Efficient and Effective Source Selection in the Web of Data: a Smart Caching Approach

PhD Session Paper

Francesco De Fino
Supervised by: Barbara Catania, Giovanna Guerrini

University of Genoa, Italy
francesco.defino@dibris.unige.it

1 Problem Statement and Motivation

In the last decade, the traditional Web is evolved in the *Web of Data*, where huge collections of structured data are gathered over distributed, heterogeneous sources. These datasets are often exploited much below their potential due to difficulties in accessing them. Processing complex requests on diverse and dynamic sources requires: (i) source selection; (ii) actual processing of the request on the relevant, and usually big in size, sources. The overall process may not guarantee user satisfaction, since the request may be processed on inaccurate, incomplete, unreliable data; or the processing time may be inadequate to the query urgency. In [1], the authors claimed that user profiles and request contexts, data and processing quality, and information about similar requests recurring over time could be exploited to inform query processing and cope with these issues.

In this thesis, we focus on recurring retrieval needs for improving query processing. Recurring retrieval needs have been considered in query processing for a very long time in terms of materialized views and caching approaches. Both of them suffer from some problems when applied to the reference scenarios. Materialized views associate a result with a given query even if, for improving efficiency, additional information could be of interest (e.g., the set of used data sources). On the other hand, caching approaches usually rely on precise query matching, not always suitable in the reference environments, and, similarly to materialized views, cached queries are usually associated with query results.

The aim of the thesis is to exploit how information about similar requests recurring over time can be exploited in order to efficiently and effectively process complex requests on heterogeneous and dynamic data sources, while guaranteeing user satisfaction and limiting as much as possible user interactions. In particular, we propose a new smart caching technique that gets over the limitations of current approaches by: (i) taking advantage of prior processing in order to obtain *shortcuts* to different points in the query processing stack, with a special emphasis on source selection, thus providing a multilayer caching approach;

SEBD 2018, June 24-27, 2018, Castellaneta Marina, Italy.

(ii) relying on relaxed matches; (iii) providing a human-in-the-loop solution by which user feedback as well as query and data contexts are taken into account in cache management.

We claim that the proposed approach could be quite effective in processing complex requests on diverse and dynamic sources. Indeed, by maintaining not only query results but also more general processing information collected during processing, like the set of selected sources, in distinct but related caches, we will be able to cope with the trade-off between data dynamicity and accuracy of the result. By exploiting relaxation, performance can also be improved, since the number of cache hits will increase. Relaxation, as well as the adoption of a human-in-the-loop solution, can finally increase user satisfaction, in terms of both quality of data (accuracy and completeness of the result) and quality of service (execution time).

Our smart cache approach is also quite innovative: as far as we know, no multilayer caching approach, taking into account source selection, has been proposed so far. The exploitation of relaxation and human-in-the-loop features in cache management is also quite original, moving to a traditional physical task features that are typical of higher level processing steps.

The proposed approach is an instantiation of the framework proposed in [2,3] in the context of Linked Data but we claim it can be easily extended to other graph-based contexts. Data sources correspond to RDF data sources[1] and queries are represented in SPARQL.

2 A Relaxed Caching Approach for Source Selection

We first introduce the reference architecture for source selection caching we rely on in the overall multilayer approach. Then, a preliminary proposal for relaxed cache matches is discussed.

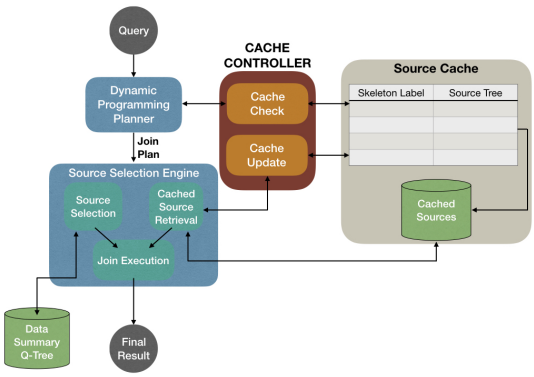

Fig. 1. Query processing architecture

[1] A RDF set of triples available on the Web.

Reference architecture. We extend the caching framework for RDF data and SPARQL queries in [6] to deal with source selection. It represents query graphs through canonical labels generated by a variation of the *Bliss* algorithm [5].

Fig. 1 shows the resulting reference architecture. Given a query Q as input, the *Dynamic Programming Planner* module, from [6], identifies the best query plan relying on a dynamic programming approach. Query plan components correspond to connected subgraphs of Q, whose results have then to be joined for generating the final result. For each connected subgraph, a cache look up is performed; if no match is found, the subgraph will be executed by a traditional graph query processing engine. The cost model takes into account the cost for the cache look up and the cost for traditional graph query execution.

