gOWL: A Fast Ontology-Mediated Query Answering

Chenhong Meng[1,5], Xiaowang Zhang[1,5,*], Guohui Xiao[3], Zhiyong Feng[2,5], and Guilin Qi[4]

[1] School of Computer Science and Technology, Tianjin University, Tianjin 300350, China,
[2] School of Computer Software,Tianjin University, Tianjin 300350, China
[3] Faculty of Computer Science, Free University of Bozen-Bolzano, Bolzano I-39100, Italy
[4] School of Computer Science and Engineering, Southeast University, Nanjing 211189, China
[5] Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin 300350, China
[*] Corresponding author.

Abstract. This poster shows an ontology-mediated query answering system (gOWL) based on pure materialization to avoid query rewriting online. gOWL shows that answering queries over a partial materialization of the chase is complete for a large fragment of practical queries with bounded depth. From a system engineering perspective, the materialization approach allows us to design a modular architecture to integrate off-the-shelf efficient SPARQL query engines. The poster will be based on DBpedia and UOBM datasets. We will compare the performance of gOWL with PAGOdA, Ontop, and Pellet (with speedup up to three orders of magnitude).

1 Introduction

OMQ (Ontology-mediated Query) is a core reasoning task in many applications [1]. OMQs have been studies intensively on lightweight ontology languages, that have canonical model properties. There are basically two approaches for query answering, namely, materialization-based and query rewriting-based. Materialization-based approaches normally compute and materialize the chase first and then execute the queries over the materialization, such as RDFox [7] and PAGOdA [8]. However, materialization is often infeasible when the chase contains infinitely existentially entailed elements. Instead, query writing-based approaches avoid materializing the chase but rewrite input queries by compiling the consequence of the reasoning into the query, such as QuOnto [2], Clipper [4] and Ontop [3]. Query rewriting comes at the cost of query rewriting and query evaluation at runtime, and the possibility of missing optimization opportunity at the data level.

In this poster, we adopt a pure materialization-based approach which allows us to design a modular architecture to integrate off-the-shelf efficient SPARQL query engines. We implement the proposed approach in a prototype gOWL, and build an OMQ systems gOWL-3X by employing RDF-3X. The preliminary encouraging experiments on DBpedia and UOBM show that gOWL outperforms PAGOdA, Ontop, and Pellet.

2 Framework of gOWL

The approach proposed in this Section has been implemented in the gOWL system [6]. The framework of gOWL contains three modules, namely, *Query Processor*, *Normalized Model Constructor*, and *Query Execution* shown in the following figure.

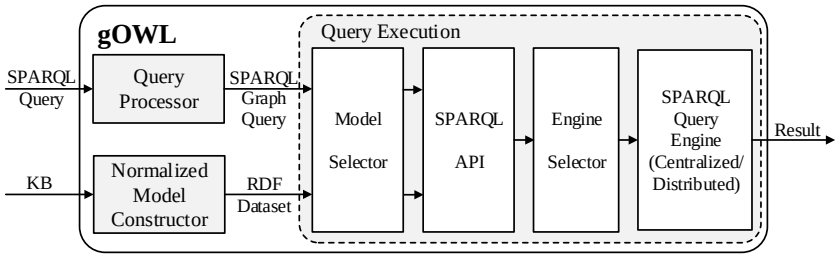

Fig. 1: The framework of gOWL

Query Processor Query Processor is a module to compute the depth of a query by applying a depth-first-search method. We use $\mathrm{dep}(q)$ to denote the depth of q, which represents the length of maximal certain path in q. The nodes in the path must be starting of a quantified-free variable of q and all other nodes are quantified variables.

