Efficient Multi-View Maintenance in the Social Semantic Web

Matthias Bröcheler
University of Maryland
A.V. Williams Building
College Park, MD 20742, USA
matthias@cs.umd.edu

Andrea Pugliese
Università della Calabria
Via P. Bucci, 41/C
Rende, Italy
apugliese@deis.unical.it

V.S. Subrahmanian
University of Maryland
A.V. Williams Building
College Park, MD 20742, USA
vs@umiacs.umd.edu

ABSTRACT

The Social Semantic Web (SSW) refers to the mix of RDF data in web content, and social network data associated with those who posted that content. Applications to monitor the SSW are becoming increasingly popular. For instance, marketers want to look for semantic patterns relating to the content of tweets and Facebook posts relating to their products. Such applications allow multiple users to specify patterns of interest, and monitor them in real-time as new data gets added to the web or to a social network. In this paper, we develop the concept of SSW *view servers* in which all of these types of applications can be simultaneously monitored from such servers. The patterns of interest are views. We show that a given set of views can be *compiled* in multiple possible ways to take advantage of common substructures, and define the concept of an *optimal merge*. We develop a very fast MultiView algorithm that scalably and efficiently maintains multiple subgraph views. We show that our algorithm is correct, study its complexity, and experimentally demonstrate that our algorithm can scalably handle updates to hundreds of views on real-world SSW databases with up to 540M edges.

Categories and Subject Descriptors

H.3.0 [**Information Systems**]: Storage and Retrieval

Keywords

RDF, Graph Database, View Maintenance

1. INTRODUCTION

Graph data is proliferating today. The W3C's RDF ("Resource Description Framework") is an ideal framework to represent the semantic content of documents (including web pages, blogs, tweets and Facebook posts). At the same time, social network researchers have developed detailed methods to study relationships amongst members of a social network (e.g. friend, follow, relationships). We refer to the combination of the two as the "social semantic web" (SSW) consisting of both kinds of links. In this paper, we are interested in developing the techniques needed to support SSW *view servers* that *simultaneously* monitor large numbers of diverse views over large SSWs.

Subgraph matching has been studied extensively in both the Semantic Web community [1, 2] and the database and social network communities [3, 4]. Recent results have shown sub-second processing times for complex queries on real world data sets with over a billion edges [4]. Views and view maintenance have also been studied in the context of graph and/or RDF databases [5, 6, 7, 8].

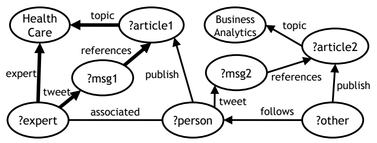

Figure 1: Example query Q1

In this paper, we build upon this past work on view maintenance in graph data to solve a different problem: suppose an SSW database $\mathcal{S}_t$ (at time t) is stored on a view server with a set $\mathbf{Q} = \{Q_1, \ldots, Q_k\}$ of registered views that need to be tracked over time. Furthermore, suppose we know the answer to each view Q_i at time t. Suppose now that some *updates* occur at time t – how can the view server incrementally compute the answer to the views in $\mathbf{Q}$ so that at time $(t+1)$, they represent the correct answer w.r.t. the database $\mathcal{S}_{t+1}$ that results after the updates? This problem is extremely important because, e.g., over 60M tweets are posted daily – we would like to incrementally compute these results rather than computing them from scratch.

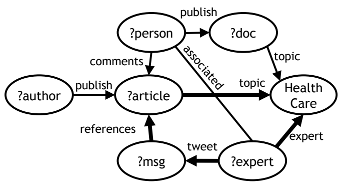

Figure 2: Example query Q2

As a motivating example, consider a SSW similar to the facebook social graph whose nodes represent individuals, companies, articles, messages, and topics. The links represent relationships between nodes such as employee-employer relationships, reference relations, publish relations, and tweet/retweet relationships.

A recruiter might want to find individuals with expertise in *health care* and *business analytics* using query Q1 shown in Figure 1. The person of interest is denoted by the vertex labeled *?person* (question marks denote variables). The query considers having published an article on the topic of health care which is referenced by an expert on the subject as an indication of knowledge (left part of the query graph).

Subgraph matching can also be used by an analyst for knowledge discovery. Query Q2 in Figure 2 searches for all authors and articles on the topic of Health Care which have been commented on by an individual who publishes on the subject and which has been referenced by an expert on health care.

2. MULTI-VIEW MAINTENANCE

To date, there is no approach to the problem of simultaneous view updates of multiple views – past work only deals with one view at a time. Updating multiple views simultaneously can result in database performance improvements, because the subgraphs that are shared among view graphs only need to be answered once. For our simple example, the shared subgraphs between query Q1 and Q2 are indicated by bold arcs.

