


Serving DBpedia with DOLCE – More than Just Adding a Cherry on Top

Heiko Paulheim¹ and Aldo Gangemi^{2,3}

¹ Research Group Data and Web Science, University of Mannheim,
Mannheim, Germany

`heiko@informatik.uni-mannheim.de`

² Université Paris 13 - Sorbonne Paris Cité - CNRS, Paris, France

³ STLab, ISTC-CNR, Rome, Italy

`aldo.gangemi@lipn.univ-paris13.fr`

Abstract. Large knowledge bases, such as DBpedia, are most often created heuristically due to scalability issues. In the building process, both random as well as systematic errors may occur. In this paper, we focus on finding *systematic* errors, or *anti-patterns*, in DBpedia. We show that by aligning the DBpedia ontology to the foundational ontology DOLCE-Zero, and by combining reasoning and clustering of the reasoning results, errors affecting millions of statements can be identified at a minimal workload for the knowledge base designer.

Keywords: Data quality · Formal ontologies · Foundational ontologies · Anti-pattern · DBpedia · DOLCE

1 Introduction

For the creation of large-scale knowledge bases, like DBpedia [18], there is often a trade off between coverage and precision. They cannot be curated manually, but only created by applying heuristic methods. Since those heuristics are most often not 100% exact, the resulting knowledge bases are not free of errors.

In this paper, we concentrate on *DBpedia*, which is a large-scale, cross domain knowledge base created from Wikipedia. To that end, Wikipedia infoboxes are mapped to a central ontology in a crowd-sourced process. Those mappings are then used to extract the DBpedia knowledge base from Wikipedia dumps. In the past years, DBpedia has become one of the central hubs of the Linked Open Data (LOD) cloud [27], with many applications using DBpedia for various purposes.

Due to the importance of DBpedia both as a linking hub in the LOD cloud, as well as a knowledge base for various applications, many works have been proposed in the recent past which target the improvement of the data in DBpedia. However, most of those approaches target at identifying *individual* errors, i.e., statements which are likely to be wrong. In contrast, in this paper, we aim at the identification of *systematic* errors, such as shortcomings of the heuristics used, or wrong mappings. Systematic errors are sets of individual errors following a similar pattern and having a common root cause (e.g., a wrong mapping).

Since DBpedia version 3.9, released in 2013, mappings of the DBpedia ontology to DOLCE-Zero [9,11], a subset of the modules of the formal ontology DOLCE, are included in the DBpedia ontology. We exploit those mappings to identify conflicting statements in DBpedia with the help of a reasoner, and use clustering to extract common patterns in the justifications. We find that in many cases, each cluster is related to a particular problem in the construction of the DBpedia knowledge base. While DOLCE has been used for improving several ontologies on the T-Box (i.e., terminological) level, this work is novel since it does not solely aim at improving the T-Box, but also the construction of the A-Box (assertional level) of a large-scale knowledge base.

The rest of this paper is structured as follows. In section 2, we review related works both w.r.t. debugging knowledge bases such as DBpedia, as well as the use of formal top level ontologies for improving such knowledge bases. We introduce our approach in section 3. An evaluation is carried out in section 4 by examining the results as well as quantifying the influence of DOLCE-Zero, and by analyzing the largest clusters identified and a sample from the long tail of non-clustered statements. We show that both views lead to the identification of a number of issues in DBpedia by inspecting only a very small fraction of selected statements. We conclude with a summary and an outlook on future work.

2 Related Work

In this paper, we target the identification of *systematic errors* in the construction of the large-scale knowledge base DBpedia. More specifically, we consider the identification of wrong relation assertions between two individuals.

There is a larger body of work which targets at finding errors in web knowledge bases such as DBpedia. The approaches vary both with respect to the methods employed as well as to the targeted type of assertions – i.e., identifying wrong type assertions, relational assertions, literals, etc.

Methods found in the literature range from statistical methods [24] and outlier detection [6,22,34] to using external methods, such as web search engines [17]. In addition, crowdsourcing [1] and games with a purpose [32] have been proposed as non-automatic means for identifying errors in knowledge bases.

