Correcting Range Violation Errors in DBpedia

Piyawat Lertvittayakumjorn[1,2] (✉), Natthawut Kertkeidkachorn[2,3], and Ryutaro Ichise[2,3]

[1] Department of Computing, Imperial College London, London SW7 2AZ, UK
pl1515@imperial.ac.uk
[2] National Institute of Informatics, Tokyo 101-8430, Japan
{natthawut, ichise}@nii.ac.jp
[3] SOKENDAI (The Graduate University for Advanced Studies), Tokyo 101-8430, Japan

Abstract. *A range violation error* is a problem when an object of a knowledge graph triple does not have a type required by the range of the triple's predicate. This paper aims to correct these erroneous triples in DBpedia by finding correct objects with the required type to replace the incorrect objects. Our approach is based on graph analysis and keyword matching. It also exploits information from the incorrect objects because, despite their incorrectness, they contain useful clues to find the correct objects. Experimental results show that our proposed approach outperforms various baseline methods, including entity search (e.g., DBpedia Lookup) and knowledge graph completion (TransE and AMIE+).

Keywords: DBpedia, Linked Data, Data Quality, Error Correction, Range Violation Error, Knowledge Graph Refinement

1 Introduction

DBpedia is a large knowledge graph extracted from structured data in Wikipedia. Thanks to its large-scale size and the multi-disciplinary characteristic, DBpedia becomes a nucleus of the Linked Open Data (LOD) project to which numerous linked data resources connect. However, DBpedia is not free of errors for many reasons such as human errors and inconsistencies within Wikipedia, which is maintained by thousands of contributors. One major type of the errors is a problem when an object of a triple does not have a type required by the range of the triple's predicate [2]. We call this error type as *a range violation error* (RVE). Currently, 18.7% of DBpedia triples whose predicate is a mapping-based object property with a defined range are suffering from this kind of error. For example, the triple `<dbr`[4]`:Sedo, dbo`[5]`:locationCountry, dbr:Cologne>` in DBpedia is erroneous because the predicate `dbo:locationCountry` requires an object with the type `dbo:Country`, which `dbr:Cologne` is devoid of since Cologne is a city,

[4] dbr: http://dbpedia.org/resource/
[5] dbo: http://dbpedia.org/ontology/

not a country. This inconsistency could undermine the effectiveness of any applications using DBpedia. To correct this error, the object `dbr:Cologne` should be replaced by `dbr:Germany`, the country where Cologne is located.

Actually, there are several strategies to fix the RVE, depending on its root cause, such as adding the missing type to the object and refining the range indicated in the ontology. Nevertheless, in this work, we aim to solve the RVE automatically by finding a correct object to replace the incorrect object for each of the erroneous triples. This makes a significant impact on the research field because (1) it complements existing works on knowledge graph refinement which follow other strategies [4–6] and (2) based on our investigation, fixing by replacing objects is applicable to more than 62.8% of all RVEs in DBpedia.

Formally, our problem formulation is "**Given** an erroneous triple $t = \langle s, p, o \rangle$ in DBpedia where (1) p is a DBpedia object property with the range r_p and (2) o is an incorrect object which has at least one DBpedia ontology class (`dbo`) as its type but does not have r_p as its type. **Find** a semantically correct object o' that has the type r_p and best replaces the incorrect object o in $\langle s, p, o \rangle$."

2 Our Approach

It is clear that only objects with the type r_p could be the correct answer, so the complete search space of this problem (S_p) is a set of all entities with the type r_p. However, S_p is enormous for some properties p. So, our approach *(i)* constructs a reduced search space (S_t) that contains only the entities related to the erroneous triple t and then *(ii)* calculates scores of the entities in S_t.

2.1 Constructing a Reduced Search Space

S_t is constructed from the union of three portions – $S_{t,1}$, $S_{t,2}$, and $S_{t,3}$. First, we define that p' is a related property of p if and only if there exists at least one (x, y), $y \in S_p$, such that both $\langle x, p, y \rangle$ and $\langle x, p', y \rangle$ are in DBpedia. After that, we create $S_{t,1}$ storing all entities in S_p that are linked to s by at least one related property of p. In some cases, s may have p'-links to objects (entities or literals) that lack the type r_p, but they may give us some hints to the correct object. So, if the conditional probability $P(\langle x, p, y \rangle | \langle x, p', y \rangle, y \in S_p)$ is larger than a threshold τ (set at 0.9 in the experiment), we transform the objects into clue texts stored in C_t in addition to the label of the incorrect object o, which is always a clue text in any case. For each clue text $c \in C_t$, we tokenize c into a set of keywords K_c. Then we create $S_{t,2}$ storing all entities in S_p whose abstract contains at least one keyword from any $c \in C_t$. Last but not least, we create $S_{t,3}$ which collects all entities in S_p that connect immediately to the incorrect object o in any direction. Finally, we merge the three portions $(S_{t,1}, S_{t,2}, S_{t,3})$ to be S_t.

2.2 Calculating Scores

We invent two scoring methods to evaluate the likelihood that a candidate object $e \in S_t$ is the correct object of the triple t.

Method 1: Graph Method Intuitively, the correct object o' should be strongly related to o compared to other entities in S_t, and the more related two entities are, the more objects connect both entities. Therefore, our scoring function based on graph analysis is $g(e) = |A(o,e)| + b(e)$ where $e \in S_t$, $A(o,e)$ is a set of entities that have direct links to both o and e regardless of the links' direction, and $b(e) = 1$ if e links immediately to the incorrect object o; otherwise, $b(e) = 0$.

