Top-k reasoning for the Semantic Web

Stefan Schlobach

Vrije Universiteit Amsterdam

Abstract. Data on the Semantic Web (SW) is ordered, for instance, through orderings over literals (e.g., age, heights, location, rating etc), resources (e.g., popularity) or triples (e.g., provenance, confidence, timestamps). Recently orderings have been established as first class citizens in an extended SPARQL algebra, with an efficient calculus for finding, e.g., top-k answers. With this paper we attempt to stimulate a similar initiative for **reasoning**, by 1) discussing various types of orderings on the SW, 2) introduce the notion of top-k closure, and 3) sketch an algorithm for calculatign this closure based on top-k database joins.

1 The Web of Data is ordered

Orderings are omnipresent on the Web of Data. Sometimes resources are directly, and **explicitly**, described by triples containing ordered literals, providing information about the number of inhabitants of cities, ratings of restaurants, longitude and latitude or dates of birth etc. However, most data on the Web of Data also comes with a variety of **implicit** ordering measures, as resources are not just ordered by size or age, but also by meta-properties such as popularity. It is well-known that one of the prime criteria for ranking search results in Information Retrieval is based on the trust-worthiness of results, which is calculated using PageRank as a proxy, i.e. the number and importance of web-sites linking to the search result. Similarly, in web-scale semantic search, we cannot ignore the trust-worthiness of resources, and their available information. This implies that resources in Semantic Web ontologies intrinsically come with implicit orderings, which could/should be taken into account when answering queries. But not only are resources ordered, so are triples. There is plenty of literature on how to extend triple with meta-data and annotations, and most of those ideas induce (at least partial) orderings on triples: based on uncertainty, temporal dimensions, provenance, trust to name but a few [6,5,1]. Clearly, orderings occur everywhere, and in a variety of forms and contexts, and querying and reasoning over ordered data becomes a critical problem on the Web of Data.

The Semantic Web community has started to cater for this need by including the ORDERED-BY operator into SPARQL, and recently, by extending the SPARQL algebra by a ranking operator [2]. This allows the application of optimisation from recent work in ranking in relational databases by extending the idea of efficient top-k joins to the SPARQL execution model. Unfortunately,

these efforts have not yet found their counter-part on the reasoning side. More concretely, although novel querying algorithms make use of the orderings of resources and triples, and although a variety of extensions of RDF and more expressive ontology languages (such as RDFS and OWL) with annotations have been proposed there is to the best of our knowledge no attempt to provide generic methods for efficient reasoning over such orderings as of yet.

Let us introduce the problem (slighly misusing notation for simplicity):

```
A'dam owl:sameAs dbpedia:Amsterdam.
LaTheatina hasRanking "5" ; locatedIn A'dam.
VURestaurant hasRanking "1" ; locatedIn A'dam.
```

```
LaValade rdf:type FrenchRestaurant;
         locatedIn dbpedia:Amsterdam.
QuickFood hasRanking "2" ;
          locatedIn Diemen.
```

Given Semantic Query languages such as SPARQL we can now query this database, for example, for a ranked list of objects according to the given property `hasRanking`. A SPARQL query such as `SELECT * WHERE (?X hasRanking ?Y AND ?X locatedIn A'dam) ORDERED-BY ?Y` returns LaTheatina and VURestaurant in this order. In addition, one can restrict the number of results by adding a `LIMIT k` to get top-k results, and combine different rankings with richer scoring functions. For queries with monotonic scoring functions over RDF data-bases efficient query execution has recently been presented [2], which is based on the exhaustive literature on top-k query answering in databases [4] (a wide class of streaming algorithms [3]). The idea is to avoid the calculation of full joins during query processing by using the underlying order of the results for the individual predicates, and by carefully combining those ordered results only if necessary. It has been shown that these methods can significantly increase efficiency of top-k queries in SPARQL.

The problem starts when reasoning is required to materialise available information in a more expressive knowledge base (e.g. containing OWL or RDFS operators). Take the following terminological information:

```
FrenchRestaurant rdfs:subClassOf (hasRanking xsd."4").
(locatedIn Diemen) rdfs:subClass (locatedIn dbpedia:Amsterdam).
```

Assuming the intended semantics of RDFS and OWL, we can derive that LaValade has a ranking of 4, and that QuickFood and the VU are located in A'dam, which is the same as dbpedia:Amsterdam. The top-2 answers to the above query are then LaTheatina and LaValade (in that order). The only currently known way for answering this query is to fully materialise the knowledge-base first, and then apply ranking over this set. The question is whether full materialisation can be avoided while still being complete w.r.t. top-k queries.

2 Reasoning with Ordered Triples

In the most general case we consider an order-aware ontology $\mathcal{O}$ to be a set of triples with possibly multiple scoring functions over the set of resources as well as a scoring function over triples. Let us consider for simplicity a single monotonic scoring function $\mathcal{F}$ that is globally used, and that possibly combines

the scoring for resources and triples. The answer to a top-k SPARQL query `SELECT * where` QP `ORDERED-BY` $\mathcal{F}$ is a ranked list of mappings according to the SPARQL algebra, where each mapping $\mu(QP)$ maps the query pattern QP to a set of ground triples t entailed by $\mathcal{O}$ and ordered by $\mathcal{F}(\mu)$.[1] We will call the position of a mapping μ in the ordering of $\mathcal{F}$ its rank (and abbreviate $rank_{\mathcal{F}}(\mu)$).

