Approximation Trade-Offs in Markovian Stream Processing: An Empirical Study

Julie Letchner[#1], Christopher Ré[+2], Magdalena Balazinska[#3], Matthai Philipose[*4]

[#]*Computer Science & Engineering Department, University of Washington, Seattle, Washington, USA*
{[1]letchner, [3]magda}@cs.washington.edu

[+]*Department of Computer Sciences, University of Wisconsin, Madison, Wisconsin, USA*
[2]chrisre@cs.wisc.edu

[*]*Intel Research Seattle, Seattle, Washington, USA*
[4]matthai.philipose@intel.com

***Abstract*— A large amount of the world's data is both *sequential* and *imprecise*. Such data is commonly modeled as *Markovian streams*; examples include words/sentences inferred from raw audio signals, or discrete location sequences inferred from RFID or GPS data. The rich semantics and large volumes of these streams make them difficult to query efficiently. In this paper, we study the effects—on both efficiency and accuracy—of two common stream approximations. Through experiments on a real-world RFID data set, we identify conditions under which these approximations can improve performance by several orders of magnitude, with only minimal effects on query results. We also identify cases when the full rich semantics are necessary.**

I. Introduction

People and computers worldwide generate exabytes of audio, video, text, GPS, RFID, and other types of multimedia and sensor data [2]. Because disk storage is cheap, most of this data is archived for future use in diverse applications from location tracking in retail and hospitals [9], to activity monitoring [7], multimedia search/retrieval [15], and more.

While some applications can use raw sensor or multimedia streams directly [6], most rely on higher-level streams *inferred* from the low-level data. Search engines, for example, can index audio files by content only after these files have been translated into text. Similarly, location tracking or activity monitoring applications require that raw sensor streams be transformed into location or activity sequences before they are queried. Due to noise in the data or ambiguity in the inference process (or both), these inferred, high-level streams are *imprecise* (e.g., a spoken word might be either "eight" or "ate"; while an RFID reading might only narrow a person's location down to one of several adjacent rooms).

The current state of the art for supporting imprecise sequences is the model-based view [1]. A model-based view allows applications to query data as if it were deterministic; internally, however, the DBMS maintains a model of the imprecise sequence and uses this model to compute query results annotated with confidence scores. Because the sequence model is decoupled from the queried view, DBMS designers may choose any of several standard imprecise sequence models. These models range from simple deterministic models (called "MAP" models in the AI literature [11]), to models that reflect sophisticated correlations both within and between sequence elements [3], [8].

In general, model choice has a significant impact on DBMS quality and performance: increased model complexity yields higher fidelity to the underlying data and thus higher accuracy, but incurs additional computational and I/O costs. These high costs naturally raise the question of whether highly sophisticated imprecise sequence models are worthwhile. Would applications notice a difference in result quality if rich, imprecise streams were approximated using simple, deterministic ones? What performance benefits could be gained from such an approximation, which would allow high-performance, deterministic stream processing engine to be leveraged? How might a system achieve a flexible trade-off between the accuracy and efficiency of imprecise sequence processing?

In this paper, we study the performance and accuracy trade-offs of three common stream models: *MAP* (a deterministic model), *independence* (an uncorrelated model), and *Markovian* (a temporally-correlated model). We perform our study in the context of two common types of sequence queries—event queries and aggregated variants—and report results obtained using real-world location sequences inferred from an office-building RFID deployment.

Using the Markovian model to provide an accuracy baseline, we find that the independence approximation does not always yield higher accuracy than the simpler MAP approximation, despite its increased expressiveness. The accuracy of the two approximate models varies significantly based on query characteristics (described in Section II-B). The performance of all three models is as expected; independence and MAP approximations accelerate query processing by one and two orders of magnitude, respectively, with respect to baseline performance on a Markovian model.

II. Data & Query Model

A. Data Model: Markovian Streams

Our study focuses on an imprecise, sequential data model called a *Markovian stream*; the Markovian model or simplified versions have been adopted in nearly all imprecise sequence management systems [3], [4], [8], [12], [13].

