Probabilistic Event Extraction from RFID Data

Nodira Khoussainova, Magdalena Balazinska, Dan Suciu

Computer Science & Engineering Department, University of Washington
Seattle, Washington, USA
nodira, magda, suciu@cs.washington.edu

***Abstract*— We present PEEX, a system that enables applications to define and extract meaningful *probabilistic high-level events* from RFID data. PEEX effectively copes with errors in the data and the inherent ambiguity of event extraction.**

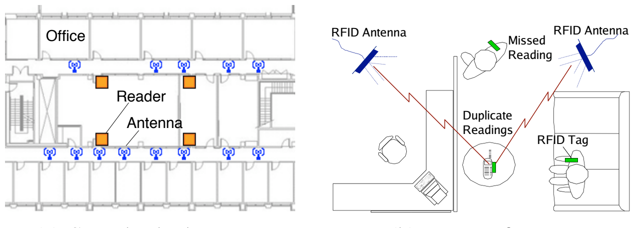
I. Introduction

The success of Radio Frequency Identification technology in industrial settings is leading many to consider pervasive deployments of the technology, where objects and people carry tags and RFID readers are scattered through the environment. RFID holds the promise of enabling many user-oriented applications from asset tracking [11] and alerting services [1], to sophisticated elder-care applications [9].

Exploiting RFID data presents significant challenges. RFID readers produce streams of low-level events of the form: "Tag 3 was seen at antenna 64 at 15:20". This low-level data must be transformed into high-level events meaningful to applications, such as "Alice entered the conference room at 15:20", or "Alice left her keys in her office". Two issues make this transformation challenging. The first issue is *reliability* [3], [5], [13]. RFID antennas frequently fail to read tags in their vicinity, causing complex events to go undetected. The second issue is *ambiguity*. Detecting a person at a sequence of locations may indicate that they are performing one of several activities: *e.g.*, Bob is printing a paper or sending a fax. Ambiguity can make it difficult to determine which from a set of high-level events actually occurred.

Previous work on RFID event detection ignores ambiguity and reliability issues [15]. These issues, however, make deterministic event detection unworkable, leading to event recalls near zero [8]. Schemes that do consider data errors deterministically clean the data [5], [6], [10], [12]. Such cleaning can improve data quality, but cannot always clean all errors. For example, if a laptop appears to be in two offices simultaneously, it is not always clear which location is correct.

Given the limitations of the deterministic cleaning and event detection techniques, we propose to use a probabilistic model to enable complex event extraction in face of uncertainty. Our contributions are (1) an event language (PeexL) for defining probabilistic events, (2) a technique for extracting events using *confidence tables* and *partial events* for handling ambiguity and reliability issues respectively, (3) an implementation of the approach in a system called *Probabilistic Event EXtractor (PEEX)*, a middleware layer on top of a relational database management system (RDBMS), and (4) an evaluation of PEEX through experiments with data collected from our building-wide RFID deployment. We have deployed 150 RFID antennas in all hallways of our building (Figure 1(a)). Antennas are polled continuously and tag sighting events are sent to a back-end database. Our deployment enables users to track personal belongings and friends' activities in the building.



(a) Sample deployment (b) Types of errors
Fig. 1. **RFID deployment and errors**

Since PEEX is based on an RDBMS and the output it produces follows standard probabilistic models, existing probabilistic DBMSs [2], [14] could be used to further manage and query extracted events. This paper only addresses the actual event extraction from RFID data.

II. Probabilistic Event Extraction

In this section, we describe PeexL and our techniques for handling ambiguity and data errors.

A. Events

Due to the uncertainty caused by data errors (as illustrated in Figure 1(b)) and ambiguity, events are probabilistic rather than deterministic in nature. PEEX uses the probabilistic data model described in [2], [4], [14] and represents probabilistic events as tuples stored in relations named for each event type.

The most important relation is `SIGHTING`, which has the following schema: `SIGHTING`(time, tagID, antID, prob). An example tuple in `SIGHTING` is (101, 10, 23, 1.0), which indicates that at time 101, the tag with id 10 was seen by antenna 23. All tuples in `SIGHTING` are deterministic (*i.e.*, have probability 1), because a tuple records the fact that the system is aware of this sensor reading.

