Big RDF Data Cleaning

Nan Tang
Qatar Computing Research Institute
Doha, Qatar
ntang@qf.org.qa

***Abstract*—Without a shadow of a doubt, data cleaning has played an important part in the history of data management and data analytics. Possessing high quality data has been proven to be crucial for businesses to do data driven decision making, especially within the information age and the era of big data. Resource Description Framework (RDF) is a standard model for data interchange on the semantic web. However, it is known that RDF data is dirty, since many of them are automatically extracted from the web. In this paper, we will first revisit data quality problems appeared in RDF data. Although many efforts have been put to clean RDF data, unfortunately, most of them are based on laborious manual evaluation. We will also describe possible solutions that shed lights on (semi-)automatically cleaning (big) RDF data.**

I. Introduction

The process of cleaning data is known to be hard, not only because of the dirtiness of real-life data – up to 30% of an organization's data could be dirty [2], but also because it is very expensive – it costs the US economy $3 trillion+ per year [1]. Despite the rich theoretical and practical contributions in all aspects of data cleaning [7], the journey of data cleaning is deemed to be extraordinary and to live long for many reasons: the burst of data in volume, the emergence of data formats such as RDF data, the varieties of errors in data, the x-factor of user involvement and above all, the pursuit of high quality data for all businesses.

Although RDF is a popular and important data format [11], [15], so far, however, there has been little discussion about how to automatically clean RDF data. RDF is normally dirty since most of them are automatically extracted from known sources, *e.g.,* YAGO [13] (Yet Another Great Ontology) extracts Wikipedia and other sources, NELL (Never-Ending Language Learner) [4] reads the web. It is reported [19] that YAGO has a good precision at 95%, since YAGO is based on a clean logical model with a decidable consistency. However, the estimated precision of NELL is only 74% [4], since NELL is more ambitious in that it tries to reads and learns the whole web, where the data itself being read could be wrong.

Despite the importance of high quality RDF data, cleaning RDF data is a hard problem, and reports from both academia [13] and industry [6] state that most of RDF cleaning tasks were done by laborious manual evaluation.

On the other hand, even if the RDF data at hand is considered to be correct, problems still remain when trying to use them effectively. One main issue is about aligning instances and relations in the same ontology, or across different ontologies. The former happens because that in general, a large ontology is contributed by different persons who will design various definitions for instances and relations, or is extracted and learned from different sources. The latter is more natural since they come from heterogeneous sources. Although work has been done to align instances and relations [16], [18], it remains and will last long to be a challenging topic.

Organization. The rest of the paper is organized as follows.

1) We will revisit data quality problems about RDF data management in Section II.
2) We identify some promising directions of cleaning RDF data in Section III.

II. RDF Data Quality Problem Revisited

Before discussing promising solutions, let us first identify data quality problems in RDF data management and their effect in emerging applications.

(1) Typos. Typos (*a.k.a.* misspellings) are normally introduced by humans. However, they can also by automatically introduced during inaccurate information extraction or segmentation. They are long-standing data cleaning problems for decades. Since RDF data is normally crafted from existing relational tables, or extracted from the web, this data quality issue about typos naturally carries over to RDF data.

Some tools for checking RDF typos have been developed[1], given some RDF triples and the website where these triples came from.

(2) Instance normalization/transformation. Often times, the same instance will be presented by different syntactic forms in an RDF source. As mentioned about the trouble with DBpedia[2], if you were trying to search U.S., you may end up with finding many objects such as "United States"@en, http://dbpedia.org/resource/United_States, or "US"@en.

In industry, ETL (extract, transform, load) rules are widely used for data normalization/transformation. They typically use dictionary lookup to transform data. For example, they can normalize all "United States"@en, http://dbpedia.org/resource/United_States, or "US"@en to "USA". Maintaining an accurate dictionary for the dynamic data on the web is a costly job.

