Towards Comprehensive Noise Detection in Automatically-Created Knowledge Graphs

Nandana Mihindukulasooriya*[1], Oktie Hassanzadeh[2], Sarthak Dash[2], and Alfio Gliozzo[2]

[1] Ontology Engineering Group, Universidad Politécnica de Madrid, Madrid, Spain
nmihindu@fi.upm.es
[2] IBM Research, USA
{hassanzadeh,sdash,gliozzo}@us.ibm.com

Abstract. Knowledge Graphs (KGs) play a key role in many artificial intelligence applications. Large KGs are often constructed through a noisy automatic knowledge extraction process. Noise detection is, therefore, an important task for having high-quality KGs. We argue that the current noise detection approaches only focus on a specific type of noise (i.e., fact checking) whereas knowledge extraction methods result in more than one type of noise. To this end, we propose a classification of noise found in automatically-constructed KGs, and an approach for noise detection focused on specific types of noise.

1 Introduction

Knowledge Graphs (KGs) are key components of most modern artificial intelligence and cognitive applications. These KGs are largely constructed from textual corpora using automatic information extraction techniques. Such techniques introduce noise in the KGs and noise detection and removal become essential steps.

Most of the current noise detection in knowledge graphs is done with human supervision. For instance, large KGs such as YAGO2, DBpedia, or Wikidata use human contributors to verify the correctness of a given fact. This is a time-consuming task and requires a lot of human effort for large KGs. Although crowd-sourcing could be a solution for public general-domain KGs, it may not be a viable solution for enterprise KGs, due to both privacy issues as well as the need to rapidly create KGs from a large number of distinct corpora and in specific domains that require deep expertise. Thus, there is a need for automatic techniques for detecting noise in KGs.

Current automatic noise detection techniques are focused on factual correctness of triples. In this paper, we discuss the need for different types of noise in KGs and how to detect those specific types of noise.

* This research is partially supported by the 4V project (TIN2013-46238-C4-2-R) and the BES-2014-068449 grant. Work done while the author was an intern at IBM.

2 Related Work

Knowledge Graph Refinement [1], more concretely, noise detection is an important concern of industrial KGs. Noise detection is discussed mainly under two tasks in the literature: *Fact Checking* (or *Factual correctness*) and *Triple Classification*. In fact checking [2], the correctness of a statement is verified by finding external sources that confirm it. DeFacto [2] looks up the Web to find statements expressed in natural language in web pages while Liu *et al.* [3] do so by finding consensus in other external knowledge graphs. Triple classification [4] is a binary classification task to predict correctness of facts using graph embeddings. Given two entities, e_1 and e_2 and a relation R, a function $g(e_1,R,e_2)$ is defined in a way that a triple is true only if its value is above a given threshold [4]. In both these tasks, the main focus is on differentiating factual true triples in a single step. We argue that this can be divided into several sub-tasks by analyzing different types of noise. Our hypothesis is that by defining methods for detecting specific types of noise, we could improve the overall KG noise detection results.

3 Types of Noise

Based on the analysis of the output of a commercial information extraction framework, which parses text corpora and produces triples, we defined the categories in Fig. 1. Out of these, we identify *Factual True* triples as the most relevant and *Inconsistent*, *Generic*, *Factual False* categories as noise.

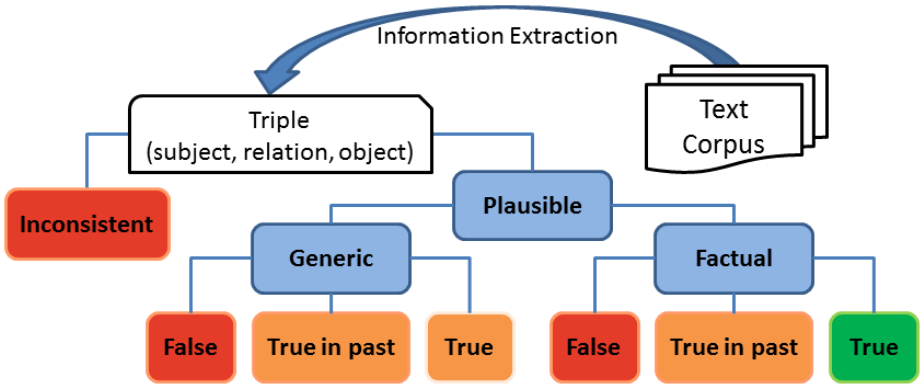

Fig. 1: Types of Noise

Inconsistent triples contradict the domain model they represent. As such, these triples are completely implausible and meaningless. For instance, (*Barack Obama*, *siblingOf*, *White House*), is not plausible. If *siblingOf* relation is specified formally to have a range of *Person* and if *Person* and *Building* are disjoint, that leads to a logical inconsistency. Nevertheless, those granular axioms are not

always present for such reasoning and data profiling can be used to identify common patterns in data that can provide heuristics of inconsistencies.

