Improving Query Processing on Imprecise Data Streams

Eugenia Gabrielova [1]
Northwestern University
Evanston, IL

Julie Letchner [2]
University of Washington
Seattle, WA

Magdalena Balazinska [3]
University of Washington
Seattle, WA

[1]genia@u.northwestern.edu
{[2]letchner, [3]magda}@cs.washington.edu

ABSTRACT

Many applications, such as monitored health care and theft detection, depend on higher-level data inferred from low-level location sensors such as RFID and GPS. This high level data occurs in the form of imprecise, correlated sequences, which are modeled by Markovian streams. Such data is too difficult to manage with traditional databases.

Lahar is a system that warehouses and processes queries on such streams, returning a set of query answers annotated with probabilities. Some queries return many partial results, which wastes computing resources. Processing streams and queries in a reversed direction may result in fewer partial matches.

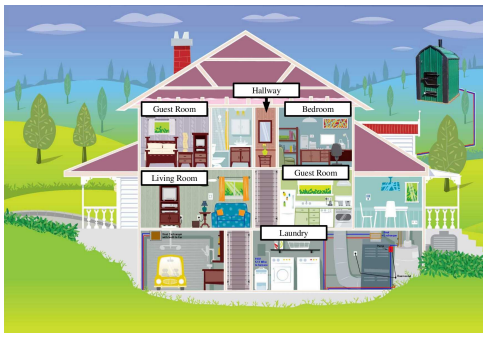
In this paper, I present an application developed to reverse Markovian streams and queries. This application also compares the efficiency of processing a query on forward and backward streams. Some properties of queries, such as a rare element at the end of a query, may make backward processing a more efficient choice. The ability to reverse and process Markovian streams backward to process such queries improves the efficiency of the Lahar system.

1. MOTIVATION

1.1 Motivating Scenario

Location sensing technology such as RFID and GPS produces a large amount of potentially very useful data. This data could very useful for applications such as theft detection, monitored healthcare, and smart homes. However, it is largely unmanageable with traditional databases because of its imprecise nature.

Consider, for example, a house designed to conserve energy and benefit the environment by tracking the behvaior of its inhabitants and managing lighting, temperature, water, and power accordingly. Each doorway will have an RFID beacon, and each member of the household would carry a card indicating to track their location.



Suppose that friends Abbie and Jake live in such a house. On a cold winter day, the house might select which rooms to heat based on where its inhabitants spend most of their time. Its energy management algorithm would benefit from knowing that Abbie and Jake are enthusiastic students and love to spend time reading and studying in their rooms and the living room. It would also be useful to know how many people access the guest room on a regular basis - if that room or another room is rarely used, the house can allocate resources accordingly.

This task is difficult because RFID sensors aren't always fully accurate, and because the activities of members of the household are often correlated over the course of day. In order to accomplish such complex energy management, the house needs a way to accurately interpret the low-level data from the RFID sensors in a useful manner.

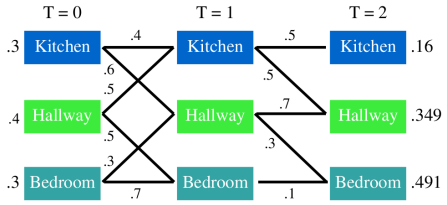
1.2 Background

Markovian Streams

The output from location sensors such as RFID or GPS appears in the form of ordered, imprecise, and correlated streams of data. Markovian streams are a compact way to store such data, and can be obtained from probabalistic models. The data from these streams is very useful in the scenario of the environmental smart home. Consider an example of Abbie and Jake preparing for final exams - they spend time studying in their rooms and in the living room, and enjoy meals and snacks in the kitchen.

These streams contain a set of *ordered* timesteps. Each timestep has a corresponding probability distribution of likely values. In the case of the smart home, these values would correspond to the likelihood of Abbie or Jake being in their rooms before lunch, or in the kitchen for a meal or snack.

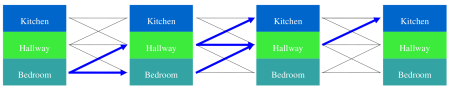
Another property of Markovian streams is that their timesteps are *correlated*. The probability of being in a certain location depends on earlier probabilities corresponding to that location. A person is more likely to be in the kitchen at 1:01 PM if they are already there at 1 PM.



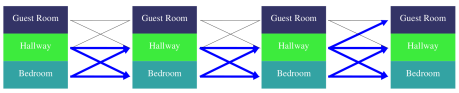
Processing Markovian Streams with Lahar

Lahar is a data warehousing system that was created to store and analyze Markovian streams by processing real-time event queries and returning a set of likely outcomes. For the example of the ecological smart home, it might be useful to query the RFID data for information about the behaviour of members of the household.