(3) Missing types or literal values. Missing types are types of certain literal values that are absent. Missing literal values are values of certain properties that are missing, although they are

[1] http://graphite.ecs.soton.ac.uk/checker/
[2] http://blog.databaseanimals.com/the-trouble-with-dbpedia

needed, such as the literal values about state of a country are desired but are missing. Both are very important in answering user queries to identify correct results.

(4) Heterogeneous ontologies. In the literature, knowledge bases in RDF format have been widely used to understand web table semantics [22]. However, one main problem raised is about the ambiguity coming from the growing number of independently designed ontologies. For example, Italy could be the type economy, country, or even a club. Such heterogeneous ontologies will hinder the users to effectively understand and reason RDF data.

There has been some work to align heterogeneous ontologies [16], [18]. However, with the growing number of heterogeneous ontologies, aligning them will remain a challenging problem for a long term.

(5) Outdated values. Many values are frequently changed, *e.g.,* position, salary or age. When extracting values from the web, many information might have been outdated such as the age of Barack Obama.

When multiple instances of the same entity are given, algorithms using currency constraints [8] are already in place. Applying such techniques in RDF data needs a further investigation.

(6) Violations of integrity constraints. Dependency (*a.k.a.* integrity constraints) based theory provides a strong foundation to validate RDF data. For example, cardinality constraints will limit the number of occurrences of a certain type *e.g.,* spouse. Data that violates a certain constraint is considered to be incorrect. However, as observed in [10], standard database-style cardinality constraints cannot be modelled in neither RDFS nor OWL. Hence, how to carry over the rich study of constraint based data cleaning in relational database to RDF data deserves a full treatment (see [20] for a survey).

III. Some Possible Directions

Having recognized the problems mentioned in Section II, there are some promising steps that researchers have taken or could take towards addressing some aspects of the above problems.

(1) Constraints based RDF cleaning. Identify functional dependency violations and compute repairs on RDF data have been studied in [10]. In general, constraint based repairing is the data cleaning problem introduced in [3]: repairing is to find another database that is consistent and minimally differs from the original database. They compute a consistent database by using different cost functions for value updates and various heuristics to guide data repairing.

Constraints expressed by RDFS and OWL are different from integrity constraints studied in relational data. The first challenge is how to modify existing profiling algorithm [12] for discovering constraints over RDF data. The following issue is how to design effective algorithms to compute a *consistent* RDF data with the minimum cost.

(2) Master data. Master data (*a.k.a.* reference data) is a single repository of high-quality data that provides various applications with a synchronized, consistent view of its core business entities. An enterprise nowadays typically maintains master data. Master data management systems are being developed by IBM, SAP, Microsoft and Oracle. In fact, master data has been used to clean relational data using editing rules [9], fixing rules [21] or Sherlock rules [14]. New matching rules across RDF data and relational master data (or trusted knowledge bases) need to be defined, such that similar techniques can be applied to clean RDF data.

(3) Interactive RDF Cleaning. It is known that heuristic based data cleaning solutions might introduce data errors [9]. In order to ensure that a repair is reliable, users have been involved as first-class citizens in the process of data repairing [9], [17]. How to leave experts in the loop of RDF cleaning is an interesting topic.

Since many RDF data is extracted from the web, experts may not have enough capacity to clean them. Recently, crowdsourcing has been proven to be a viable and cost-effective alternative solution than data experts. Effectively involving the crowd requires dealing with traditional crowdsourcing issues such as forming easy-to-answer questions for the new data cleaning tasks and optimizing the order of issuing questions to reduce monetary cost.

(4) Big RDF data cleaning. In order to deal with big RDF data, a natural idea is to leverage existing database engines or distributed systems.

Let us explain how our commodity data cleaning system NADEEF [5] works for big RDF data cleaning. NADEEF is an extensible, generic and easy-to-deploy date cleaning system that provides simple programming APIs, which isolates users from the low level optimizations to achieve efficiency and scalability.