Normalized Model Constructor In Normalized Model Constructor, we compute n-step universal models $\mathcal{U}_{\mathcal{K}}^{n}$ (i.e., $\mathcal{U}_{\mathcal{K}}^{0} \ldots \mathcal{U}_{\mathcal{K}}^{n}$) for given a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ a natural number n. n represents the count of expanded steps from ABox. For example, $\mathcal{U}_{\mathcal{K}}^{0}$ means to expand 0 step from ABox, which equivalent to itself; $\mathcal{U}_{\mathcal{K}}^{1}$ means to expand 1 step from $\mathcal{U}_{\mathcal{K}}^{0}$ (i.e., ABox), the expansion condition is $(\mathcal{T}, \mathcal{U}_{\mathcal{K}}^{0}) \models \exists\ R_1(a)$ but $R_1(a,b) \notin \mathcal{U}_{\mathcal{K}}^{0}$, for all $b \in \mathrm{Ind}(\mathcal{U}_{\mathcal{K}}^{0})$; $\mathcal{U}_{\mathcal{K}}^{n}$ means to expand 1 step from $\mathcal{U}_{\mathcal{K}}^{(n-1)}$ (i.e., n-step from ABox), the expansion condition is $(\mathcal{T}, \mathcal{U}_{\mathcal{K}}^{(n-1)}) \models \exists\ R_1(a)$ but $R_1(a,b) \notin \mathcal{U}_{\mathcal{K}}^{(n-1)}$, for all $b \in \mathrm{Ind}(\mathcal{U}_{\mathcal{K}}^{(n-1)})$.

Intuitively speaking, n-step universal models of a KB are hierarchical extensions of the ABox via the TBox. In fact, based on the statistical analysis of practical SPARQL queries, over 96% of queries contain at most 7 triple patterns (i.e.,7 triples in a SPARQL query) [5]. Therefore, we only consider n no more than 7 in this poster.

Query Execution The module of Query Execution contains four parts, namely Model Selector, SPARQL API, Engine Selector, and SPARQL Query Engine. Model Selector is used to select a suitable $\mathcal{U}_{\mathcal{K}}^{i}$ for a given query that $i = dep$ can be satisfied. Through SPARQL API, the query and dataset are passed on Engine Selector which utilizes the information of them and the characteristics of each engine to recommend the suitable one. Finally we employ SPARQL Query Engine to return solutions.

[6] `https://github.com/liulovemeng/gOWL.git`

3 Experiments and evaluations

The experiments are carried out on a machine running Linux, which has 4 CPUs with 6 cores and 64GB memory. RDF-3X is used as the underlying SPARQL query engines. We utilized UOBM and DBpedia data as a standard of evaluation.

We evaluate on a dataset (around 12 million triples) of DBpedia ontology in [8] and $\mathrm{U}^7_{\mathcal{K}}$ is computed in 48 hours. The experimental results of 10 queries ($\mathbf{Q}_1 \sim \mathbf{Q}_{10}$) over three engines (i.e., gOWL-3X, PAGOdA, Pellet) are shown in the Figure 2. In a same way, the evaluate on UOBM [8] over three engines (i.e., gOWL-3X, PAGOdA, Ontop) are shown in the Figure 3 and Figure 4 (Note that we only list experimental results in 24 hours and the ordinate represents the total online time of query answering).