An example of a higher-level event is a `MEETING` event with schema: `MEETING`(time, person1, person2, room, prob). An example tuple is (103, 'Alice', 'Bob', 435, 0.4), which represents that at time 103, PEEX believes that Alice and Bob are having a meeting in 435 with probability 0.4.

Uncertainty propagates as events are aggregated into higher-level events. *e.g.*, if there is limited confidence in underlying `ENTERED-ROOM` events, the confidence in a `MEETING` event will be accordingly lower.

```
FORALL SIGHTING S1, !SIGHTING S,  SIGHTING S2
CTABLE FLOOR5-STATS C
WHERE  SEQ(S1, S, S2)
       AND S1.antID = 'ant035'  AND S2.antID =  'ant036'
       AND S1.tagID = S2.tagID AND S.tagID = S1.tagID
       AND S1.tagID = C.tagID
CREATE EVENT  ENTERED-ROOM E
SET E.tagID = S1.tagID,
    E.room = C.room
```

Fig. 2. **Example of event specification in PeexL**

B. PeexL - Event Language

Users define *complex probabilistic events* from SIGHTING or other previously defined events using *PeexL*, the declarative query language for PEEX. PeexL queries take the form:

```
FORALL $I_1$, $I_2$, ..., $I_n$
[ CTABLE C ]
WHERE Condition
CREATE EVENT E
SET Assignments
```

The arguments of the FORALL clause, I_1, ..., I_n, correspond to primitive or other composite events, or to regular database tables and may be preceded by a negation !. The CTABLE clause specifies the confidence table and serves to handle event ambiguity (see Section II-C). The WHERE clause is as in SQL. Finally, E and the SET clause define the name and the attributes of the new event. Figure 2 illustrates a PeexL query that generates an ENTERED-ROOM event for any tag X (given by S1.tagID) and some room R if X is seen at antenna 35 followed by 36 with no sightings in between. The ! preceding S indicates that event S should NOT exist. R and the probability assigned to the event are computed using the confidence table FLOOR5-STATS as we discuss next.

C. Confidence Tables

A given combination of RFID tag-reads defines a composite event only with limited certainty. Such event confidences often depend on different attributes of the event: *e.g.*, the owner of office 501 is more likely than others to enter room 501 than the adjacent conference room 555. To specify such correlations, PEEX uses *confidence tables* that take the form: CTABLE(A_1, A_2, ..., A_n, conf) and are populated from labeled historical data. Figure 2 illustrates the approach for ENTERED-ROOM events. The confidence value is computed from a FLOOR5-STATS confidence table, which appears in the WHERE clause. The schema of FLOOR5-STATS is simply (tagID, room, conf). An example of a tuple in FLOOR5-STATS is (3, 501, 0.9), which indicates that if tag 3 is seen at antenna 35 followed by 36, then there is 0.9 probability that it enters room 501.

D. Partial Events

RFID errors dramatically impact event detection, because error rates are amplified at each level in the event hierarchy. To illustrate, imagine that a user defines the ENTERED-ROOM event with five underlying tag-read events. If each tag-read event has an error rate of 15%, the ENTERED-ROOM event has an error rate of 56%. If three ENTERED-ROOM events are now needed to detect a MEETING, the latter has an error rate of 91%! Deterministic event detection is thus unworkable in an error-prone environment.

It is often possible, however, to detect composite events even when some errors occur. For example, if only two out of five tag-read events are detected, the system may still conclude that an ENTERED-ROOM event occurred, although with lower confidence. PEEX captures this intuition through the use of *partial events*. Given a definition of a composite event consisting of n lower-level events (some may be negated), PEEX detects the composite event as soon as some non-empty subset of the n events occur (or do not occur). In our example, PEEX detects the ENTERED-ROOM event as soon as at least one tag-read event occurs. Of course, the more errors occur, the lower the confidence that the high-level event occurs.

