Evolutionary RDF Quotient Summaries - A Good Description?

Christian Aeboloe[1], Olivier Pelgrin[1], and Haridimos Kondylakis[2]

[1] Aalborg University, Denmark
{caebel,olivier}@cs.aau.dk
[2] ICS FORTH, Greece
kondylak@ics.forth.gr

Abstract. The proliferation of the amount of published RDF data in recent years, thanks to the increasing numbers of community efforts as well as private initiatives, can pose significant challenges for data management. RDF graphs can often be extremely large, with up to several billions of triples, and therefore can be very challenging for systems to handle. Moreover, such graphs often evolve over time as data is added or deleted from them. Graphs summaries are a solution to handle very large graphs by reducing them into smaller graphs but are often limited to static graphs. In this paper we investigate the problem of evolving RDF graph summarization by proposing an algorithm capable of updating previously generated summaries with the changeset to a new revision.

1 Introduction

The proliferation of the amount of data available on the Linked Open Data (LOD) cloud has resulted in a rapid growth in not only the number of datasets available in the RDF format, but also in the size of each individual dataset. Furthermore, such datasets are frequently updated with new or changing information [4]. However, such datasets are often quite complex and heterogeneous in structure, making it difficult for users to make sense of.

RDF Archives, such as the one presented in [4], make it possible to store and query the entire history of updates to a given knowledge graph. This technology has a wide range of applications, such as enabling users to collaboratively ensure that knowledge graphs are up to date as in [1]. However, while several approaches have previously suggested different summarization techniques for RDF graphs [2, 5], summarizing evolutionary knowledge graphs often requires building the entire summary from scratch over the newest version. Since RDF Archiving systems tend to experience quite frequent updates, building a new summary from scratch every time an update is applied can easily become an unfeasible task.

To avoid the issue of frequently building summaries from scratch, in this assignment, we explore evolutionary RDF summaries. That is, we assess whether or not applying updates to knowledge graphs directly to the summaries of a previous version provides a descriptive summary of the updated knowledge graph.

In particular, we focus on RDF Quotient summaries [2] and propose an algorithm that, given the quotient summary of the initial knowledge graph and the updates in terms of added and deleted triples, computes an estimation of the summary of the updated knowledge graph. We then assess whether or not this approach provides a descriptive summary of the updated knowledge graph by comparing the output to the actual summary of the later version.

This assignment is structured as follows. Section 2 provides a detailed overview over our approach, while Section 3 describes our tentative experimental results. Lastly, Section 4 concludes the assignment.

2 Approach

In this section, we describe our approach to apply an update to a knowledge graph. Due to space restrictions, we leave the definition of quotient summaries purposely concise and refer the interested reader to [2] for a more detailed description hereof. Given that a knowledge graph G can be described as a set of RDF triples $(s, p, o) \in (U \cup B) \times U \times (U \cup B \cup L)$ where U is the set of all URIs, B the set of ball blank nodes, and L the set of all literals, a *quotient summary* is defined as follows.

Definition 1 (Quotient Summary). *A quotient summary of a knowledge graph G, S_G, is a 4-tuple $S_G = \langle N, T, S, E \rangle$ where:*

- *N is a* super node *mapping from a URI to a super-node, i.e., $N : U \mapsto U$.*
- *T is a set of triples such that $\forall t = (s, p, o) \in G$, $t \in T$ if $s, o \notin N$ or $(N(s), p, N(o))$ otherwise.*
- *S is a node support mapping that, given a super node n, returns the number of nodes in T mapped to n (called the node support), i.e., $S : U \mapsto \mathbb{Z}^+$. If n does not exist in T, $S(n) = n$.*
- *E is an edge support mapping that, given two super nodes n_1 and n_2 and a predicate p, returns the number of times the edge between n_1 and n_2 appears with p (called the edge support), i.e., $E : U \times U \times U \mapsto \mathbb{Z}^+$. If $(n_1, p, n_2) \notin T$, then $E(n_1, n_2, p) = 0$.*

Given two super nodes n_1 and n_2, a predicate p, and a quotient summary S_G, we say that $(n_1, n_2, p) \in S_G.E$ if $S_G.E(n_1, n_2, p) > 0$. Likewise, we say that $n_1 \in S_G.S$ if $S_G.S(n_1) > 0$. With the definition of a quotient summary, we define an update as follows.

Definition 2 (Update). *An* update *over a knowledge graph G, U_G is a tuple $U_G = \langle A, D \rangle$ where A and D are sets of triples that describe additions and deletions respectively, and $\forall t_1 \in A, t_2 \in D$, it is the case that $t_1, t_2 \in G$.*

Given the definitions of quotient summaries and updates, Algorithm 1 shows the algorithm that applies an update to a quotient summary.

