Guided Link-Traversal-Based Query Processing

Ruben Verborgh and Ruben Taelman
IDLab, Dept. of Electronics and Information Systems, UGent – imec, Belgium

Abstract. Link-Traversal-Based Query Processing (LTBQP) is a technique for evaluating queries over a web of data by starting with a set of seed documents that is dynamically expanded through following hyperlinks. Compared to query evaluation over a static set of sources, LTBQP is significantly slower because of the number of needed network requests. Furthermore, there are concerns regarding relevance and trustworthiness of results, given that sources are selected dynamically. To address both issues, we propose *guided* LTBQP, a technique in which information about document linking structure and content policies is passed to a query processor. Thereby, the processor can prune the search tree of documents by only following relevant links, and restrict the result set to desired results by limiting which documents are considered for what kinds of content. In this exploratory paper, we describe the technique at a high level and sketch some of its applications. We argue that such guidance can make LTBQP a valuable query strategy in decentralized environments, where data is spread across documents with varying levels of user trust.

1 Querying Data on the Web

Link-Traversal-Based Query Processing (LTBQP) enables the evaluation of SPARQL queries over hyperlinked RDF documents on the Web rather than RDF databases. Evaluating SPARQL over the Web of Data is supported by a family of *reachability semantics* [2], where the source of the RDF data used for a query is an expansible set of *seed documents* to which new documents are added dynamically based on the already obtained RDF triples.

LTBQP suffers from performance problems compared to SPARQL query evaluation over one or more RDF databases, given the dynamic nature of its sources and the impossibility of predicting which reachable documents will contribute to the result set. While heuristics can *prioritize* certain links in order to reduce response times [4], they cannot safely *prune* the document search tree. Given the Web's openness, issues such as trustworthiness and license compliance emerge when incorporating unknown documents.

These problems can be addressed by tailoring queries to a specific user and context. For instance, the query language could explicitly express which traversals are permitted [3]. However, we aim to maintain independence between applications on the one hand, and the structure of the data network and the user's content preferences on the other hand. Therefore, we introduce the concept of *Guided* Link-Traversal-Based Query Processing (Guided LTBQP), in which a user-controlled context guides the query engine's process.

2 Example Use Case

We will discuss an example using personal Linked Data [6], for which document-based data organization is common. Consider an address book app displaying contacts, operated by a user Uma with identifier *https://uma.ex/#me*. Its data demand is captured by the query in Listing 1. The contents of Uma's own profile are displayed in Listing 2. Note

```
SELECT ?friend ?name ?email ?picture WHERE {
  <https://uma.ex/#me> foaf:knows ?friend.
  ?friend foaf:name ?name.
  OPTIONAL { ?friend foaf:mbox ?email.
             ?friend foaf:img  ?picture. }
}
```

Listing 1: Application query in SPARQL

```
<https://uma.ex/#me> foaf:knows
  <https://ann.ex/#me>, <https://bob.ex/#me>.
<https://bob.ex/#me> foaf:img <bob.jpg>.
```

Listing 2: RDF from *https://uma.ex/*

```
<https://ann.ex/#me> foaf:isPrimaryTopicOf <https://ann.ex/about/>.
<https://ann.ex/#me> foaf:weblog <https://ann.ex/blog/>.
<https://ann.ex/#me> foaf:maker <https://photos.ex/ann/>.
```

Listing 3: RDF from *https://ann.ex/*

```
<https://bob.ex/#me> foaf:name "Bob";
  foaf:email <mailto:me@bob.ex>;
  foaf:img <funny-fish.jpg>.
<https://uma.ex/#me> foaf:knows
  <http://dbpedia.org/resource/Mickey_Mouse>.
<https://ann.ex/#me> foaf:name "Felix";
```

Listing 4: RDF from *https://bob.ex/*

```
<https://ann.ex/#me> foaf:name "Ann";
  foaf:email <mailto:me@ann.ex>;
  foaf:img <ann.jpg>.
```