The problem of this definition is that the current way for answering those queries is full materialisation of $\mathcal{O}$ w.r.t. the intended semantics in order to efficiently check entailment, before ordering the query results by $\mathcal{F}$. The research question stipulated in this position paper is whether it is possible to find a subset of the full closure of an ontology $\mathcal{O}$, which allows top-k query answering and can be calculated significantly more efficiently than the full materialisation of $\mathcal{O}$.

Definition 1. *Let $\mathcal{O}$ be an ontology with possibly multiple scoring functions over resources and triples and a monotonic scoring function $\mathcal{F}$, and . a set of query patterns Q. The* ***top-k closure of O w.r.t. Q*** *is a subset of the closure of O that is sufficient for computing the top-k answer to any query in Q; i.e., a triple t must be in $TK(O)$ if there is a query-pattern q in Q and a mapping μ s.t. $\mu(q)$ is entailed by O, t is in $\mu(q)$ and $rankF(u) \leq k$.*

This definition comes with a number of open questions that we will take as a road-map for future research, such as the following:

1. Can such a top-k closure be calculated efficiently, more efficiently than just calculating the full closure, and do ranking later?
2. What is the relation between the choice of scoring function and query patterns and their respective top-k closure
3. Can we combine and reuse different such closures for different functions and query patters?

A sketch of how to calculate top-k entailment We briefly want to sketch an algorithm for efficiently calculating top-k entailment for rule-based ontology languages. We assume that there is calculus using a set of rules $\{p_1 \rightarrow c_1, \ldots, p_n \rightarrow c_n\}$, where p_i are sets of atomic triple patterns, the elements which are called the premises, c_i a triple pattern called consequence.

Our initial idea was to devise a backward reasoning algorithm (such as [8,7]) which we thought would be suitable as it would always only have to pull the top-k premises from which a top-k consequence could be derived. This, however, will not work as all premises would have to have to be checked for completeness, even those not in the top-k of $\mathcal{F}$.

Given the fact that the application of a rule can be seen as a join operation, we can however make use of the same optimisations proposed for ranked query answering. Given that we want to derive the top-k conclusions, we can make use of the fact that the matches for the premises are ordered. This means we will not have to pull the full set of triples instantiating each of the premises, but just enough to perform the top-k join, which we know how to do efficiently.

[1] We refer to the SPARQL spec for the definitions of the used terms.

Basically, for every rule $p_i \rightarrow c_i$ the instantiation of the conclusion c_i has to be added, where instantiation contains the top-k answers w.r.t the joins of the variable instantiations optained by retrieving the explicit answers for the query patterns p_i from the knowledge base, which can be done efficiently as proposed in [2]. This process needs to be repeated until saturation.

Of course, this does not necessarily give us a minimal top-k closure, as we need to exhaustively loop over the rule application until all top-k results are found. This means that through the calculus we will derive too many consequences that are not top-k answers. Still we conjecture that this top-k closure is useful and efficient to calculate.

3 Conclusion

We have introduced the problem of top-k reasoning for the Web of Data, which is about finding ranked results for order-aware queries. Those can be both for explicit orderings, such as numeric information on height, date-of-birth, time or location, but also for implicit orderings over resources and triples, often including meta-data (trust, provenance, confidence etc).

We have formally introduced the notion of a top-k closure as superset of an ontology which is sufficient to answer top-k queries in a complete way (without requiring further calculations). We finally sketch an idea for a rule-based calculus to determine such a top-k closure, which we conjecture to be efficient as it helps avoiding unnecessary join-calculations by using top-k join algorithms [4]. Of course, this being a position paper, this is a collection of ideas rather than a presentation of results.

References

1. Bonatti, P.A., Hogan, A., Polleres, A., Sauro, L.: Robust and scalable linked data reasoning incorporating provenance and trust annotations. J. Web Sem. 9(2), 165–201 (2011)
2. Bozzon, A., Della Valle, E., Maagliacane, S.: Towards an efficient sparql top-k query execution in virtual rdf stores. In: Proceedings of DBRank 2011
3. Henzinger, M.R., Raghavan, P., Rajagopalan, S.: External memory algorithms. chap. Computing on data streams, pp. 107–118. American Mathematical Society, Boston, MA, USA (1999)
4. Ilyas, I.F., Beskales, G., Soliman, M.A.: A survey of top- query processing techniques in relational database systems. ACM Comput. Surv. 40(4) (2008)
5. Lopes, N., Polleres, A., Straccia, U., Zimmermann, A.: AnQL: SPARQLing up annotated RDFS. In: Proceedings of ISWC 2010. pp. 518–533 (2010)
6. Schenk, S.: On the semantics of trust and caching in the semantic web. In: International Semantic Web Conference. pp. 533–549 (2008)
7. Simancik, F., Kazakov, Y., Horrocks, I.: Consequence-based reasoning beyond Horn ontologies. In: IJCAI 2011. pp. 1093–1099 (2011)
8. Urbani, J., van Harmelen, F., Schlobach, S., Bal, H.: Querypie: Backward reasoning for owl horst over very large knowledge bases. In: Proceedings of ISWC 2011 (2011)