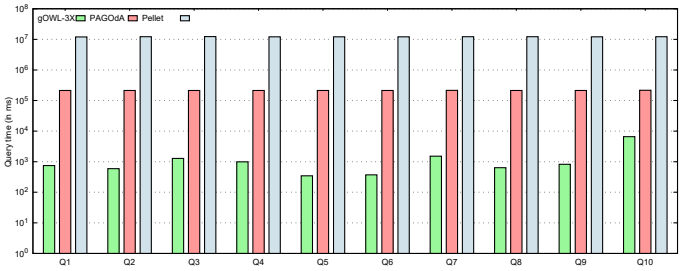

Fig. 2: Evaluations on DBpedia

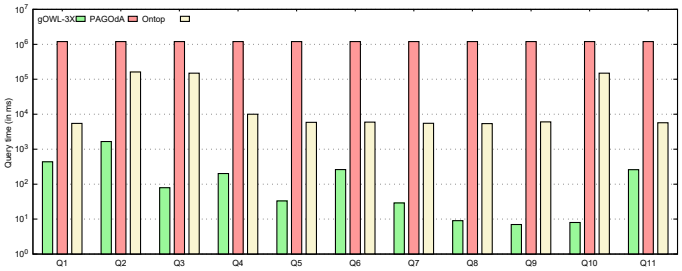

Fig. 3: Evaluations on UOBM100

We find that gOWL-3X significantly improve the performance of all 21 queries comparing to PAGOdA, Ontop and Pellet which represent materialization-based approach, query rewriting-based approach and traditional reasoning machine, respectively. Since Ontop doesn't have mapping information about DBpedia, it cannot handle DBpdiea dataset. From the figures we can see that gOWL-3X is several orders of magnitude higher than other methods, because it has the advantage of changing data processing to offline before the query coming. In addition, its performance remains good with data size increases (UOBM100 $\sim$ UOBM1000). Under the same conditions, PAGOdA couldn't handle large-scale dataset effectively because its query execution level is dependent on RDFox, which finishes processing the data online, so the memory requirements are much stronger.

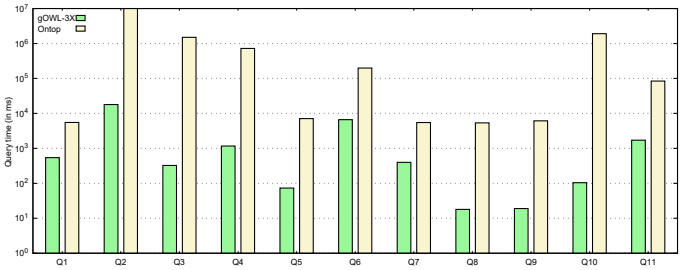

Fig. 4: Evaluations on UOBM1000

4 Conclusions

In this poster, we have presented the gOWL system for ontology-mediated query answering. The approach take advantage of high-performance of off-on-shelf SPARQL query engines for supporting large-scale ontology query answering in an efficient and simple way. Based on DBpedia and UOBM datasets, gOWL outperforms existing engines significantly.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (61502336), the National Key R&D Program of China (2016YFB1000603), the Key Technology R&D Program of Tianjin (16YFZCGX00210), and the Seed Foundation of Tianjin University (2018XZC-0016).

References

1. Bienvenu, M. (2016). Ontology-mediated query answering: Harnessing knowledge to get more from data. In *Proc. of IJCAI'16*, pp.4058–4061.
2. Botoeva, E., Calvanese, D., Santarelli, V., Fabio Savo, D., Solimando, A. and Xiao, G. (2016). Beyond OWL 2 QL in OBDA: Rewritings and approximations. In *Proc. of AAAI'16*, pp. 921–928.
3. Calvanese, D., Cogrel, B., Komla-Ebri, S., Kontchakov, R., Lanti, D., Rezk, M., Rodriguez-Muro, M., and Xiao, G. (2016). Ontop: Answering SPARQL queries over relational databases. *Semantic Web J.*, 8(3):471–487.
4. Eiter, T., Ortiz, M., Simkus, M., Tran, T.K., and Xiao, G. (2012). Query rewriting for Horn-SHIQ plus rules. In *Proc. of AAAI'12*, pp.726–733.
5. Han, X., Feng, Z., Zhang, X., Wang, X., Rao, G., and Jiang, S. (2016). On the statistical analysis of practical SPARQL queries. In *Proc. of WebDB'11*, article 2, ACM.
6. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., and Zakharyaschev, M. (2010). The combined approach to query answering in DL-Lite. In *Proc. of KR'10*. AAAI Press.
7. Motik, B., Nenov, Y., Piro, R., Horrocks, I., and Olteanu, D. (2014). Parallel materialisation of Datalog programs in centralised, main-memory RDF systems. In *Proc. of AAAI'14*, pp.129–137.
8. Zhou, Y., Grau, B., Nenov, Y., Kaminski, M., and Horrocks, I. (2015). PAGOdA: Pay-as-you-go ontology query answering using a datalog reasoner. *J. Artif. Intell. Res.*, 54(1):309–367.