The real-world Markovian streams in our study are derived from an office-building RFID deployment in which RFID readers are mounted in hallways. Figure 1(a) shows a small

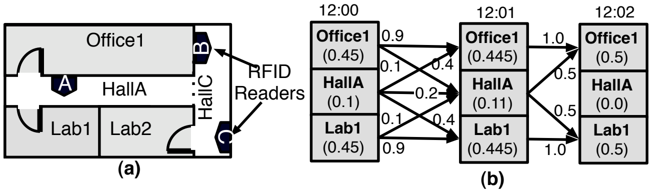

Fig. 1. (a) A schematic view of an RFID deployment. (b) A Markovian stream representing a distribution over paths through the environment in (a). The number on an arrow $d \to d'$ indicates the conditional probability $p(d' \mid d)$; the number inside each box represents the box's marginal probability.

portion of this office environment, including RFID readers A, B, and C. When a mobile RFID tag (attached to a person or an object) moves within the detection range of a reader, the reader records the time and presence of the tag using tuples of the form `(Tag_id, Reader_id, Time)`.

A Markovian stream over a tag's location can be inferred from these low-level tuples using any of many standard *probabilistic inference* algorithms. Such a Markovian stream represents a probability distribution over the paths the tag may have taken through the building. This stream is *imprecise*, both because of sensor noise (readers often fail to detect nearby tags) and because of *inherent ambiguity* (e.g. even on noise-free data it is impossible to distinguish between Office1 and Lab1). The stream in Figure 1(b), for example, expresses that Bob was in either Office1 or Lab1 with equal probability at time 12:02. The Markovian stream is also *temporally correlated*: the distribution over the tag's location at time $t+1$ depends on its uncertain location at time t. For example, Figure 1(b) expresses that, *given* that Bob was in Office1 at 12:01, he stayed in Office1 at time 12:02 with probability 1.0. The full-length version of this paper contains a formal definition and relational schema for Markovian streams [5].

Markovian streams are the output of *probabilistic inference* algorithms. These inferred streams, once materialized on disk, are the data queried in this study.

B. Query Model: Event Queries and Aggregation

Our study evaluates accuracy/performance trade-offs on both *event* (sequence) queries, and aggregated event queries:

1) Event Queries: At the heart of Markovian stream processing are *event queries*, which search for patterns in a Markovian stream (e.g. *"Find all times when Bob entered the coffee room last week."*). An event query is defined as an ordered sequence of *query links*, which are simply NFA states with 1) a single incoming edge and 2) an optional self-loop edge. Examples are shown in Figures 2(a1), (a2), and (a3). An event query result comprises a $\langle seqID, p\rangle$ tuple for each timestep in the stream, indicating the probability p with which the pattern was satisfied at stream element $seqID$. A set of such tuples is plotted in Figure 2(b1).

Event queries are processed on Markovian streams using NFA-based algorithm developed in prior art [8]. This algorithm maintains a joint distribution over ⟨ current NFA state(s), last input ⟩. Updating this distribution for a single Markovian stream timestep, to process a query with k links, requires $O(2^k D^2)$ time (here, D is the size of the Markovian stream domain, which in our study is 966 locations).

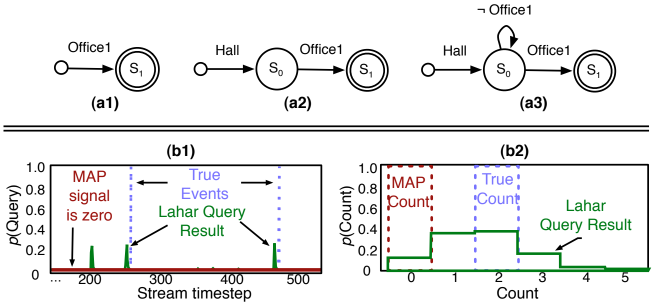

Fig. 2. (a1-a3) Markovian stream event queries. (b1) Event query output (unaggregated) on a real-world Markovian stream. (b2) Output of a count-based aggregation of the event query shown in (b1).

2) Aggregated Event Queries: We introduce here two types of *temporal aggregations* over Markovian streams. The first type is a Boolean aggregation which determines whether a particular event query was satisfied at *any* timestep (e.g. *"Did Bob enter the coffee room at any time last Monday?"*). The second type is a count-based aggregation, which determines the *number* of timesteps that satisfied the event query (e.g. *"How many times did Bob...?"*).

Temporal aggregation results cannot be computed simply by post-processing event query results, because event query results are correlated with each other. In the Markovian stream in Figure 1(b), for example, the pattern (HallA, Office1) is satisfied at timesteps 12:01 and 12:02, but these matches are mutually exclusive because they depend on Bob being in two different locations at time 12:01.