Listing 5: RDF from *https://ann.ex/about/*

	?friend	?name	?email	?picture
1	<https://ann.ex/#me>	"Ann"	<mailto:me@ann.ex>	<https://ann.ex/about/ann.jpg>
2	<https://bob.ex/#me>	"Bob"	<mailto:me@bob.ex>	<https://uma.ex/bob.jpg>
3	<https://bob.ex/#me>	"Bob"	<mailto:me@bob.ex>	<https://bob.ex/funny-fish.jpg>
4	<https://ann.ex/#me>	"Felix"	<mailto:me@ann.ex>	<https://ann.ex/about/ann.jpg>
5	dbr:Mickey_Mouse	"Mickey Mouse"@en	*NULL*	*NULL*

Table 1: Possible results of LTBQP of the query in Listing 1 with *https://uma.ex/* as seed

how Uma's contact Ann (Listing 3) maintains her personal details in a separate document (Listing 5). Contact Bob (Listing 4) is a self-professed jokester who picks a funny picture for himself, declares Mickey Mouse to be Uma's friend, and Ann's name to be "Felix".

With traditional link-traversal-based query evaluation under several (but not all) reachability semantics [1], a query engine could fetch at least 7 documents: profile documents of 3 people (Uma, Ann, Bob), 3 documents referred to by Ann's profile (Listing 5), and the DBpedia entry on Mickey Mouse. Results could include those in Table 1.

3 Leveraging Document Linking Structure

A first issue with the evaluation of the query in Listing 1 is that more documents than needed are downloaded in order to find the results, because the query engine cannot distinguish between relevant and irrelevant documents. For example, Ann splits her profile into 3 separate documents (Listing 3), only one of which contains her contact details (Listing 5), the others listing blog posts and photos. This contrasts with Uma and Bob (Listings 2 and 4), who place their contact details directly in their profile document.

Suppose the query engine is supplied with knowledge about the linking structure of documents, for instance through a shape description language such as SHACL. The shape could detail that Ann stores her contact details in the document referred to by the `foaf:isPrimaryTopicOf` relation. Likewise, for Uma and Bob, their shape could express that all of their contact details reside inside of their main profile document. While the availability of such structural knowledge seems far-fetched on the public Web, in the context of personal data spaces within the Solid ecosystem [6], this structure needs to be explicit anyway so applications know where to *write* new data. As such, users do not need to provide this along with queries, since every person or domain can publish their own.

Based on this structure, the engine can decide to skip the links to *https://ann.ex/blog/*, *https://photos.ex/ann/*, and *http://dbpedia.org/resource/Mickey_Mouse* to collect contact details for Ann and Bob. After all, their data spaces indicate conformance to certain shapes, so for Ann it is sufficient to follow the `foaf:isPrimaryTopicOf` link to *https://ann.ex/about/*, and for Bob to remain on *https://bob.ex/*. This reduces the number of required network requests for Ann from 4 to 2.

Note that the set of query results obtained after link pruning could be different from Table 1 if the structural descriptions are not truthful, leading to the query engine to skip documents with matching triples. However, the user remains in full control of which structural descriptions are to be taken into account by the query engine.

4 Incorporating Content Policies

A second issue is that, within the query graph formed by the union of the RDF graphs of each traversed document, all triples are considered to be equally applicable. This conflicts with the real world, in which we rely on certain sources for certain kinds of facts, but not for others. For example, Bob's profile document contains false statements about Uma and Ann, and an unhelpful statement about his own profile picture.

User Uma might want to capture the notion that, for statements about her relation to others, she only considers her own profile document authoritative. In contrast, she allows people to make statements about their own name and other contact details. Yet if she specifies a profile picture for a person, she wants that to take precedence over others.

If a query engine takes these preferences into account, then the evaluation of the query from Listing 1 would yield only results the user considers relevant, namely rows 1 and 2 from Table 1. Rows 3–5 are undesired, since they are based on triples that fall outside of the user-specified content policy, which indicates what triples are to be considered. Because of this restriction, the document at *http://dbpedia.org/resource/Mickey_Mouse* does not need to be retrieved, as the triple mentioning it also falls outside of the policy. Combined with the reductions obtained in Section 3, the query engine thus only needs to download 4 documents instead of 7 to find all answers the user deems relevant, namely the 3 profile documents (Listings 2 to 4), and Ann's details (Listing 5).