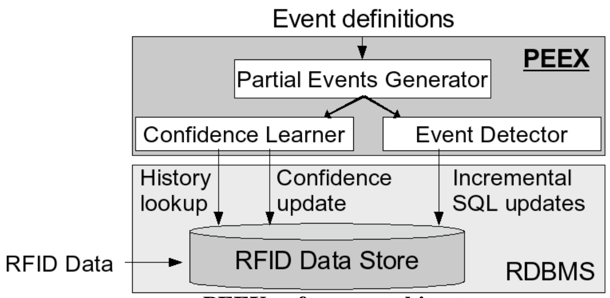


Fig. 3. **PEEX software architecture**

III. PEEX System

PEEX is a layer on top of a traditional RDBMS. This design enables us to demonstrate the benefits of probabilistic event extraction, while leveraging the features of an existing RDBMS. As illustrated in Figure 3, PEEX comprises an Event Detector, a Confidence Learner and a Partial Events Generator.

A. Event Detector

The Event Detector extracts events specified by the event definitions. All primitive and composite events are stored persistently in the RFID Data Store with one relation per event type. Primitive events are inserted into the store as they arrive.

To leverage the underlying DBMS, the Event Detector transforms PeexL event definitions into incremental SQL queries that it executes each time it runs. The interval between Event Detector executions is set by the administrator (*e.g.*, every 5 seconds). For each newly detected event, the Event Detector inserts a tuple into the appropriate relation and computes its probability using the corresponding confidence table. To avoid the proliferation of very low-probability events, PEEX drops events that have a confidence lower than some ϵ threshold defined by the administrator.

The probability assigned to a composite event is the product of the probabilities assigned to its underlying events and the corresponding value in the confidence table. For example, if a MEETING event is defined as two people entering a room (*i.e.*, two ENTERED-ROOM events), then the probability assigned to a particular MEETING is the probability that both ENTERED-ROOM events occur (which are obtained from the ENTERED-ROOM table), times the conditional probability that a MEETING occurs given that both ENTERED-ROOM events happened (which is obtained from the confidence table). In order to detect correlations between composite events, PEEX recursively inlines their definitions until all events in the query are independent; see [8] for more details.

B. Confidence Learner

To automatically populate confidence tables, PEEX uses annotated historical data that includes primitive events and labeled composite events. The confidence for an event is then determined by two sets: the set of historical composite events that match the event definition and the subset of those events that are also associated with the appropriate label. The ratio of the two sets, grouped by the appropriate attributes gives the confidence value for the event. We refer the reader to our technical report [8] for more details.

C. Partial Event Generator

To detect partial events, PEEX relies on the Partial Event Generator. Given an event definition that depends on lower level events $E_1, E_2, ..., E_n$, the Partial Event Generator generates a partial event for each subset of events with at least one positive event E_i and no consecutive negated events E_j, E_{j+1}. The Partial Event Generator adds each partial event to the system and from there on, these events are handled like regular events, each with its own confidence table.

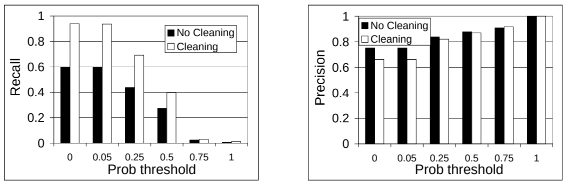
IV. Evaluation

To evaluate PEEX, we collected data for one hour with ten participants in our building-wide RFID deployment. Each participant was given several tags (*e.g.*, person, keys, laptop, mug), a schedule of meetings and lunch breaks, and was told to take coffee breaks and trips to the printer at their discretion. We collected 11585 SIGHTING events. We extracted several higher-level events including entered-room, coffee and meeting events. The Event Detector ran every 5 seconds. Each participant labeled their activities with timestamps during the collection period. We used half the data for populating the confidence tables and the other half for evaluating PEEX.

We measure recall/precision of the extracted ENTERED-ROOM events at different probability thresholds meaning that only events above the threshold are considered. Deterministic techniques are equivalent to setting the probability threshold to 1 (extracting only certain events) or 0 (extracting any event that has any chance of occurring). Clearly, PEEX can provide a higher recall than the first deterministic approach, from less than 1% to 60% (with threshold 0.05) and can deliver a higher precision than the second approach to applications that require it. PEEX allows applications to choose their desired trade-off between recall and precision (Figure 4).