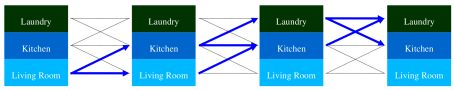
How likely is it that Abbie studies in her bedroom for a few hours and then goes downstairs to the kitchen for a snack?



How frequently is the guest room visited?



When does Jake take a break from watching movies in the living room to eat a meal and check his laundry?



Lahar has the ability to process these and other challenging queries, but the magnitude of the data it must analyze is immense. The quantity of deterministic data in one Markovian stream can be exponential. While it has been demonstrated that reading a stream is fairly quick, processing queries is much more time-consuming.

In some cases, indexes are available and can save time by allowing access to the relevant parts of a Markovian stream. However, it is not realistic to assume that useful indexing will always exist. Any optimization that narrows the field of possible results for a query, or enables Lahar to reach an answer more quickly, improves the speed of the system.

Though some systems exist that query large amounts of uncertain data, Lahar is especially useful for the data output from location-sensing technology. Warehousing this data offers potential for technological advancement in a number of applications, including smart homes, theft detection, and monitored healthcare.

1.3 Intended Contribution

The queries that Lahar analyzes can be described in terms of predicates, such as ”before-10-am” and ”this-is-the-kitchen”. If a predicate is rarely satisfiable in the data (suppose the guest room is rarely occupied), then it has high selectivity. The selectivity of a predicate is low if it is satisfied often in the data.

In Lahar, an event query that starts with a set of low-selectivity predicates may result in a number of partial query matches. For example, if Abbie spends a lot of time studying and Dmitrii spends a lot of time watching television in the living room, queries that include those events will return many possible results. If the end of the query contains any number of high-selectivity predicates, many of those partial matches will become dead ends. By satisfying more highly selective predicates first, Lahar will require fewer calculations to reach an answer. This means that Lahar will generate fewer partial query results.

This improved efficiency can be achieved by processing the stream in reverse, starting from the end of the stream and the end of the query, and going backwards in time.

2. TECHNICAL CONTRIBUTION

Stream Reversal Tool

The goal of my research was to develop a standalone application for reversing Markovian streams in Lahar. This would require compatibility with streams of various types, as well as a straight-forward experimentation method for testing relative efficiency of backward and forward processing.

2.1 Requirements

The purpose of the Stream Reversal Tool is to streamline the Lahar user experience in terms of efficient query processing. These are the initially defined goals

- Compatible with Lahar data formatting
- Takes as input a forward Markovian stream and produces the reversed stream
- Does not require user input beyond initiation with a forward stream

2.2 Reversal Algorithm

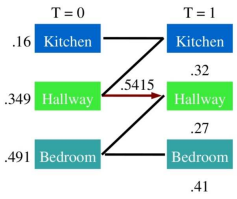
Generating a backward stream does not require any query information - only the original forward stream and its marginal probabilities. For a stream with 3 timesteps, T0-Forward is

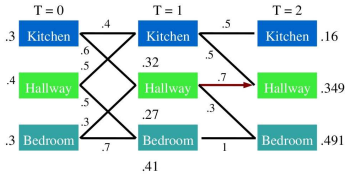
T2-Backward, and T2-Forward is T0-Backward. probabilities. Next, a brief illustration of the reversal process, and the algorithm used for implementation.

Example segment: b → b

$$P_{new} = \frac{(p_{after})(p_{FromForward})}{p_{before}}$$

$$P_{new} = (0.27 * 0.7) / 0.349$$
$$= .5415$$



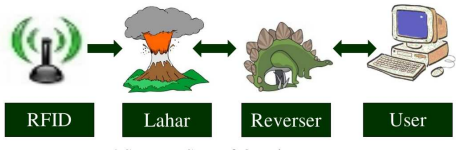


Reversing a Forward Stream

Input: Forward stream F
Let *vector* m = final marginal probability for each node [a, b, c] from F

for each backward time step
for each node [a, b, c]
Let x_m = p[node] from m
Let $x_{forward}$ = marginal probability from previous timestep in forward stream
Let p = probability on this segment from forward stream
next = p * $x_{forward}$ / x_m

2.3 Stream Reverser Overview



The most up-to-date version of the Stream Reverser can be run from within Lahar, or it can be initiated with hardcoded Markovian stream data. A tool to read in forward streams from text files behaves inconsistently. Input processing with Lahar's BDB utility has been unsuccessful, so the Stream Reverser is most useful when called from harcoded streams or from within Lahar.