Method 2: Keyword Method We develop the keyword method to find $e \in S_t$ which the clue texts in C_t support. The scoring function of this method is

$$m(e) = \sum_{c \in C_t} \frac{|\{w \in K_c : w \text{ is in } abs(e) \wedge cap(w) \wedge w \text{ is in } prof(o)\}| + 1}{|K_c| + 1} + r(e).$$

The score of an entity e is calculated by aggregating scores of e with respect to each clue text $c \in C_t$. The score with respect to c reflects a proportion of keywords in c which are found in the *abstract* of e. To ensure that the keywords refer to a named-entity which is the replacement of o, we count only keywords w that begin with a capital letter ($cap(w)$) and are related to o (w is in the $profile$ of o). Additionally, the term $r(e)$ is a bonus point which will be 1 only if (1) $\exists p'[\langle s, p', e\rangle \wedge P(\langle x, p, y\rangle | \langle x, p', y\rangle, y \in S_p) > \tau]$ and (2) e is in the $profile$ of o. Satisfying these conditions is equivalent to the match of one clue text where the clue is an entity in the search space S_t.

Ranking Candidate Objects After calculating the scores (by using g or m), we can rank the candidate objects and select one with the highest score to replace o in t. In case the scores are equal, we have two more criteria to prioritize candidate objects – (1) the number of portions of S_t that cover the candidate, i.e., $|\{i : e \in S_{t,i}\}|$ and (2) the number of p in-links to the candidate.

3 Experiment

We tested our approach using four manually created datasets from four object properties – `dbo:locationCountry`, `dbo:formerTeam`, `dbo:employer`, and `dbo:birthPlace`. Each of the datasets contains one hundred erroneous triples $\langle s, p, o\rangle$ of a particular property together with their correct objects o'. We compare our approach to four baseline methods. Two are normally used for entity search (DBpedia lookup[6] and `dbo:wikiPageDisambiguates`) finding entities that have the required type r_p and could be the correct object o' from a given query o. The other two baseline methods are originally devised for knowledge graph completion (TransE [1] and AMIE+ [3]) finding the correct object given the subject s and the property p.

The results are presented in Table 1. M, @1, and @10 stand for three evaluation metrics – Mean reciprocal ranking (MRR), HITS@1, and HITS@10, respectively. All of them were calculated using the ranks of the correct objects provided

[6] http://wiki.dbpedia.org/projects/dbpedia-lookup

Table 1. The performance of our approach compared to other baseline methods

Method	locationCountry			formerTeam			employer			birthPlace		
	M	@1	@10	M	@1	@10	M	@1	@10	M	@1	@10
DBpedia lookup	0.02	1	3	0.06	6	6	0.07	6	10	0.04	3	5
wikiDisambiguates	0.04	3	4	0.04	3	4	0.04	3	7	0.11	8	18
TransE	0.19	11	37	0.02	1	1	0.00	0	3	0.24	20	35
AMIE+	0.42	40	43	0.06	6	6	0.00	0	0	0.01	1	1
Graph Method	**0.89**	**88**	**91**	0.37	25	61	**0.62**	**53**	**74**	0.44	24	**75**
Keyword Method	0.58	48	77	**0.60**	**49**	**78**	0.36	33	45	**0.48**	**36**	72

by each method. It is noticeable that both of our methods outperform baseline methods for all datasets. However, the graph method is more effective when an incorrect object corresponds to only one entity in S_p (as in locationCountry and employer datasets), because there are relatively many objects connecting this entity pair. Conversely, if an incorrect object is related to more than one entity in S_p, only information from the graph structure is not sufficient to find the correct object. The keyword method, in contrast, is more effective in such cases (e.g., formerTeam dataset) since it also exploits the information from s and p.

4 Conclusion

This paper aims to fix range violation errors in DBpedia automatically by finding correct objects to replace the incorrect objects. We developed an algorithm to construct a small search space of candidate objects and two scoring methods to evaluate the candidates. As exploiting information from all components of the erroneous triples, our proposed approach is effective to find the correct objects as demonstrated in the experiment. For future work, we plan to apply this idea to fix similar errors in other knowledge graphs such as Wikidata and NELL.

References

1. Bordes, A., Usunier, N., Garcia-Durán, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: 26th NIPS. pp. 2787–2795 (2013)
2. Dimou, A., Kontokostas, D., Freudenberg, M., Verborgh, R., Lehmann, J., Mannens, E., Hellmann, S., Van de Walle, R.: Assessing and refining mappings to rdf to improve dataset quality. In: 14th ISWC. pp. 133–149 (2015)
3. Galárraga, L., Teflioudi, C., Hose, K., Suchanek, F.M.: Fast rule mining in ontological knowledge bases with amie+. The VLDB Journal 24(6), 707–730 (2015)
4. Paulheim, H.: Data-driven joint debugging of the dbpedia mappings and ontology. In: 14th ESWC (2017)
5. Paulheim, H., Bizer, C.: Improving the quality of linked data using statistical distributions. IJSWIS 10(2), 63–86 (2014)
6. Tonon, A., Catasta, M., Demartini, G., Cudré-Mauroux, P.: Fixing the domain and range of properties in linked data by context disambiguation. In: LDOW (2015)