Challenges in Supporting Faceted Semantic Browsing of Multimedia Collections

Daniel Alexander Smith[1], Alisdair Owens[1], mc schraefel[1], Patrick Sinclair[1], Paul André[1], Max L. Wilson[1], Alistair Russell[1], Kirk Martinez[1] and Paul Lewis[1]
[1] IAM Group, School of Electronics and Computer Science, University of Southampton, Southampton, United Kingdom
{das05r,ao,mc,pass,pa2,mlw05r,ar5,km,phl}@ecs.soton.ac.uk

Abstract. We discuss three approaches, 3store, D2R and MySQL, we have explored to support efficient querying of multimedia data sources via mSpace, a rich UI. Our results underline key research challenges facing the development of high performance RDF query layers to support complex real-time UIs.

Keywords: triple stores, RDF browser, mSpace, multimedia indexing

1 Introduction

Web based interfaces to multimedia collections traditionally enable collection search by simple keyword queries which produce a list of links to be explored further. More sophisticated user interfaces (UIs) also allow browsing by single topics/categories/facets[1] which can then be sorted in a variety of ways,[2]. Over the past several years we have been looking at mechanisms to enable richer strategies for exploring multimedia archives. The mSpace framework presents a multicolumn faceted browsing interface that allows a person to select instances in a column/facet on the data, and have the data in the columns to the right be filtered by that selection. Each selection also populates information about that selection into an associated pane. The columns are placed in a "slice" and can be moved around. Likewise different facets or dimensions in the data space can be added or subtracted from that slice. Information about any selected instance can also be saved for later reference. This approach has been described in detail elsewhere [1].

In the original mSpace UI, each column was populated one at a time, much like the Apple OS X Finder: the first column has a set of data; a selection in that set determines the instances which populate the next column; a selection must be made in that column to populate the one next to it. In user studies, we saw that people found the interface more tractable if every column was fully populated in advance. We call this approach "pre-pop." A feature added to complement pre-pop is "backward highlighting." With backward highlighting, a selection means the data in the columns to the right are filtered, showing every possible result within the restrictions applied

[1] http://flamenco.berkeley.edu/demos.html
[2] http://www.open-video.org/

by the selection; second, the *possible* paths back from the initial selection in the columns to the left of the selection are highlighted. Thus, backwards highlighting provides additional cues for users to understand with what information their current selection is associated. While effective, these features have meant significantly increased query complexity and hence an increased hit on performance.

In the following sections we describe our experience with three approaches to optimize query performance and UI experience where our efforts have been motivated to maintain a Semantic Web deployment; we conclude with consideration of research challenges for future performance with heterogeneous data sources.

2 3Store: Performance Testing Semantic Web Querying

The mSpace framework was designed initially as a Semantic Web system in order to support the aggregation of heterogeneous data sources, and to deliver the above exploratory search experience [2] to users. In previous iterations of mSpace, we utilised 3store [3], an RDF triple store backed by MySQL as the storage and query layer. This was motivated by the attractiveness of RDF(S) as a data format: it allows for relatively simple integration of a wide variety of data sources, as well as basic inferential capabilities. Further, it uses a simple, standard query language (initially RDQL, now SPARQL), allowing people or agents to browse the dataset using mechanisms other than the mSpace browser. A forerunning project, CSAKTiveSpace [4], was based on the same technology and had coped with a substantial large dataset (40 million+ triples), we were confident that 3store would offer sufficient performance to support mSpace. This was initially borne out: our early similarly sized datasets performed very adequately.

As we implemented the new pre-pop and backward highlighting features on a dataset of approximately 100 times the size of CSAKTiveSpace's, query response times became unacceptably slow for real-time interactive use. Pre-pop in particular can result in many more queries being performed due to having to reload every column, rather than just one, some of which can be particularly demanding if the selections made do not restrict the search space very much. Performance issues were particularly noticeable on queries that returned a large result set, or less predictable results. For Example, an issue in optimising 3store for high query performance is that it is a tool for storing generic RDF, and a given piece of RDF data does not have a predictable structure. This means that it is difficult to generate a representative, optimised schema for 3Store's MySQL backend. As it stands, 3store conceptually stores a triple in one table with columns including a hash of each of subject, predicate, and object. This table (as well as a few supporting tables) are extremely long, containing a row for triple, and data is retrieved via self-joins. This schema gives the MySQL query optimiser little opportunity to perform significant optimisation on queries, and gives us relatively little control over indexing. We needed to find a solution that would offer us the ability to define a more detailed schema that more closely represented the data while providing us with a SPARQL[3] interface to

[3] http://www.w3.org/TR/rdf-sparql-query/

minimise the effort involved with a code transition, and to maintain Semantic Web compatibility. This motivation lead us to try D2R.