Generic triples do not mention specific entities and have less information value. For example, (*family*, *residesIn*, *New York*). Nevertheless, if both entities are generic such triples can provide schema-level information, for example, (*family*, *residesIn*, *city*). Finally, *factually false* triples are the ones that contain incorrect information. For example, (*Boston*, *capitalOf*, *USA*) is factually false.

4 Noise Detection Workflow

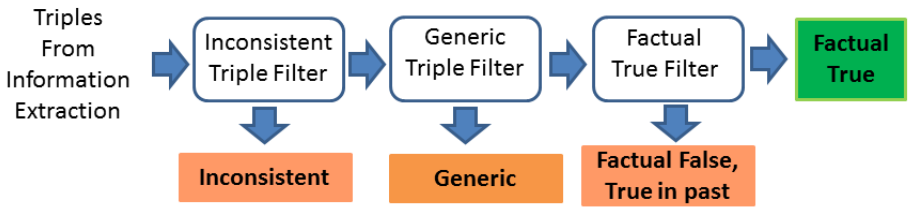

Fig. 2: Noise Detection Workflow

We propose to filter inconsistent and generic triples prior to fact checking as illustrated in the Fig. 2. This section describes approaches for detecting inconsistent and generic triples using the knowledge in external KGs.

Inconsistent triple checking is performed by mapping both entities as well as relations to external KGs and considering both ontological axioms that define formal conceptulizations (e.g., domain and range of properties or disjoint types) and common patterns in data. A relation mapping is one-to-one if both relations in information extraction and properties in KG have same granularity, for example, *siblingOf* to *dbo:sibling*. Otherwise, the mapping is conditional based on the domain class, for example, *partOfMany* maps to *dbo:country* in *dbo:City* class and *dbo:album* in *dbo:Single* class.

Generic triple detection is performed using part-of-speech tagging using NLP tools. We use the intuition that when the subject or the object is not a proper noun, it typically refers to a generic entity rather than a specific one. If either the subject or the object is generic, we label the triple generic. For determining factual correctness, we use a similar approach to fact checking by looking for evidences that confirm a given triple in an external KG using entity disambiguation and relation mapping described in Algorithm 1.

Fact checking is performed by finding evidences confirming a given triple in external knowledge graphs similar to the approaches in Section 2. We conducted preliminary experiments to evaluate the algorithms using 2,342 manually-labelled triples. Each triple was labelled with its type (i.e., *Inconsistent*, *Generic*, or *Factual*) and its truth value (i.e., *True*, *False*, or *True in the past*) by human annotators. The results are presented in Table 1.

Algorithm 1: Inconsistent triple detection

```
Data: A set of triples, T(subject,relation,object)
Result: The set of triples with labels, L(t ∈ T,l ∈ {consistent, inconsistent})
L ← ∅;
relationMap ← ∅;
distinctRelations ← getDistinctRelations(T);
foreach relation ∈ distinctRelations do
    K ← getTopKtriples(T, relation);
    propMap ← getKGProperty(K.subjects, K.objects);
    relationMap.put(relation, propMap);
end
foreach t ∈ T do
    sType ← getType(t.subject);
    oType ← getType(t.object);
    prop ← relationMap.get(t.relation).get(sType);
    if (ontoInconsistency(sType, prop, oType)) then
        L.add(t, inconsistent);
    else
        patternScore ← getScore(sType, prop, oType);
        if patternScore>threshold then
            L.add(t, consistent);
        else
            L.add(t, inconsistent);
end
```

Table 1: Experimental Results

Category	Configuration	Precision	Recall
Inconsistent triple detection		86.84%	62.26%
Generic triple detection	Standford NLP	85.65%	100%
	Open NLP	78.16%	100%
	Combined	98.25%	100%

5 Ongoing and Future Work

Our intuition is that the more-specific noise detection will improve the overall quality of KGs. One challenge for evaluating this hypothesis is the lack of a large gold standard with noise types. We plan to create one using crowd-sourcing.

Further, we also plan to test another alternative approach based on graph embeddings to demote triples that are inconsistent with other triples. For this, we learn representations for entities and relations by constructing functions that represent interactions between related entities.

References

1. Paulheim, H.: Knowledge Graph Refinement: A Survey of Approaches and Evaluation Methods. Semantic Web **8**(3) (2017) 489–508
2. Gerber, D., Esteves, D., Lehmann, J., Bühmann, L., Usbeck, R., Ngomo, A.C.N., Speck, R.: DeFacto - Temporal and Multilingual Deep Fact Validation. Web Semantics: Science, Services and Agents on the World Wide Web **35** (2015) 85–101
3. Liu, S., dAquin, M., Motta, E.: Measuring Accuracy of Triples in Knowledge Graphs. In: International Conference on Language, Data and Knowledge. (2017) 343–357
4. Socher, R., Chen, D., Manning, C.D., Ng, A.: ReasoningWith Neural Tensor Networks for Knowledge Base Completion. In: Advances in Neural Information Processing Systems 26. (2013) 926–934