Computation of temporal aggregations requires maintaining the joint distribution over ⟨ current NFA state(s), last input, aggregate-value ⟩. Updating this distribution with a single Markovian stream timestep requires $O(2^k D^2 A)$ time, where A is the size of the aggregated value's *domain*. This computation is described further in the full-length version of this paper [5]. For Boolean aggregation, $A \in \{0, 1\}$; however, for count-based aggregation, A grows linearly in the length of the Markovian input stream (in the worst case). Importantly, we show in Section III that the performance consequences in practice are usually minor.

C. Markovian Stream Approximations

Here we briefly introduce the MAP and independence Markovian stream approximations. The full-length version of this paper [5] includes more details, including theoretical upper bounds on the errors incurred by these approximations.

1) MAP approximation represents a Markovian stream with its single most likely deterministic sequence (called the **M**aximum **a** **P**osteriori sequence). MAP views achieve quadratic space and computational savings by reducing the size of each timestep from $O(D^2)$ to $O(1)$.

2) Independence approximation discards the temporal correlations in a Markovian stream. The probability of a path using independence approximation is the product of independent values drawn from the stream's marginal distributions, instead of from its conditional distributions (temporal correlations). Independence approximation thus reduces the size of a Markovian stream timestep from $O(D^2)$ to $O(D)$.

III. Empirical Study

In this section, we study the performance and accuracy trade-offs of three imprecise stream models: Markovian, independent, and deterministic (MAP). We use real streams inferred using a particle filter from readings collected in a real deployment comprising 160 RFID readers installed in the hallways (and only the hallways) of our 6-story office building [14]. From over 6.6 million tag sighting events, we have curated two data sets which we label *unambiguous* and *ambiguous*. Each set comprises five distinct RFID traces reflecting a series of brief 1-minute office visits, for a total of 2.2 hours of Markovian streams (sampled at 1Hz), and 172 Mb of data. The five unambiguous Markovian streams contain uncertainty, but identify a single most likely location at each timestep. In contrast, the ambiguous streams often identify 2-3 rooms with roughly equal probability (recall the *inherent ambiguity* problem from Section II-A). In the ambiguous streams, temporal correlations capture the fact that objects cannot move arbitrarily (e.g. they can't move through walls, teleport, *etc.*).

A. Performance Study

In this section we identify the performance bottlenecks of Markovian stream processing and evaluate the effects on performance of independence and MAP approximations.

1) Performance Bottlenecks: We identified query processing bottlenecks by measuring baseline (no approximation) CPU and I/O performance on a set of representative queries. These queries search for the entered-room events present in our data. For simplicity, we restrict our study to *fixed-length* queries [4], on which performance and accuracy trends are most clear. The set of test queries includes four basic event specifications (NFAs including 1, 2, 4, and 8 links), each processed using all three temporal aggregation semantics. These results appear in Figure 3(a) (note the logscale y-axis), which reports average performance over five separate evaluations of each query.

The most interesting feature of Figure 3(a) is its demonstration that **CPU costs dominate performance**, sometimes by several orders of magnitude (the ratio between CPU and disk I/O time is shown above each bar in the plot). This feature sets Markovian stream processing apart from standard data warehousing where disk I/O costs, rather than CPU costs, would traditionally dominate in an untuned prototype. Figure 3(a) also demonstrates that **CPU costs increase exponentially with query length**; this scaling is noted in prior art [8] and is demonstrated most clearly on the unaggregated and Boolean-aggregated queries. Finally, Figure 3(a) shows that **aggregation latency is minimal**. Recall from Section II-B that the performance of count-based aggregation degrades as the size of the count domain increases. This aggregation cost is included in the CPU cost of the count-aggregated query performance in Figure 3(a). This figure shows that short event queries are satisfied relatively frequently, producing larger count domains (511 for the 1-link count-aggregated query in Figure 3(a)). Longer event queries are satisfied infrequently, so their count domains remain small (27 for the 8-link counting query in Figure 3(a)). Thus only queries satisfied frequently incur additional, non-trivial CPU costs.

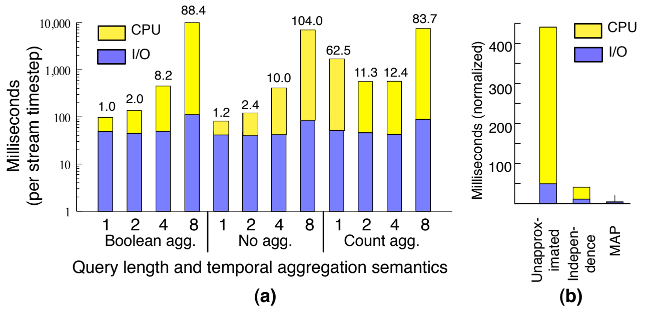

Fig. 3. (a) Overview of average performance on 5 trials of each query. Note the logscale y-axis. Numbers above each query bar give the ratio of $\frac{CPU}{I/O}$ latency. (b) Average performance of 4-link queries using MAP and independence approximations.

