Data Cleansing Consolidation with PatchR

Magnus Knuth and Harald Sack

Hasso-Plattner-Institut für Softwaresystemtechnik GmbH,
Prof.-Dr.-Helmert-Str. 2–3, 14482 Potsdam, Germany
`{magnus.knuth,harald.sack}@hpi.uni-potsdam.de`
`http://www.hpi.uni-potsdam.de`

1 Introduction

The Linking Open Data (LOD) initiative is turning large resources of publicly available structured data from various domains into interlinked RDF(S) facts to constitute the so-called "Web of Data". But, this Web of Data is by no means a perfect world of consistent and valid facts. Linked Data has multiple dimensions of shortcomings ranging from simple syntactical errors over logical inconsistencies to complex semantic errors and wrong facts. Multiple efforts target data quality assessment or aim to detect and to resolve such shortcomings in Linked Data datasets, such as crowdsourcing based, statistical, or heuristical approaches. These approaches rather address particular problems or datasets than to be generalizable for any kind of error. Moreover, results are published in various forms, which makes it hard to combine their results.

In this paper we propose the aggregation of heterogeneous Linked Data cleansing efforts by using the Patch Request ontology [1]. This allows to include less assured outcomes in order to reach a higher coverage.

2 Linked Data Cleansing Approaches

SDtype [2] deduces new entity types based on statistics about the usage of properties with entities of known type, e. g. for the *DBpedia Type Completion Service*[1]. Due to the statistical character of this approach, the results are somewhat vague and in order to achieve a high precision only the most probable type mappings are applied. That means the published dataset[2] still contains incorrect mappings while on the other hand valid mappings have been omitted due to their potential vagueness.

RDFUnit (a.k.a. Databugger) [3] allows to unit test RDF(S) datasets. Therefore, tests in form of SPARQL queries are derived from the set of schemas applied in the dataset according to predefined patterns, which detect failing resources. These tests are generally applicable and the introduction of restrictions to the schema leads to higher test coverage. Currently, these tests indicate erroneous

[1] `http://wifo5-21.informatik.uni-mannheim.de:8080/DBpediaTypeCompletionService/`
[2] `http://dbpedia.org/Downloads39#mapping-based-types-heuristic`

RDF(S) triples in the dataset but do not provide solutions. Though for some patterns equally generic solutions can easily be found. Likewise, *Inconsistency Checker* [4] detects logical inconsistencies in DBpedia using a reasoner based on an enriched ontology model with strict type constraints. A default solution proposal could be the deletion of triples involved in failed tests.

Crowdsourcing is a valuable (and high quality) mean to detect inconsistency of data to the real world. *TripleCheckMate* [5] have collected erroneous triples in DBpedia. Games with a purpose (GWAP), such as *WhoKnows?* [6], aim to identify errors and inconsistencies on various levels of semantic expressivity.

3 The Patch Request ontology

The PatchR ontology[3] [1] allows to describe patches, i. e. the removal or insertion of particular RDF(S) triples (or if necessary, wider spanning subgraphs) within a dataset in conjunction with provenance information of how this patch has been detected. Patch description may also include confidence values that express the self-determined reliability of this patch. This confidence must be expressed as a numerical value in the range of $(0, 1]$, whereas a high value means higher confidence and a value of 1 signifies absolute certainty.

Some of the referenced approaches, e. g. *SDtype*, deliver a confidence by default. For other approaches either a ranking could be determined depending on the origin of the patch or a manually provided default value can be applied. As for this experiment we have assumed the confidence of patches from the crowdsourcing approaches *WhoKnows?* and *TripleCheckMate* as relatively high (0.8), and for *RDFUnit* the confidence value depends on the applied pattern. Nevertheless, any agent may be wrong and can not be genuinely trusted. Therefore, each agent obtains a trust value, which might compile from the validation of previous statements.

Such descriptions will allow the aggregation and comparison of multiple efforts' results.

4 Examples

SDtype excludes results of low reliability in the provided dataset. If these results are fostered by other approaches they should likely enter the DBpedia dataset as well. As e. g., the assignment of type `dbo:Artist` to the resource `dbp:Maria_Callas` is omitted since it has received a relatively low score of ≈ 0.18:

```
:patch-1 a pat:Patch ;
  pat:advocate :SDtype ;
  pat:appliesTo <http://dbpedia.org> ;
  pat:status "active" ;
  pat:update [
    guo:target_subject dbp:Maria_Callas ;
```

[3] `http://purl.org/hpi/patchr`

```
    guo:insert [
      a dbo:Artist
  ] ]
  prov:wasGeneratedBy [
    prov:wasAssociatedWith :SDtype ;
    pat:confidence "0.17729138"^^xsd:double
  ] .
```