In order to design a cache for source selection, we first modified the cache module with the aim of associating sets of source identifiers, instead of sets of data items, to cached queries. All cached queries sharing a common structure (called *skeleton*) are cached together, relying on a tree structure. The processing engine has then been modified to cope with source selection execution. To this aim, we relied on the index-based source selection approach proposed in [7] (but any other source selection technique can be used as well): a data summary, called QTree, is used to index available sources by representing triples as points in a three-dimensional space and source portions as hyper-rectangles in such a space. Preliminary results show that our smart caching framework outperforms the approach in [7] for different types of recurring SPARQL queries.

Relaxation. Exact matching can represent a limitation to fully exploit caching potential in speeding up source selection. If cached queries represent recurrent requests, exact matching precludes the planner to choose entries that are similar but not exactly equal to the current user request. For this reason, we propose to extend our smart caching approach with query relaxation to fully exploit the cache. The idea is to exploit a reference ontology $\mathcal{O}$ expressed by RDFS [4] and to consider relaxed matches w.r.t. the ontology. As a starting point, we consider $\mathcal{O}$ as a predicate taxonomy, i.e., a taxonomy in which predicates are related by a *rdfs:subPropertyOf* relationship.

The problem is now the following: given a query Q and n cached queries Q_1,...,Q_n, find $\overline{Q} \in \{Q_1, ..., Q_n\}$ such that $\overline{Q}$ generalizes Q and it has the minimum relaxation cost $relax\_\mathcal{C}$ from Q, i.e., $relax\_\mathcal{C}(Q, \overline{Q}) = min_{i=1}^{n} relax\_\mathcal{C}(Q, Q_i)$. The relaxation cost is defined on query canonical representations. $relax\_\mathcal{C}(Q_1, Q_2)$ returns ∞ if Q_2 does not generalize Q_1 (i.e., it is not possible to find in Q_2, for each triple pattern in Q_1, a triple pattern with a more general predicate, according to the taxonomy $\mathcal{O}$, and with the same subject and object). Otherwise, the returned value depends on how "far" is each triple pattern in Q_2 from the corresponding one in Q_1. Distance is defined in terms of the path length between predicates in the taxonomy $\mathcal{O}$.

In order to efficiently select query $\overline{Q}$ as described above, we assume that cached queries are indexed by a tree, where each node corresponds to a query triple pattern and each path from the root to a leaf corresponds to a cached

query. The cached query at the minimum cost is then selected by a variation of the A^* algorithm.

We are currently working on an extension of this algorithm to take into account not only the generalization cost but also the cost for accessing sources. To this purpose, we plan to associate some statistics related to sources with each index leaf and use them, combined with the generalization cost, in computing the total cost. Another relevant extension concerns the usage of more generic ontologies for defining relaxed matches.

3 Future Research Plan

We plan to extend the work presented in Section 2 in several directions:

- Integrating the relaxation approach in the reference architecture.
- Integrating the source cache with a traditional cache for query processing over Linked Data and defining specific cache policies for a combined usage of both caches, depending on data dynamicity and target result accuracy, for both precise and relaxed queries.
- Extending the proposed smart caching approach so that the choice of the queries to cache is not only based on a recurrence principle, but also on a diversification principle, thus avoiding to keep in cache queries that are too similar or one is a slight relaxation of the other.
- Extending the proposed smart caching approach to bring the human in the loop, through the usage of specific cache management policies taking care of user feedbacks as well as query and data contexts.
- Experimentally evaluating the proposed approach, with reference to Linked Data sets, different query patterns, and different recurrence patterns. We will start with a synthetic workload but we will then look for a real workload. The main aim of the evaluation will be to measure the increase in efficiency (in terms of source selection and overall query processing costs), with varying amounts of available space, and the result accuracy due to the degree of approximation introduced by relaxation.

References

1. B. Catania et al. Wearable Queries: Adapting Common Retrieval Needs to Data and Users. *DBRank Workshop* (co-located with VLDB 2013), 2013.
2. B. Catania et al. Recurring Retrieval Needs in Diverse and Dynamic Dataspaces: Issues and Reference Framework. *EDBT/ICDT Workshops* 2017.
3. F. De Fino et Al. Exploiting Recurrent Retrieval Needs in Querying Heterogeneous and Dynamic Graph Dataspaces. *SEBD* 2017.
4. R. Frosini et al. Flexible Querying for SPARQL. *Semantic Web* 8(4), 2017.
5. T. Junttila et al. Engineering an Efficient Canonical Labeling Tool for Large and Sparse Graphs. *ALENEX. Society for Industrial and Applied Mathematics*, 2007.
6. N. Papailiou et al. Graph-aware, Workload-adaptive SPARQL Query Caching. *ACM SIGMOD* 2015.
7. J. Umbrich. et al. Comparing Data Summaries for Processing Live Queries over Linked Data. *World Wide Web* 14(5-6), 2011.