2) Performance on Approximate Streams: We study the effects on performance of MAP and independence approximations using the twelve representative queries in Figure 3(a). The *relative* effects of each approximation technique are consistent across all query lengths and aggregations, so for clarity we present results for a single, unaggregated 4-link query. Figure 3(b) shows these performance results, averaged over five trials of each query/approximation. Note that the y-axis of Figure 3(b) is *not* in logscale.

As expected, Figure 3(b) shows that **the dimensionality of stream timesteps dominates efficiency**. Independence and MAP approximations yield performance gains of one and two orders of magnitude, respectively, due to the fact that they reduce the processing state matrix to a vector and a constant, respectively. However, we also find that **the query processing engine must be designed to exploit these approximations**, as we show in the full-length version of this paper [5].

B. Accuracy Study

We study the effects of approximation on accuracy using a subset of the test queries already introduced; for simplicity we discuss only the accuracy of 4-link queries (results were similar on queries of other lengths). We define error as the Earth Mover's Distance [10] between result distributions (for Boolean queries, this reduces to the absolute difference between result probabilities) as computed using approximate vs. Markovian streams. Figures 4(a-f) plot this error as computed on all ten test streams (five from each data set).

1) MAP Error: **MAP error is highly bimodal** on non-aggregated (i.e. event) queries in our RFID/location domain, where event query matches tend to have probabilities well below 0.3. Most MAP errors stem from false negatives (the MAP stream "misses" events contained in the Markovian stream), and thus **the magnitude of most MAP errors equals the (low) probability of "missed" events**. However, in the rare cases when the MAP estimate includes a false positive event match, this false match incurs an error that approaches the maximum error of 1.0. **MAP error is on average higher on Boolean-aggregated queries because the true probability of Boolean aggregates tends to be higher** than the true probability of individual (non-aggregated) event queries, increasing the error incurred by false negatives.

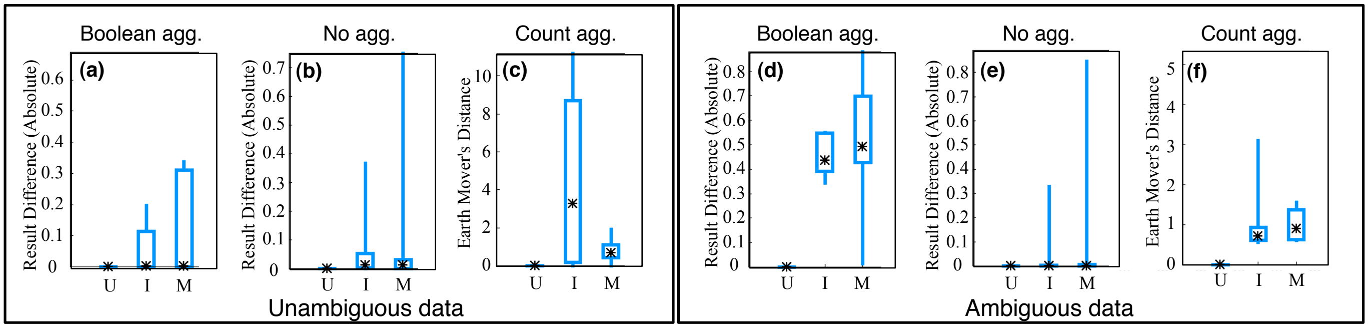

Fig. 4. Differences in query results computed on full vs. approximate Markovian streams, reported as the Earth Mover's Distance between distributions over the result value. Differences are shown for each approximation on a set of five 4-link queries with varying temporal aggregation semantics, on two trace sets.

Finally, on count-aggregated queries, MAP approximation is surprisingly accurate. Although the MAP count is inaccurate in proportion to the number of false positives/negatives in the MAP stream, **the MAP count is generally within the right order of magnitude** (we will see shortly that this is not true of independent approximations). In general, the quality of MAP approximations depends on the fidelity of the MAP estimate to the original Markovian stream; this in turn depends on the amount of uncertainty present in a data set.