Query processing functionality in the Stream Reverser is extended from Lahar's current method. Two primary adaptations occurred to make reversal and efficiency measurement possible. First, the stream generation method now takes stream structures as an input. Second, the reverser calculates efficiency with a modified implementation of the Lahar state calculator.

2.4 Recommended Next Steps

There are three primary improvements that could improve the utility of the Stream Reverser Tool.

1. **Lahar IO Interface:** The effort to implement Lahar IO compabitility into the Stream Reversal Tool was not successful. This would be a useful feature as it would enable the swift writing of streams to disk. To accompany this feature, it may also be helpful to implement an interface for creating Markovian streams, which would then write them to disk via BDB or some other database system in preparation for processing.

2. **Recording Start Time Steps for Satisfied Queries:** Lahar is currently capable of storing the ending timesteps for sequences that satisfy queries. Storing start time would enhance the quality of information available for stream reversal.

3. **Query Direction Optimizer:** Properties including frequency and connectedness of elements in a stream make reverse processing more fitting than forward processing. Harnessing these properties into a utility that decides for the user whether or not backward or forward processing is more efficient would eliminate the need to process queries both forward and backward.

3. EVALUATION

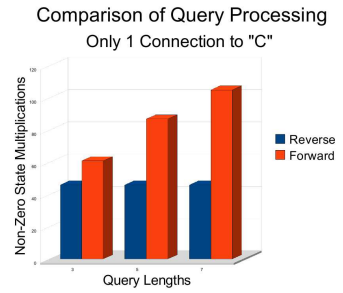
3.1 Experiment Set Up

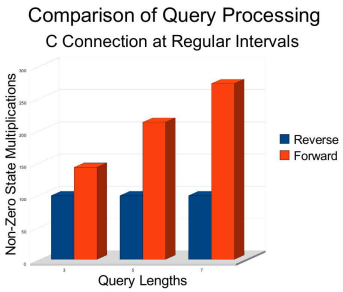
The Reversal Experiment Layer is a package that wraps around the stream reverser application to faciliate experiments on query processing efficiency. It also implements I/O for stream reversal. At this time, input from hardcoded stream data is functional. In the future, this layer of the application will be able to read streams from disk after they are processed by Lahar. However, at this time is dependent on Lahar's direct output.

One of the primary challenges of writing this utility was instantiation of Reg Operator elements from the Lahar system code. A workaround is implemented in the code comments but a fix would be a useful improvement for longterm use.

3.2 Results

Backward processing proved most effective on streams with one or more rare element. The control case - a well-connected stream, did not yield improved efficiency with backward processing. In many cases, backward processing improved efficiency by a factor of 1.3 to 2. In the following graphics, which represent the most notable performance, backward processing required a consistent quantity of operations while forward processing lost efficiency with increased query complexity and time.





All test cases have been processed on streams with three possible stream coordinates (A, B, C, for example). It may be useful to consider data with more members in future applications. Similar results, on a scale of 1.2 to 1.5 efficiency improvement, occurred with much longer streams (with length 800 - 122 timesteps). Control cases with zero or few disconnected stream elements did not display any significant difference between forward and backward processing.

4. CONCLUSIONS

The ultimate goal of this project was to improve Lahar's speed, and to optimize the manner in which it handles diverse event queries. The Stream Reversal Tool will improve the efficiency of Lahar in cases where reverse processing is beneficial. Additionally, the application improves the Lahar user experience by performing calculations necessary for reversing streams, which are often tedious beyond a very short stream length.

While reverse stream processing is not applicable in all cases, experimentation with the stream reversal tool demonstrates that when reversal is applicable, increase in efficiency is significant. In a number of cases, the quantity of calculations required to find the probability of satisfying a query decreases by a factor of 1.5 to 2 when processed in reverse.

The Stream Reversal Tool is in a stable state, and successfuly achieves its objective of reversing Markovian streams. It is practical for real-life NFA queries, and has potential for audio stream processing.

5. ACKNOWLEDGMENTS

I would like to thank Julie Letchner and Dr. Magdalena Balazinska for their guidance and mentorship in the Database group at the University of Washington this summer. Thanks to the DREU program providing the opportunity for this research experience.

6. REFERENCES

[1] J. Letchner, C. Ré, M. Balazinska, and M. Philipose. LaharOLAP Demonstration: Warehousing Markovian Streams. In *35th International Conference on Very Large Databases*, 2009.

[2] J. Letchner, C. Ré, M. Balazinska, and M. Philipose. Access methods for markovian streams. In *25th International Conference on Data Engineering*, 2009.

[3] J. Letchner, C. Ré, M. Balazinska, and M. Philipose. LaharOLAP: Supporting OLAP queries on Markovian streams. Technical Report #CSE-09-03-03, University of Washington, March 2009.