In this paper, we propose the use of reasoning, in combination with further processing of the reasoning results by means of data mining. The DBpedia ontology – as many schemas used for providing Linked Open Data – is not very expressive, in particular with respect to the presence of disjointness axioms. Thus, there is a natural limitation for reasoning-based approaches. Hence, such approaches are often combined with ontology learning as a preprocessing step to enrich the ontology at hand [16,19,31]. In contrast to those approaches, we exploit the links to the foundational ontology DOLCE-Zero, and the high-level disjointness axioms defined therein.

This approach has been applied in the past, starting from the creation of the DOLCE foundational ontology in 2002 [10], and its use in the restructuring of WordNet [9,12]. Indeed, one of the main goals of upper level and foundational

ontologies, jointly with meaning negotiation among ontology designers, and harmonization of ontologies, is that of “cleaning” a schema or a knowledge base by inducing inferences (which can produce inconsistencies, or not) due to the axioms defined on the classes and properties that are used as alignment targets of the knowledge base schema. Examples include: [26], which describes the detection of thousands of incoherences in a large collection of medical ontologies; [7], which uses a foundational ontology to detect incoherences in anti-money-laundering rules, as well as in suspicious financial transactions; [20], which uses foundational axioms to integrate and cleanup alternative service ontologies; [8], which also describes the detection of incoherences emerging from the formalization of a collection of thesauri and classification schemes in the fishery domain.

To our knowledge, foundational ontologies have never been used to detect inconsistencies in very large knowledge bases at the scale of millions of individuals and facts, although using this approach was actually suggested pretty soon [15]. More recently, an attempt [29] has been made in using a lightweight (non-foundational) upper ontology (UMBEL) to populate DBpedia 3.7 with disjointness axioms, and derive inconsistencies. However, only 55,829 inconsistencies are detected, from 5 logical-level types, with many of them due to a multi-hierarchical categorization of certain buildings in that version of DBpedia. To our knowledge, the latter is also the only approach that tries to target the identification of *systematic* errors, instead of individual ones.

3 Approach

We pursue a multi-stage process, as shown in Fig. 1. First, a reasoner is used to list all property assignment statements that are inconsistent with the ontology, along with their explanations. Then, those statements are clustered in order to isolate patterns in the inconsistent statements. The patterns are then examined manually to assess the inconsistencies identified.

While DBpedia contains type, relational, and literal assertions, we concentrate on identifying problems with relational assertions. The type assertions in DBpedia are mostly correct, but incomplete [23], there exists a reasonable amount of noise in the relational assertions. For example, Weaver et al. [33] determine the fraction of wrong relational assertions in Wikipedia links to be 2.8%, a number that can also serve as a rough estimate for DBpedia.

3.1 The Graph

The graph considered is constituted by the DBpedia 2014 ontology, mapping-based types, and mapping-based properties datasets.¹ The alignment to DOLCE-Zero (see below) is included in the ontology dataset (T-Box: the alignments of classes and properties), and in the mapping-based types dataset (A-Box: the

¹ <http://wiki.dbpedia.org/Downloads2014>. The namespaces used are:
<http://dbpedia.org/resource/>, prefix=dbpedia,
<http://dbpedia.org/ontology/>, prefix=dbo

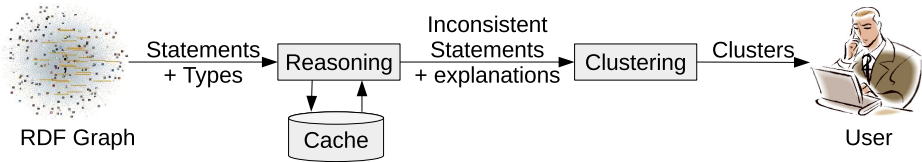


Fig. 1. The overall process. Statements from the DBpedia RDF graph are examined together with the subject’s and object’s types. The resulting inconsistent statements are clustered by similar explanations, and an expert user examines the clusters.

materialized types). The schema mapping has been defined by a DOLCE-Zero designer (one of the authors of this paper) at the time of the call for mappings (issued in 2014 by DBpedia maintainers); the mapping is relative to the DBpedia 3.9 ontology and dataset. The alignment has been created by carefully inspecting the T-Box axioms and a sample of the A-Box axioms for each class and property, in order to compare the wiki-based ontology development to the actual data models. The instance type materialization is based on the schema mapping.