3 D2R Server: Best of Both Worlds - or Not Quite?

D2R Server[5] is an application layer that translates SPARQL queries into SQL, in order to provide a "semantic shim" on to a relational database. The key reason for changing the storage layer was that specific compound indexes across our schema could be created in a way that was not possible with the triple-storage method that 3store used. This approach also provides more structurally specific information to the SQL query optimiser, thus providing additional performance increases.

The subsystem that configures mSpace for a specific dataset is known as the mSpace Model [1]. Previously this was stored in the triplestore, and queried out as part of the processing. Due to the way D2R is configured as a layer between a relational database and a SPARQL query-point, asserting a small amount of arbitrarily structured RDF like the mSpace Model would have been problematic to implement with D2R, so the decision was made to instead alter the mSpace server such that the model was stored separately to the data. This is also attractive as it gives the additional benefit that remote SPARQL end-points that we do not control can be configured locally and explored using mSpace, without any agreement with, or alteration of, the remote data.

The solution looked promising, but ultimately was not suitable for several reasons. First, the SQL statements that the D2R Server created were inefficient, and considerable and timely effort would be needed to experiment with optimizing these queries. That D2R Server has been implemented using Java means that memory is filled much earlier than if the database was queried directly by the mSpace Server. It was also not possible to perform keyword matching efficiently across the data, due to a lack of an indexable query system for keyword string matching, with only the SPARQL REGEX syntax supported. D2R performs these in its local virtual machine memory space, rather than in the database itself, again causing a performance hit. At this point, we wanted to see if it were possible to achieve the kind of performance we needed by going directly to a highly indexable, tuneable system, so we tried MySQL.

4 SQL and MySQL: Losing the Semantic Web for Performance

SQL is a lower-level language than SPARQL. In SPARQL, one describes the pattern of data that is desired, and it is the responsibility of the triple store to decide how to retrieve the information. In SQL, it is possible to express the same query in a variety of different ways, depending on how we wish the database to achieve the result. This is both an advantage and a disadvantage: it offers the ability to tweak our queries manually to achieve optimal performance, but this is a laborious process. Ultimately it would be preferable to have the backing data store do the optimising work for us. Since the mSpace server now had the model separated from the data store, moving from a D2R storage to a straight SQL one was relatively cheap and offered

considerable performance gains over a system using the same database, but with a D2R layer included. It also allowed the system to capitalise on the full-text indexing available in MySQL for scalable string searches of the data.

The focus of this approach is purely the performance of getting information to the UI in real time. While this means that the UI is responsive and scales well to large datasets, it means sacrificing the original Semantic Web aspects of this work. As such, future work will re-explore the possibilities of re-incorporating support for aggregated heterogeneous sources, looking at how best to translate the performance gains we do get from SQL while utilising Semantic Web tools for data aggregation.

5 Conclusions

Integration of new, more backend-intensive technologies, as well as working with larger collections of data has raised interesting research and technical challenges for mSpace in particular and Semantic Web technologies in general. The very reason for the Semantic Web is to bring together heterogeneous data to enable rich queries of it: our work has challenged Semantic Web technologies to support such interfaces for even a single data set. We discovered that existing triple storage solutions were unable to provide interactive-level real-time performance as our needs grew, and were forced to implement traditional relational database support. While this transition aided performance to the UI, it has meant that benefits gained from the use of Semantic web technologies, such as ease of data aggregation, and the simple interface of SPARQL, are lost. Future optimisations of the mSpace server may yield some performance improvements, but in order to perform the desired move back to solely using Semantic technologies there is a requirement for triple stores with response times several orders of magnitude better than that offered by 3store. This will likely require stores that have moved beyond the long triple list format, and begun to involve indexing and query optimisations that cope with RDF data's unpredictable structure.

References

1. schraefel, m. c., Smith, D. A., Owens, A., Russell, A., Harris, C. and Wilson, M. L. (2005) The evolving mSpace platform: leveraging the Semantic Web on the Trail of the Memex. In Proceedings of Hypertext, 2005, Salzburg
2. schraefel, m.c., Wilson, M., Russell, A., and Smith, D. A. 2006. mSpace: improving information access to multimedia domains with multimodal exploratory search. *Commun. ACM* 49, 4 (Apr. 2006), 47-49
3. Harris, S. Gibbins, N., 3store: Efficient Bulk RDF Storage. In 1st International Workshop on Practical and Scalable Semantic Systems, (Sanibel Island, Florida., 2003), 1-15
4. Shadbolt, N., Gibbins, N., Glaser, H., Harris, S., schraefel, m.c., "CS AKTive Space, or How We Learned to Stop Worrying and Love the Semantic Web," *IEEE Intelligent Systems*, vol. 19, no. 3, pp. 41-47, May/Jun, 2004
5. Bizer, C., Cyganiak, R. D2R server-publishing relational databases on the Semantic Web (poster). In Proceedings of the International Semantic Web Conference (Florida, ISWC), (2003)