Distributed Stream Reasoning

Rehab Albeladi, Kirk Martinez, Nicholas Gibbins

Electronics and Computer Science, University of Southampton, UK
{raab1g09, km, nmg }@ecs.soton.ac.uk

Abstract. Stream Reasoning is the combination of reasoning techniques with data streams. In this paper, we present our approach to enable rule-based reasoning on semantic data streams in a distributed manner.

Data streams are being continually generated in diverse application domains such as traffic monitoring, smart buildings, and so on. Continuous processing of such data has been intensively investigated in the database community, where a special class of management systems [1][2] has been introduced to perform on-the-fly processing of data streams. However, these data streams lack standard formats, which make interoperability a real challenge. On the other hand, Semantic Web data has well-defined meanings and a number of semantic formats have been standardised. Semantic reasoners have been developed that can perform complex reasoning tasks on this data. Nevertheless, reasoning upon streaming data has received far less attention than reasoning upon static data. Stream Reasoning is the area that aims to combine reasoning techniques with data streams [3].

Research in this area mainly focuses on extending SPARQL to process RDF streams. We focus more on the infrastructure of the reasoning process. Our approach enables reasoning in a continuous manner using low level operators; this approach differs from that in [4], in which each query is split into a static and a dynamic part and the dynamic part is passed to a DSMS system. In the implementation of our stream reasoner, we have focussed on the two main issues of reasoning and distribution.

Stream Reasoning. To enable the rule-based reasoning process, we use the RETE algorithm [5] for pattern matching. Rules are translated into RETE networks of nodes. The nodes represent different operators and the data flows between these nodes. The tree-like network divides the matching process into multiple steps that perform different checks, so if a data element does not match the first node, it is simply discarded and does not complete its way through the network. A typical RETE network has two types of node: filter (or alpha) nodes, and join (or beta) nodes. A filter node is similar to the select operation in query languages; it only propagates statements that match its condition. On the other hand, a join node is responsible for joining some data elements of its two input streams depending on a specified condition. Each join node manages a time-based or tuple-based sliding window on each stream input.

A prototype RDFS reasoner for RDF data streams has been fully implemented, combining features from both reasoning techniques and stream processing techniques;

it performs the inference task as a rule engine using a Rete network, while the implemented Rete network performs some DSMS operations, such as converting streams into relations by using the sliding window technique. The system is fed by RDF streams, it matches them against the RDFS entailment rules, and produce new sets of data in a continuous manner. In our initial evaluation, we have been able to demonstrate the tradeoff between completeness and execution time by varying window sizes.

Distribution. For efficient processing of large volume data, scalability is a major concern. Distributed processing of data streams enables more scalable and fault-tolerant systems, so we distribute our reasoning networks using the eXtensible Messaging and Presence Protocol (XMPP). We have chosen XMPP for its push based distribution style which satisfies the real-time requirement of streaming applications with minimal latency, and for its ease of integration with Web technologies.

In order to minimise network traffic, we use a graph-based definition for the RDF stream data type. Instead of triples, we define the RDF stream as an ordered sequence of RDF graphs associated with a time element. This can achieve other advantages besides being more efficient in terms of transportation; some queries can be evaluated by viewing the graph as a single record, and data provenance can be tracked more easily at a graph level.

We have built a prototype system that can process RDF data streams using distributed RETE networks, in which nodes are distributed in multiple machines and can communicate with each other using the XMPP in a publish/subscribe pattern, and are now working on combining this system with our previous reasoner to perform continuous reasoning on streaming RDF data in a distributed manner.

References

1. Abadi, D., Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Lee, S. Stonebraker, M., Tatbul, N., Zdonik, S.: Aurora: A New Model and Architecture for Data Stream Management. The VLDB Journal, 12(2), 120--139 (2003)
2. Arasu, A., Babcock, B., Babu, S., Datar, M., Ito, K., Motwani, R., Nishizawa, I., Srivastava, U., Thomas, D., Varma, R., Widom, J.: STREAM: The Stanford Stream Data Manager, IEEE Data Engineering Bulletin, 26 (2003)
3. Della Valle, E., Ceri, S., Barbieri, D.F., Braga, D., Campi, A.: A First Step Towards Stream Reasoning. In: Domingue, J., Fensel, D., Traverso, P. (eds.) FIS 2008. LNCS, vol. 5468, pp. 72--81. Springer, Heidelberg (2009)
4. Barbieri, D., Braga, D., Ceri, S., Grossniklaus, M.: An execution environment for C-SPARQL queries. In: 13th International Conference on Extending Database Technology. ACM, Lausanne, Switzerland (2010)
5. Forgy, C.: Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. Artificial Intelligence, 19, 17--37 (1982)