For the 2014 release, the DBpedia maintainers have created the ontology dataset and the materialized instance mappings of DOLCE-Zero with reference to DBpedia 2014, with the mapping having been defined for the DBpedia 3.9 ontology. This results in a few issues. In particular, as new classes and properties have been added to the DBpedia 2014 version, some domain and range restriction axioms have changed, and some infobox mapping has changed as well, some of the alignments to DOLCE-Zero need to be revised, while some alignments are missing because of the new classes and properties. For this experiment, we have not revised the alignments, because the instance type materialization was still based on the 3.9 mapping in the DBpedia 2014 dataset.²

DOLCE-Zero consists of two OWL ontology modules derived from DOLCE [9] and the D&S [11] ontologies. The original design was made in the S5 modal logic [4] (DOLCE), and KIF [14] (D&S). DOLCE covers general distinctions concerning physical and social objects, events, abstractions, attributes, dimensional regions, as well as mereological (part), participation, inherence (attributive), and localization relations. In other words, DOLCE covers some of the core ontology design patterns that are typically assumed in the majority of conceptual schemata. D&S defines a vocabulary for roles, frames, concepts, and situations, which help representing many domains (e.g., biomedicine, law, business, organizations) that are often ambiguous in using words for expressing actual entities vs. concepts describing various collections of entities. In other words, D&S complements DOLCE with conceptual-level ontology design patterns.

The original DOLCE was hardly reusable on the Semantic Web, because of idiosyncratic terminological choices, and the strong expressivity of many of its axioms (n -ary relations, possible world and temporal indexing of relationships,

² Actually, our approach was capable of automatically discovering the places in which changes in the DBpedia ontology lead to invalid alignments (cf. Section 4), and led to fixes that will become part of the DBpedia 2015 version.

non-trivial first-order co-reference of variables). Therefore, the DOLCE designers decided soon to create a lighter version in OWL by relaxing n-ary relations, removing possible world and temporal indexing, and ignoring the most complex axioms. During the years, additional modules have been developed to cover e.g. WordNet’s top level, as well as to link the light versions of DOLCE and D&S. Eventually, two modules have emerged as mostly useful to work with LOD:

- an OWL module called DOLCE+DnS Ultralight (DUL),³ which contains a simplification of the original DOLCE axioms, with some additional concepts and relations, and the D&S vocabulary
- an upper level module, called DOLCE-Zero (D0)⁴ that simplifies some of the distinctions in DUL, which has been created to optimize the alignment of WordNet used by the Tipalo method for automatic typing of Wikipedia resources [13].

D0 is a small set of classes on top of DUL, which deal with ambiguity and completeness issues. Firstly, it introduces four “union classes” (`d0#Characteristic`, `d0#Eventuality`, `d0#Activity`, and `d0#Location`) that generalize some disjoint classes from DUL that are sometimes considered too “picky”, e.g. qualities vs. dimensional regions, events vs. situations, actions vs. tasks, space regions vs. physical locations. In practice, those distinctions are seldom represented in lightweight ontologies and natural language lexicons, and often originate debatable inconsistencies. Secondly, D0 introduces three top-level classes that have never been clarified and eventually accepted in DOLCE: `d0#CognitiveEntity`, `d0#System`, and `d0#Topic`. Those classes are needed in order to align existing ontologies. The combination of DBpedia Ontology and D0 is coherent in itself with one exception: in DBpedia, the class `dbo#Library` is a case of metonymy, since is a subclass of both Building and Organization. However, the alignments are respectively to `d0#PhysicalObject` and to `d0#SocialObject`, which are disjoint classes.