This concept of content policies, which filter what triples of a given document are to be considered, has several purposes in addition to end-user trust. For example, a user could demand to skip documents with a closed license in order to avoid copyright issues, skip documents marked as suspicious by fact checkers to combat fake news, or selectively allow publication metadata from suspicious documents for reference reasons, but ignore their data triples that are regarded as nonfactual.

5 Query Semantics

As a simplification, and borrowing notation from existing literature [1], the evaluation of a SPARQL query Q over a web of documents W under *reachability-based semantics* is equal to the evaluation of Q over the union of all RDF triples extracted from documents that are transitively reachable from a set of seed documents $\mathcal{S}$, namely $[\![Q]\!]^r_W = [\![Q]\!]_{W^r}$ with $W^r = \{t \in \mathcal{T} \mid t \in \mathit{parse}(d) \wedge d \in \mathit{reachable}(\mathcal{S})\}$.

We incorporate the concept of *linking structure* through descriptions of specific linking structures $ls \in \mathcal{LS}$, where $ls(d, d', tp) = true$ iff, starting from a document d, the document d' should be considered for matches to the triple pattern tp. The linking structure associated with a given document d is retrieved via a function *obtainLS*.

We incorporate the concept of a *content policy* as a set of relevance criteria $rc \in \mathcal{RC}$, where $rc(t, d) = true$ iff a triple t from a document d is considered relevant.

This leads to an initial query semantics for guided LTBQP. Instead of Q, the user passes an augmented query $Q' = (Q, obtainLS, \mathcal{RC})$ to the query engine. We consider the evaluation of Q' under *guided link-traversal-based semantics* as the evaluation of Q over the set containing every triple that a) is obtained from any reachable document adhering to the linking structure and b) is part of the relevant set of triples for that document, so $[\![Q']\!]^g_W = [\![Q]\!]_{W^g}$ with $W^g = \{t \in \mathcal{T} \mid t \in parse(d) \wedge d \in D \wedge \exists rc \in \mathcal{RC} : rc(t, d)\}$ and where $D = \mathcal{S} \cup \{d \mid \exists d' \in D : d' \neq d \wedge ls = obtainLS(d') \wedge \exists tp \in Q : ls(d, d', tp)\}$.

6 Conclusion and Future Work

When guiding LTBQP query engines, document linking structure and content policies can vastly reduce the number of considered documents and triples, yet for different reasons. The usage of structure has *performance* as a goal, so changing results is likely undesired; content policies aim for *result alterations*, thereby possibly eliminating network requests. Since the effects occur because of differences across contexts rather than modifications to the query, guided LTBQP leads to a *context-dependent notion of result completeness*. Whereas the same query can thus have different results in different user contexts, the results are considered complete in relation to the user-bound notion of *relevant* results.

One of our main goals is bringing LTBQP techniques to competitive levels of performance for cases where it would be challenging to gather the data in a query interface. Concretely, we target decentralized networks of personal data, where each user stores their own data in their personal data vault, guarded by access control [6]. Given that orthogonal data interface features such as access control and versioning are much more straightforward to provide on simple document-based interfaces rather than more complex database-driven APIs, there exists a strong case for document-based query techniques.

This article only provides an initial sketch of the problem space. Future work is required on the description of linking structures and content policies. We need a detailed study of guided LTBQP semantics and its relation to other techniques and languages, as well as theoretical and empirical evaluations of its performance in different contexts. We plan to use the Comunica query engine platform [5] as a basis for experimentation.

References

1. Hartig, O.: SPARQL for a Web of Linked Data: semantics and computability. In: ESWC. pp. 8–23 (2012)
2. Hartig, O.: An overview on execution strategies for Linked Data queries. Datenbank-Spektrum pp. 89–99 (2013)
3. Hartig, O., Pérez, J.: LDQL: a query language for the Web of Linked Data. Journal of Web Semantics (2016)
4. Hartig, O., Özsu, M.T.: Walking without a map: Ranking-based traversal for querying Linked Data. In: ISWC. pp. 305–324 (2016)
5. Taelman, R., Van Herwegen, J., Vander Sande, M., Verborgh, R.: Comunica: a modular SPARQL query engine for the Web. In: ISWC. pp. 239–255 (2018)
6. Verborgh, R.: Re-decentralizing the Web, for good this time. In: Linking the World's Information: Tim Berners-Lee's Invention of the World Wide Web. ACM (2020)