Consider an RDF dataset containing students, professors, and universities, where each student is enrolled in one university and has one professor as advisor. The top-left side of Fig. 1 shows this RDF dataset as set of triples (RDF input) and Fig. 2 shows the graph representation of such an RDF dataset. Let us assume that there cannot exist two graduate students in two different universities and have the same professor as advisor. According to this rule, there are two violations in this RDF dataset: (Paul, John) and (Paul, Sally).

Figure 1 depicts how NADEEF works to clean this RDF dataset. It starts with the Scope operator where it removes unwanted RDF predicates on attributes Subject and Object and passes only those triples with $advised\_by$ and $study\_in$ predicates to the next operator. After that, we apply the first Block to group triples based on student name (*i.e.,* Paul) followed by Iterate operator to join the triples in each group. The output of this Iterate is passed to the second Block and Iterate operators to group incoming triples based on advisor name (*i.e.,* William). The Detect operator generates violations based on incoming triples with different university names. Finally, the Fix operator suggests fixing the university names in incoming violations.

In fact, NADEEF has different back-ends to support real

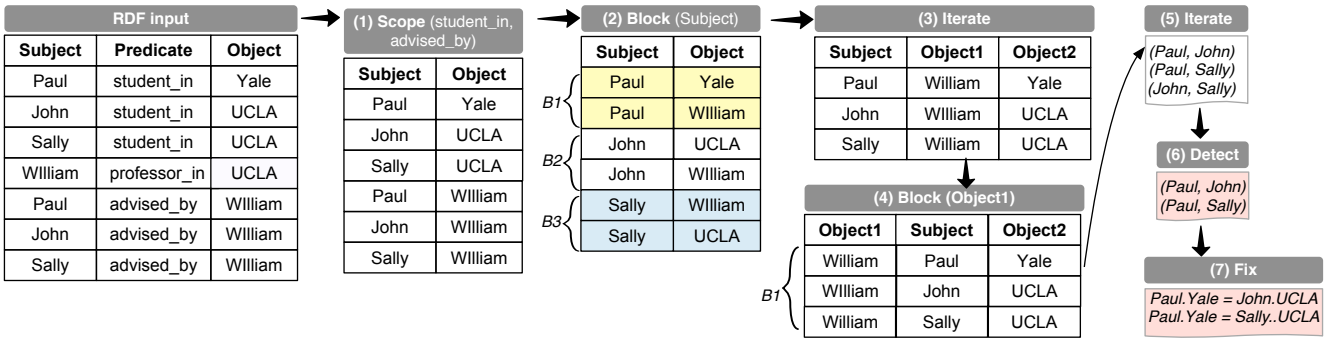

Figure 1. Logical operators execution for the RDF rule example.

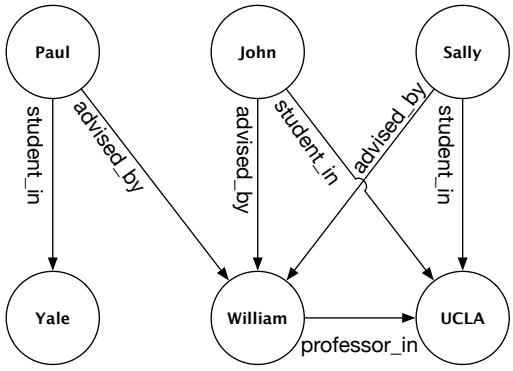

Figure 2. Example of an RDF graph.

execution of the above workflow, ranging from databases such as and PostgreSQL to distributed systems such as Spark.

IV. Conclusion

This article developed a proposal that

1) Revisits data quality issues in RDF data management.
2) Proposes possible solutions to address data quality problem in (big) RDF data.

References

[1] Dirty data costs the U.S. economy $3 trillion+ per year. http://www.ringlead.com/dirty-data-costs-economy-3-trillion/.

