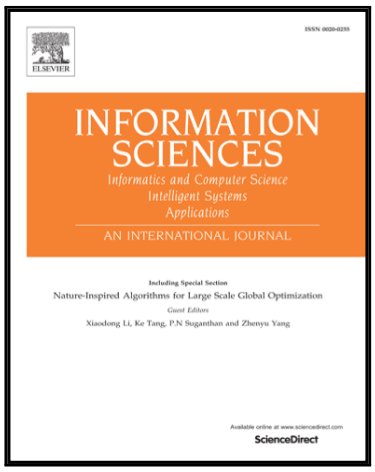

Natural Language Aggregate Query over RDF Data

Xin Hu , Depeng Dang , Yingting Yao , Luting Ye

PII: S0020-0255(16)31061-1
DOI: 10.1016/j.ins.2018.04.042
Reference: INS 13585

To appear in: *Information Sciences*

Received date: 27 September 2016
Revised date: 2 February 2018
Accepted date: 10 April 2018

Please cite this article as: Xin Hu , Depeng Dang , Yingting Yao , Luting Ye , Natural Language Aggregate Query over RDF Data, *Information Sciences* (2018), doi: 10.1016/j.ins.2018.04.042

Natural Language Aggregate Query over RDF Data

Xin Hu, Depeng Dang*, Yingting Yao, Luting Ye
College of Information Science and Technology, Beijing Normal University, Beijing 100875, China
*Corresponding author. Tel: +86 13121915369 E-mail address: ddepeng@bnu.edu.cn

ABSTRACT:

Natural language question/answering over RDF (Resource Description Framework) data has received widespread attention. Although several studies can address a small number of aggregate queries, these studies have many restrictions (e.g., interactive information, controlled questions or query templates). Thus far, there has been no natural language querying mechanism that can process general aggregate queries over RDF data. Therefore, we propose a framework called NLAQ (Natural Language Aggregate Query). First, we propose a novel algorithm to automatically understand a user's query intention, which primarily contains semantic relations and aggregations. Second, to build a better bridge between the query intention and RDF data, we propose an extended paraphrase dictionary *ED* to obtain more candidate mappings for semantic relations, and we introduce a predicate-type adjacent set *PT* to filter out inappropriate candidate mapping combinations in semantic relations and basic graph patterns. Third, we design a suitable translation plan for each aggregate category and effectively distinguish whether an aggregate item is numeric, which will greatly affect the aggregate result. Finally, we conduct extensive experiments over real datasets (QALD benchmark and DBpedia). The experimental results demonstrate that our solution is effective.

Keywords: RDF, question answering, natural language, aggregate query

1. Introduction

As increasing amounts of data become available on the web, academics and industry researchers must invest much more in bold strategies that can achieve natural language searching and answering [11]. RDF (Resource Description Framework) has been widely used as a W3C standard to describe data in the Semantic Web. Thus, natural language question/answering (Q/A) over RDF data has received widespread attention [48,3,13,41]. Although these methods are easy to use and can produce interesting results, they do not accommodate even simple aggregate queries, such as "*How many books by Kerouac were published by Viking Press?*"

Few works can address a small number of aggregate queries over RDF data [40,18,8,15], and users cannot access RDF data conveniently. Some of these works constructed an interactive interface [18,8], which requires users to fill out or choose aggregate items and aggregate categories. The input of Squall2sparql is a controlled English question [15], and users must specify the precise entities and predicates (denoted by URIs) in the question. TBSL [40] is a template-based approach that does not require users to do something extra, but the query templates in TBSL are fixed and must be constructed by analyzing a huge set of candidate queries. In conclusion, these methods answer aggregate queries over RDF data with too many restrictions and can only address a small number of aggregate queries, primarily because identifying and transforming aggregates are very difficult issues.

Additionally, two stages must be improved in RDF Q/A systems: *query understanding* and *mapping*. In the first stage, existing studies [15,40,48,3,12,41] concerning the identification of semantic relations completely depend upon the verb phrase in the query and paraphrase dictionary *D*, which records the semantic equivalence between verb phrases and predicates. The essential idea is to find two associated arguments of *rel* in the query according to linguistic rules, in which *rel* is also a verb phrase in *D*. Then, the verb phrase *rel*, together with two associated arguments, forms a semantic relation <*arg1*, *rel*, *arg2*>. However, there is a major disadvantage in this method. For Query1, "*How many books by*[1] *Kerouac were published by Viking Press?",* the verb phrase "*published*" is most likely to be found in *D*, whereas the non-verb phrase "*by*[1]" is not. Therefore, existing studies can identify the triple <*Kerouac, published, Viking Press*> and overlook the triple <*books, by, Kerouac*>.

In the second stage, existing studies [15,40,48,3,12,41] have not been able to obtain more candidate mappings for semantic relations and effectively filter out inappropriate mappings when the mappings have the same (or approximately the same) similarity score. Their essential idea is to strictly map the verb phrase *rel* and arguments *arg1*/*arg2* to the candidate predicate and entity/type, respectively; then, some sets of candidate mappings with high similarity scores are selected. On the one hand, strict mapping can improve the accuracy of mapping for a query that has no ambiguity. However, natural language has a wide range of ambiguity, and strict mapping will reduce the number of candidate mappings of triples and make most queries unanswerable (see the example in section 5.2.1). On the other hand, after mapping, existing studies depend upon similarity scores alone to select candidate mappings and