However, before we can exploit common subgraph structures during view updates we need to identify it. For this purpose, we develop the concept of a *merged view* that leverages *common subgraphs* amongst the views in **Q** to identify a "center" edge that view maintenance will focus on. The core idea is to overlay all views in **Q** into a single merged view for each possible center edge. At view update time, we then only need to evaluate the merged view where the center edge type corresponds to the inserted edge type. There can be many possible merged views and we therefore define the concept of an optimal merged view where optimality is defined in terms of degree of overlap between views. Intuitively, the higher the overlap the less processing has to be done during view updates. Figure 3 shows the optimal merged view of Q1 and Q2 on the "tweet" center edge where different edge strokes indicate different overlaps as explained in the legend.

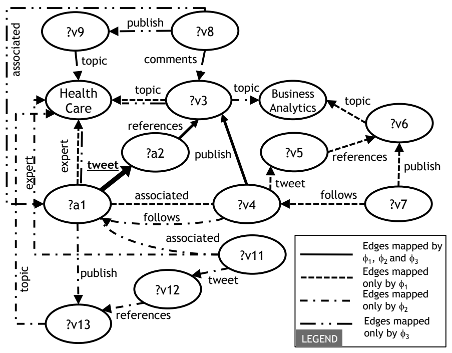

Figure 3: Optimal merged view of $\{Q1, Q2\}$ for the tweet edge.

Finding an optimal view is NP-hard, however, we developed merged-view search algorithm that can be transformed into a heuristic version that is significantly faster while producing near optimal results.

At view update time, we only need to process a single merged view since all affected views are contained therein. However, evaluating merged views is significantly different from answering standard subgraph matching queries since we have to take the degree of overlap of individuals edges into account when evaluating query plans and need to keep track of the status for each contained view. Our merged-view answering algorithm, MultiView, achieves these goals by using incrementally updated data structures for efficient processing.

3. RESULTS

We thoroughly evaluated the multi-view maintenance algorithm on 6 distinct real-world SSW datasets of varying size and density, the largest of which has over 540M edges and 11M vertices. We automatically generated sets of queries of varying complexity (i.e. number of edges and vertices in the query) and overlap to be maintained as views for each dataset. We generated 12,000 queries in total, with 4 to 11 vertices and 4 to 16 edges.

We compared the performance of our MultiView algorithm against the state-of-the-art view maintenance approach of updating each view individually.

We conducted 750 individual experimental trials for all 6 datasets where each trial measured the performance of adding 10,000 edges while updating a randomly chosen set of views of different size. In 94.4% of all trials, MultiView outperformed the baseline, by 477% on average. Figure 4 shows those statistics for each of the 6 SSW datasets. The grey dot indicates the percentage of trials on which the MultiView algorithm outperformed baseline (right vertical axis). The bar shows the average performance improvement of MultiView comapared against the baseline (left vertical axis). We observe that MultiView performed exceptionally well on the Flickr dataset with an average 9-fold improvement on almost 100% of all trials. Interestingly, the MultiView algorithm performed worst on the smallest dataset where it outperformed the baseline only 85% of all trials.

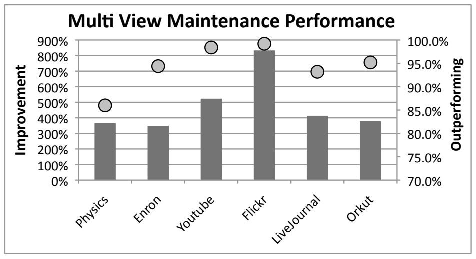

Figure 4: Performance of MultiView compared to baseline for each dataset

4. REFERENCES

[1] A. Harth, J. Umbrich, A. Hogan, and S. Decker, "YARS2: A federated repository for querying graph structured data from the web," in *ISWC*, 2007, pp. 211–224.

[2] T. Neumann and G. Weikum, "Scalable join processing on very large RDF graphs," in *SIGMOD*, 2009.

[3] M. Bröcheler, A. Pugliese, and V. S. Subrahmanian, "COSI: cloud oriented subgraph identification in massive social networks," in *ASONAM*, 2010.

[4] ——, "A Budget-Based algorithm for efficient subgraph matching on huge networks," in *ICDE Workshops*, 2011.

[5] A. Gupta, I. S. Mumick, and V. S. Subrahmanian, "Maintaining views incrementally," in *ACM SIGMOD Record*, vol. 22, 1993.

[6] Y. Zhuge and H. Garcia-Molina, "Graph structured views and their incremental maintenance," in *ICDE*, 1998.

[7] R. Volz, S. Staab, and B. Motik, "Incremental maintenance of materialized ontologies," in *CoopIS/DOA/ODBASE*, 2003.

[8] E. Hung, Y. Deng, and V. S. Subrahmanian, "RDF aggregate queries and views," in *ICDE*, 2005.