The *RDFUnit* INVFUNC pattern instantiated with the property dbo:keyPerson, and a player of *WhoKnows?* have identified the same triple to be incorrect in DBpedia. Both patch requests can be aggregated as follows:

```
:patch-2 a pat:Patch ;
  pat:advocate :RDFUnit/INVFUNC , :WhoKnows/Player-1 ;
  pat:appliesTo <http://dbpedia.org> ;
  pat:status "active" ;
  pat:update [
    guo:target_subject dbp:Vimeo ;
    guo:delete [
      dbo:keyPerson dbp:President
  ] ]
  prov:wasGeneratedBy [
    prov:wasAssociatedWith :RDFUnit/INVFUNC ;
    pat:confidence "0.9"^^xsd:double
  ] , [
    prov:wasAssociatedWith :WhoKnows/Player-1 ;
    pat:confidence "0.7"^^xsd:double
  ] .
```

4.1 Combining Confidence Values

In case multiple agents propose the same patch, the confidence grows that this change is valid and should be applied to the dataset. To combine multiple confidences ($c_{p|a}$ and $c_{p|b}$) for the same patch p, the following associative, commutative, uniformely continuous operation $\oplus$ can be applied:

$$\begin{aligned} c_{p|a,b} = c_{p|a} \oplus c_{p|b} &= 1 - ((1 - c_{p|a}) * (1 - c_{p|b})) \\ &= c_{p|a} + c_{p|b} - (c_{p|a} * c_{p|b}) \end{aligned} \tag{1}$$

To achieve the reliability of a patch, we multiply the confidence with the trust in the respective agent. To combine reliabilities we apply again the operation $\oplus$.

$$\begin{aligned} r_{p|a,b} = r_{p|a} \oplus r_{p|b} &= r_{p|a} + r_{p|b} - (r_{p|a} * r_{p|b}) \\ &= t_a c_{p|a} + t_b c_{p|b} - (t_a c_{p|a} * t_b c_{p|b}) \end{aligned} \tag{2}$$

The combined confidence value of :patch-2 proposed by *RDFUnit* (R) and *WhoKnows?* (W) from the listing above calculates to 0.97. Assuming a trust of 0.75 in both agents the reliability of the patch is ≈ 0.85.

$$\begin{aligned} c_{p2|R,W} &= c_{p2|R} + c_{p2|W} - (c_{p2|R} * c_{p2|W}) \\ &= 0.9 + 0.7 - (0.9 * 0.7) = \underline{0.97} \end{aligned} \tag{3}$$

$$\begin{aligned} r_{p2|R,W} &= t_R c_{p2|R} + t_W c_{p2|W} - (t_R c_{p2|R} * t_W c_{p2|W}) \\ &= 0.75 \cdot 0.9 + 0.75 \cdot 0.7 - (0.75 \cdot 0.9 * 0.75 \cdot 0.7) = \underline{0.845625} \end{aligned} \tag{4}$$

5 Conclusion and Outlook

We described a simple way to collect and aggregate patch descriptions from multiple heterogeneous agents. Aggregating patch descriptions allows to increase the coverage of incorrect and missing RDF(S) triples, whereas the accuracy might decrease.

Further processing of these patches could be the direct application of patches to datasets in order to achieve higher quality. Since the actual sources of errors can be diverse, we plan to identify classes of errors. Therefore, a larger collection of patches can be helpful. We currently generate the patch descriptions for the named approaches using the PatchR API[4]. Many datasets are derived from external sources, such as DBpedia is derived from Wikipedia, it might be necessary or appropriate to fix errors in the original version of the data. On the other hand, the extraction process itself might be imperfect in a way that it produces incorrect triples. In case of DBpedia this includes the extraction framework as well as the mappings that are used to create the triples.

References

1. Knuth, M., Hercher, J., Sack, H.: Collaboratively patching linked data. In: Proceedings of 2nd International Workshop on Usage Analysis and the Web of Data (USEWOD 2012), co-located with the 21st International World Wide Web Conference 2012 (WWW 2012), Lyon, France (April 2012)
2. Paulheim, H., Bizer, C.: Type inference on noisy RDF data. In: International Semantic Web Conference. (2013)
3. Kontokostas, D., Westphal, P., Auer, S., Hellmann, S., Lehmann, J., Cornelissen, R., Zaveri, A.J.: Test-driven evaluation of linked data quality. In: Proceedings of the 23rd international conference on World Wide Web. (2014)
4. Töpper, G., Knuth, M., Sack, H.: DBpedia ontology enrichment for inconsistency detection. In: Proceedings of the 8th International Conference on Semantic Systems (I-SEMANTICS 2012), September 5-7, 2012 Graz, Austria. ACM International Conference Proceedings, Graz, Austria, ACM (September 2012) 33–40
5. Kontokostas, D., Zaveri, A., Auer, S., Lehmann, J.: Triplecheckmate: A tool for crowdsourcing the quality assessment of linked data. In: Proceedings of the 4th Conference on Knowledge Engineering and Semantic Web. (2013)
6. Waitelonis, J., Ludwig, N., Knuth, M., Sack, H.: WhoKnows? – evaluating linked data heuristics with a quiz that cleans up DBpedia. International Journal of Interactive Technology and Smart Education (ITSE) **8**(3) (2011) 236–248

[4] `https://github.com/mgns/patchr`