2) Independence Error: Independence error is low on unaggregated queries in our domain. Here, assuming **independence tends to yield slightly *under*estimated event query probabilities**, because the independence assumption reduces the probabilities of high-probability paths containing query matches. Note, however, that Figures 4(b) and (e) include *only non-zero errors*, since the vast majority of timesteps incur zero error and hence obscure the behavior of the remaining errors.

In contrast, **assuming independence on temporally-aggregated queries (whether Boolean or count-based) tends to yield results that *over*estimate the true values**. This overestimation is often significant, because the mutual-exclusivity constraints encoded in temporal correlations are ignored. The effect of overcounting can be characterized by the degree of *temporal uncertainty* present in a data set; see the full version of this paper for more details [5].

3) Query Optimization: Query optimization techniques to choose the best (or no) approximation for a given Markovian stream and query are beyond the scope of this study; however, as a general guideline, our results show that MAP is preferred for count-aggregated queries, while the less aggressive independence approximation is preferable for Boolean queries.

This study demonstrates that the full complexity of the Markovian model is not always justified—many queries can be processed with high accuracy on simple, approximate models. At the same time, however, this study also demonstrates that there do exist queries that can be answered at a reasonable level of accuracy only by processing a fully Markovian model. Thus the accuracy/performance trade-off space is rich; a formal cost model for optimizing this space is an important area for future work.

IV. Conclusion

In this paper, we studied the performance/accuracy trade-offs of several standard approximations of Markovian streams, in an RFID-based location tracking domain. We found that the trade-off space is rich, affording many opportunities for query acceleration with minimal impact on query error.

Acknowledgements

This work was partially supported by NSF grant IIS-0713123, NSF CRI grants CNS-0454425 and CNS-0454394; by gifts from Intel Research, and by M. Balazinska's Microsoft Faculty Fellowship. J. Letchner is supported in part by an NSF graduate research fellowship.

References

[1] A. Deshpande and S. Madden. MauveDB: supporting model-based user views in database systems. In *Proc. of the SIGMOD Conf.*, pages 73–84, 2006.

[2] IDC. The expanding digital universe: A forecast of worldwide information growth through 2010. An IDC White Paper sponsored by EMC., March 2007.

[3] B. Kanagal and A. Deshpande. Efficient query evaluation over temporally correlated probabilistic streams. In *Proc. of the 25th ICDE Conf.*, pages 1315–1318, 2009.

[4] J. Letchner, C. Ré, M. Balazinska, and M. Philipose. Access methods for markovian streams. In *Proc. of the 25th ICDE Conf.*, 2009.

[5] J. Letchner, C. Ré, M. Balazinska, and M. Philipose. Approximation trade-offs in a markovian streamm warehouse: An empirical study. Technical Report #UW-CSE-09-07-03, University of Washington, July 2009.

[6] E. Lo et al. Olap on sequence data. In J. T.-L. Wang, editor, *Proc. of the SIGMOD Conf.*, pages 649–660. ACM, 2008.

[7] D. J. Patterson, D. Fox, H. A. Kautz, and M. Philipose. Fine-grained activity recognition by aggregating abstract object usage. In *Proc. of the ISWC Conf.*, pages 44–51, 2005.

[8] C. Ré, J. Letchner, M. Balazinska, and D. Suciu. Event queries on correlated probabilistic streams. In *Proc. of the SIGMOD Conf.*, pages 715–728, 2008.

[9] RFID Journal. Hospital gets ultra-wideband RFID. `http://www.rfidjournal.com/article/view/1088/1/1`, Aug. 2004.

[10] Y. Rubner, C. Tomasi, and L. J. Guibas. A metric for distributions with applications to image databases. In *Proc. of the ICCV Conf.*, pages 59–66, 1998.

[11] S. J. Russell and P. Norvig. *Artificial intelligence : a modern approach.* Prentice Hall, 2nd edition, 2003.

[12] Z. Shen, H. Kawashima, and H. Kitagawa. Probabilistic event stream processing with lineage. In *Proc. of Data Engineering Workshop*, 2008.

[13] T. Tran, C. Sutton, R. Cocci, Y. Nie, Y. Diao, and P. Shenoy. Probabilistic inference over rfid streams in mobile environments. In *Proc. of the 25th ICDE Conf.*, 2009.

[14] University of Washington. RFID Ecosystem. `http://rfid.cs.washington.edu/`.

[15] R. Yan, A. Hauptmann, and C. Rong Jin. *Image and Video Retrieval*, chapter Multimedia Search with Pseudo-Relevance Feedback, pages 649–654. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2003.