3.2 Identifying Conflicting Statements

For each relation $r(s, o)$ that holds between a subject s and an object o , we collect all direct types of the subject and the object, i.e., $T_s = \{T|T(s)\}$, $T_o = \{T|T(o)\}$. Those statements – i.e., the relation as well as the type assertions – are presented to a reasoner, together with the DBpedia and D0 ontologies. Typically, a statement is detected as conflicted if the domain or range assignment of r contains a class which is disjoint with a class in T_s or T_o , respectively. Fig. 2 illustrates such an inconsistency.

Once a reasoner detects such an inconsistency for a statement, it can also deliver an *explanation*, i.e., the set of axioms that, together, are inconsistent. In the example in Fig. 2, it comprises the original relation assertion, the type assertion for the object, the property’s range assertion, as well as all the subclass,

³ <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl>, prefix=dul

⁴ <http://www.ontologydesignpatterns.org/ont/dul/d0.owl>, prefix=d0

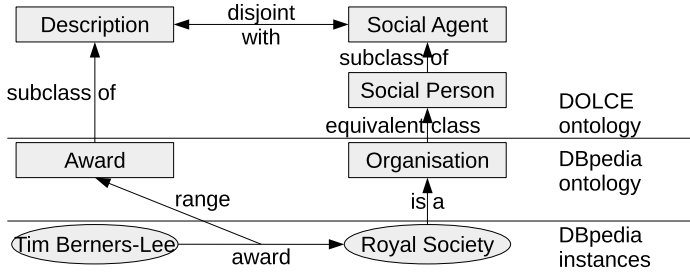


Fig. 2. Conflict detection example for the statement `dbpedia#Tim_Berners-Lee dbo#award dbpedia#Royal_Society`

equivalence, and disjointness assertions depicted in the figure. It is important to note that a detected inconsistency for a statement does not mean that the statement as such is wrong. It is rather the case that one of the axioms comprising the explanation (including the original statement) are wrong, which can also apply to a type assertion or any axiom in the DBpedia or D0 ontology.

Using a reasoner to check each and every statement in a large-scale dataset like DBpedia would lead to intractable runtimes. However, it is obvious that for two pairs of statements $r(s, o)$ and $r(s', o')$, the results of an inconsistency check are equivalent if $T_s = T_{s'}$ and $T_o = T_{o'}$. Hence, we can cache the reasoning results for a *characteristic signature* $\langle r, T_s, T_o \rangle$ and only invoke the reasoner for statements with a previously unseen characteristic signature.

3.3 Clustering Conflicts

The result of the previous step is a set of conflicting statements, where each statement has a set of axioms that lead to the conflict. In order to group similar conflicts, we use *clustering* and assign statements with similar axiom sets to the same cluster.

For determining the clusters, we represent each conflicting statement as a binary feature vector, where the features are the ontology axioms. As a distance function, we use the Manhattan distance between the feature vectors, i.e., the number of axioms by which the two explanations differ:

$$d(A_1, A_2) = |A_1 \cup A_2 - A_1 \cap A_2|, \tag{1}$$

where A_1 and A_2 are the axiom sets that lead to the inconsistency of two statements. Manhattan distance was chosen for simple computation and interpretability (i.e., a distance of n means that two explanations differ in n axioms).

As a clustering algorithm, we use DBSCAN [5]. That clustering algorithm forms clusters based on *density*, given two parameters ϵ and M . A cluster is formed around an instance if at least M instances are within a distance of ϵ of that instance. In our example, this means that clusters contain conflicting statements whose explanations do not differ by more than ϵ axioms. DBSCAN

was used since (a) it does not require specifying the number of clusters upfront, (b) its efficiency, and (c) its capacity of isolating noise.

Instances which are not assigned to any cluster are marked as *noise*. These are conflicting statements whose explanation is not similar to that of at least M other statements. Hence, choosing the parameter M gives us control on the minimum frequency of a particular conflict to regard it as a *systematic* error.

4 Evaluation

We have evaluated our approach on DBpedia 2014, using the mapping-based types and properties datasets. We have run the conflicting statement detection and clustering once with, once without the D0 ontology. For reasoning, we used the HermiT reasoner [28], for clustering, we used a slightly modified version of the DBSCAN implementation in RapidMiner^{5,6}.

