><![CDATA[]]></element>
   </feature>
</persistent>
```

Fig. 2. Meta data for the example log file

```
<log>
   <sample timestamp="1068993793">
      <feature id="ActiveWindow.ActiveWindow">0</feature>
      <feature id="Audio.Mean">35.12744140624988600</feature>
      <feature id="Audio.Peaks">0.00000000000000000</feature>
      <feature id="Audio.Band.0">115.05078125000006000</feature>
      <feature id="Audio.Band.1">112.98437500000001000</feature>
      <feature id="Power.Plugged">1</feature>
      <feature id="Wlan.ActiveEssid">0</feature>
      <feature id="Wlan.ActiveMode">2</feature>
      <feature id="Wlan.ActiveSignalLevel">0.62000000000000000</feature>
      <feature id="Wlan.ActiveMacAddress">0</feature>
      <feature id="Wlan.Peers">100</feature>
      <feature id="Wlan.NumPeers">1.00000000000000000</feature>
   </sample>
   <sample timestamp="1068993824">
      <feature id="ActiveWindow.ActiveWindow">1</feature>
      <feature id="Audio.Mean">0.08825171921780387</feature>
      <feature id="Audio.Peaks">249.00000000000000000</feature>
      <feature id="Audio.Band.0">-1.51129771706587120</feature>
      <feature id="Audio.Band.1">-0.69374532185629911</feature>
      <feature id="Power.Plugged">1</feature>
      <feature id="Wlan.ActiveEssid">0</feature>
      <feature id="Wlan.ActiveMode">2</feature>
      <feature id="Wlan.ActiveSignalLevel">0.63000000000000000</feature>
      <feature id="Wlan.ActiveMacAddress">0</feature>
      <feature id="Wlan.Peers">1000</feature>
      <feature id="Wlan.NumPeers">1.00000000000000000</feature>
   </sample>
 </log>
```

Fig. 3. Embedding sensor data in XML elements