Algorithm 1 Apply an update to a quotient summary

Input: Quotient summary S_G; Update U_G
Output: Quotient summary S'_G

1: **function** *applyUpdate*(S_G,U_G)
2: $S'_G \leftarrow S_G$
3: **for all** $t = (s, p, o) \in U_G.A$ **do**
4: $n_1 \leftarrow getSuperNode(S_G.N, s)$
5: $n_2 \leftarrow getSuperNode(S_G.N, o)$
6: $S'_G.T \leftarrow S'_G.T \cup \{(n_1, p, n_2)\}$
7: $S'_G.S(n_1) \leftarrow S_G.S(n_1) + 1$
8: $S'_G.S(n_2) \leftarrow S_G.S(n_2) + 1$
9: $S'_G.E(n_1, n_2, p) \leftarrow S_G.E(n_1, n_2, p) + 1$
10: **for all** $t = (s, p, o) \in U_G.D$ **do**
11: $n_1 \leftarrow getSuperNode(S_G.N, s)$
12: $n_2 \leftarrow getSuperNode(S_G.N, o)$
13: $S'_G.T \leftarrow S'_G.T \setminus \{(n_1, p, n_2)\}$
14: $S'_G.S(n_1) \leftarrow S_G.S(n_1) - 1$
15: $S'_G.S(n_2) \leftarrow S_G.S(n_2) - 1$
16: $S'_G.E(n_1, n_2, p) \leftarrow S_G.E(n_1, n_2, p) - 1$
17: **return** S'_G

3 Results

We evaluate the quality of the summaries generated by our algorithm by comparing them to precomputed RDF Quotient summaries. We compute the similarity between the summaries as the measure of quality[3].

3.1 Experimental setup

We evaluated our method on two revision of the DBpedia [3] ontology. More specifically we use the ontology from the 2015-10 revision and the 2016-10 revision. The 2015-10 ontology is comprised of 30K triples while the 2016-10 is comprised of 31K triples. We compute the RDF Quotient summaries of both revisions and we manually generate the delta between the two revisions as a set of added and deleted triples. Between the two revisions, 4640 triples were added while 3908 triples were deleted. Thereafter we use our algorithm on the first summary and the delta as input, which generate an updated summary. This updated summary in then compared to the previously computed summary for the second revision of the ontology. The way we calculate the similarity between those summaries is explained on section 3.2.

3.2 Computing the similarity

The process of evaluating the similarity of several generated summaries is a non-trivial problem. The allocation of super-nodes in summaries will not be

[3] The code and additional files can be found on our GitHub at https://github.com/Chraebe/SummarizationCourse.

consistent between two different inputs. The main challenge when comparing two summaries is that two logically equivalent triples in the summaries could be interpreted as being different due to the super-node being numbered differently. Meaning that any naive algorithm attempting to access the similarity between the two summaries will very likely overestimate the differences between the two summaries. Ideally we would be able to to exactly identify super-nodes between the two summaries that are equivalent, unfortunately this is beyond the scope of this project.

We can compute a lower bound estimate of the similarity between the two summaries by computing the set difference. This metric is very likely to underestimate significantly the similarity between the two summaries due to the fact that some node are logically equivalent while being named differently. By this metric, the summaries are shown to be at least 30% similar, with the possibility that this number is a significant underestimation.

4 Conclusions

In this paper we investigated the problem of updating RDF graph summaries in the context of evolving datasets. We proposed a novel algorithm which generate a new RDF Quotient summary from an existing summary and the changeset to the new revision of the graph. We evaluated the quality of the generated summaries by analysing its similarity to a reference RDF Quotient summary. This evaluation proved to be difficult because super-nodes can be different between two summaries while being logically equivalent. The lack of a good solution to find these equivalent nodes makes an accurate comparison of the summaries difficult. Future work would involve the design of an algorithm capable of comparing two summaries while being able to detect equivalent triples in order to properly take them into account. Similarly, future work should evaluate the viability of this approach on large RDF graphs and assess the scalability of the algorithm.

References

1. Aebeloe, C., Montoya, G., Hose, K.: ColChain: Collaborative linked data networks. In: WWW 2021. pp. 1385–1396 (2021)
2. Guzewicz, P., Manolescu, I.: Quotient RDF summaries based on type hierarchies. In: ICDE Workshops. pp. 66–71 (2018)
3. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web **6**(2), 167–195 (2015)
4. Pelgrin, O., Galárraga, L., Hose, K.: Towards fully-fledged archiving for rdf datasets. Semantic Web (Preprint), 1–24 (2021)
5. Troullinou, G., Kondylakis, H., Daskalaki, E., Plexousakis, D.: RDF digest: Efficient summarization of RDF/S kbs. In: ESWC. pp. 119–134 (2015)