In addition to comparing PEEX to deterministic techniques, we demonstrate the need for and the effect of probabilistic cleaning [7]. Figure 4 also shows the result of applying a single cleaning constraint on ENTERED-ROOM events after they have been detected by PEEX. With cleaning, PEEX can detect ENTERED-ROOM events at 93% recall (with 66% precision even though the probability threshold is only 0.05).

Finally, in order to determine whether PEEX is able to run in near real-time, we measure the average time it takes to detect events within each five second window. Our results showed that PEEX can detect events in 110 ms per time window given ten event definitions and thus is easily able to run in near real-time for a deployment with tens of users.



(a) Recall (b) Precision

Fig. 4. **PEEX event detection performance**

V. Conclusion

In this paper, we presented a probabilistic approach to high-level event extraction from RFID data. We also presented the design, implementation, and evaluation of PEEX, a data management system that effectively extracts probabilistic events from RFID data using three key techniques: it translates event definitions into SQL queries, it relies on confidence tables to determine the probability of ambiguous events, and it uses partial events to handle data errors.

Our long-term goal is to build an RFID data management system, where a plethora of applications can extract, manage, and possibly share high-level events. We view the work in this paper as an important step towards this goal.

References

[1] G. Borriello, W. Brunette, M. Hall, C. Hartung, and C. Tangney. Reminding about tagged objects using passive RFIDs. In *Proc. of the 6th Ubicomp Conf.*, Sept. 2004.

[2] N. Dalvi, C. Re, and D. Suciu. Query evaluation on probabilistic databases. *IEEE Data Engineering Bulletin*, 29(1):25–31, 2006.

[3] C. Floerkemeier and M. Lampe. Issues with RFID usage in ubiquitous computing applications. In *Proc. of the 2nd Pervasive Conf.*, Apr. 2004.

[4] T. J. Green and V. Tannen. Models for incomplete and probabilistic information. *IEEE Data Engineering Bulletin*, 29(1):17–24, March 2006.

[5] S. Jeffery, G. Alonso, M. J. Franklin, W. Hong, , and J. Widom. Declarative support for sensor data cleaning. In *Proc. of the 4th Pervasive Conf.*, Mar. 2006.

[6] S. R. Jeffery, M. Garofalakis, and M. J. Franklin. Adaptive cleaning for RFID data streams. In *Proc. of the 32nd VLDB Conf.*, Sept. 2006.

[7] N. Khoussainova, M. Balazinska, and D. Suciu. Towards correcting input data errors probabilistically using integrity constraints. In *Proc. of the Fifth MobiDE Workshop*, June 2006.

[8] N. Khoussainova, M. Balazinska, and D. Suciu. Peex: Extracting probabilistic events from RFID data. Technical Report 2007-11-02, Department of Computer Science and Engineering, University of Washington, 2007.

[9] M. Philipose, K. P. Fishkin, M. Perkowitz, D. J. Patterson, D. Fox, H. Kautz, and D. Hahnel. Inferring activities from interactions with objects. *IEEE Pervasive Computing*, 3(4), 2004.

[10] J. Rao, S. Doraiswamy, H. Thakkar, and L. S. Colby. A deferred cleansing method for RFID data analytics. In *Proc. of the 32nd VLDB Conf.*, Sept. 2006.

[11] V. Stanford. Pervasive computing goes the last hundred feet with RFID systems. *IEEE Pervasive Computing*, 2(2), Apr. 2003.

[12] F. Wang and P. Liu. Temporal management of RFID data. In *Proc. of the 31st VLDB Conf.*, Sept. 2005.

[13] E. Welbourne, M. Balazinska, G. Borriello, and W. Brunette. Challenges for pervasive RFID-based infrastructures. In *Proc. of PERTEC Workshop*, 2007.

[14] J. Widom. Trio: A system for integrated management of data, accuracy, and lineage. In *Proc. of the Second CIDR Conf.*, pages 262–276, Jan. 2005.

[15] E. Wu, Y. Diao, and S. Rizvi. High-performance complex event processing over streams. In *Proc. of the 2006 SIGMOD Conf.*, June 2006.