[2] Firms full of dirty data. http://www.itpro.co.uk/ 609057/firms-full-of-dirty-data.

[3] M. Arenas, L. E. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *Proceedings of the Eighteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 31 - June 2, 1999, Philadelphia, Pennsylvania, USA*, pages 68–79, 1999.

[4] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. H. Jr., and T. M. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, 2010.

[5] M. Dallachiesa, A. Ebaid, A. Eldawy, A. K. Elmagarmid, I. F. Ilyas, M. Ouzzani, and N. Tang. NADEEF: a commodity data cleaning system. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*, pages 541–552, 2013.

[6] O. Deshpande, D. S. Lamba, M. Tourn, S. Das, S. Subramaniam, A. Rajaraman, V. Harinarayan, and A. Doan. Building, maintaining, and using knowledge bases: a report from the trenches. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*, pages 1209–1220, 2013.

[7] W. Fan and F. Geerts. *Foundations of Data Quality Management*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2012.

[8] W. Fan, F. Geerts, N. Tang, and W. Yu. Inferring data currency and consistency for conflict resolution. In *29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013*, pages 470–481, 2013.

[9] W. Fan, J. Li, S. Ma, N. Tang, and W. Yu. Towards certain fixes with editing rules and master data. *VLDB J.*, 21(2):213–238, 2012.

[10] C. Fürber and M. Hepp. Using semantic web resources for data quality management. In *Knowledge Engineering and Management by the Masses - 17th International Conference, EKAW 2010, Lisbon, Portugal, October 11-15, 2010. Proceedings*, pages 211–225, 2010.

[11] F. Goasdou, Z. Kaoudi, I. Manolescu, J.-A. Quian-Ruiz, and S. Zampetakis. Cliquesquare: Flat plans for massively parallel rdf queries. In *IEEE 31th International Conference on Data Engineering, Seoul, ICDE 2015, Korea, April 13 - April 17, 2015*, 2015.

[12] A. Heise, J. Quiané-Ruiz, Z. Abedjan, A. Jentzsch, and F. Naumann. Scalable discovery of unique column combinations. *PVLDB*, 7(4):301–312, 2013.

[13] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum. YAGO2: A spatially and temporally enhanced knowledge base from wikipedia. *Artif. Intell.*, 194:28–61, 2013.

[14] M. Interlandi and N. Tang. Proof positive and negative in data cleaning. In *IEEE 31th International Conference on Data Engineering, Seoul, ICDE 2015, Korea, April 13 - April 17, 2015*, 2015.

[15] Z. Kaoudi and I. Manolescu. Rdf in the clouds: A survey. *VLDB J.*, 2015.

[16] S. Lacoste-Julien, K. Palla, A. Davies, G. Kasneci, T. Graepel, and Z. Ghahramani. Sigma: simple greedy matching for aligning large knowledge bases. In *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*, pages 572–580, 2013.

[17] V. Raman and J. M. Hellerstein. Potter's wheel: An interactive data cleaning system. In *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy*, pages 381–390, 2001.

[18] F. M. Suchanek, S. Abiteboul, and P. Senellart. PARIS: probabilistic alignment of relations, instances, and schema. *PVLDB*, 5(3):157–168, 2011.

[19] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 697–706, 2007.

[20] N. Tang. Big data cleaning. In *Web Technologies and Applications - 16th Asia-Pacific Web Conference, APWeb 2014, Changsha, China, September 5-7, 2014. Proceedings*, pages 13–24, 2014.

[21] J. Wang and N. Tang. Towards dependable data repairing with fixing rules. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, pages 457–468, 2014.

[22] J. Wang, H. Wang, Z. Wang, and K. Q. Zhu. Understanding tables on the web. In *Conceptual Modeling - 31st International Conference ER 2012, Florence, Italy, October 15-18, 2012. Proceedings*, pages 141–155, 2012.