4.1 Basic Results

Without D0, a total of 97,749 statements (0.65% of all statements) was found to be inconsistent, with 630 different axioms involved in the corresponding explanations. With D0, this number increases to 3,654,255 statements (24.36% of all statements), with 1,467 axioms involved in the corresponding explanations.

As discussed above, we use caching of reasoner results. In total, the reasoner had to examine only 34,554 out of 15,001,543 statements, i.e., 0.03%. Computing the consistency and explanation of a statement took 2.6 seconds on average, which totals to 25h (i.e., without caching, the whole consistency checking step would take more than one year). The clustering took around five minutes. All computations were performed on a standard laptop.

We ran DBSCAN with $\epsilon = 2$ and $\epsilon = 4$, and used 5,10,25,50, and 100 as values for M . The cluster sizes and number of noise instances are depicted in Table 1. Although the absolute number of clusters when using D0 is only four times larger, while the total number of detected inconsistent statements is 37 times larger. The reason is that some of the clusters found when using D0 are quite large, i.e., the inconsistencies found affect a large number of statements. In fact, the largest 16 clusters are clusters that use D0 axioms in their explanations.

The values in the table also demonstrate the value of clustering: even if inspecting only a few hundred clusters (each presented as an example statement

⁵ <http://www.rapidminer.com>

⁶ The modification was made in order to make DBSCAN incorporate instance weights when counting instances. This allows us to reduce the dataset to only one instance per conflicting set of axioms, and set the number of statements exposing that conflict as the instance weight. With this modification, the dataset size to be processed in RapidMiner could be reduced by a factor larger than 1,000. The modification is contained in the *Mannheim RapidMiner Toolbox Extension*, available at https://marketplace.rapidminer.com/UpdateServer/faces/product_details.xhtml?productId=rmx_maratool

Table 1. Number of clusters and noise instances with and without D0 for DBSCAN different parameter settings of ϵ and M . #C denotes the number of clusters, $\varnothing C$ denotes the average cluster size, and #N denotes the number of noise instances.

ϵ	M	without D0			with D0		
		#C	$\varnothing C$	#N	#C	$\varnothing C$	#N
2	5	218	447	355	915	3,992	1,835
4		182	536	264	745	4,903	1,390
2	10	180	540	614	681	5,361	3,392
4		151	644	474	565	6,463	2,574
2	25	129	747	1,406	457	7,981	6,746
4		108	894	1,160	380	9,602	5,454
2	50	98	972	2,529	338	10,779	10,958
4		84	1,139	2,075	288	12,658	8,709
2	100	68	1,370	4,623	254	14,321	16,797
4		62	1,519	3,570	219	16,624	13,537

and the corresponding explanation), a user can fix several million statements in DBpedia.

At a first manual inspection, we found out that the clusters with the larger ϵ value of 4 still looked rather coherent, i.e., the explanations are reasonably similar to group them together. Moreover, to keep the number of clusters tractable, we focus on the configuration with $\epsilon = 4$ and $M = 100$. Fig. 3 shows the distribution of clusters with this configuration.

4.2 Major Sources of Inconsistencies

Table 2 depicts the top 10 disjointness axioms, asserted classes, and asserted properties which occur in the inconsistencies identified by our approach. Again, the trend is confirmed that the absolute number of problems discovered when exploiting D0 is by several orders of magnitude larger than when relying only on the DBpedia ontology.

Furthermore, we can observe that some of the major problem sources, such as the modeling of species, career stations, and musical artists/bands (see below), are not captured without D0. This shows that in those areas, the formalization depth of the DBpedia ontology is rather low.

When using D0, seven out of the ten top 10 disjointness assertions are from the D0 ontology, while only three come from the DBpedia ontology. The fundamental distinction between social and physical objects is responsible for by far most of the inconsistencies detected.

It is further noteworthy that even for the disjointness axioms asserted in the DBpedia ontology, the number of conflicts detected when using D0 are higher in absolute numbers. For example, the disjointness between `dbo#Person` and `dbo#Event` is involved in over 10 times more inconsistencies when using D0. This is due to the fact that other assertions within D0, such as inverse properties, are also exploited by the reasoner.

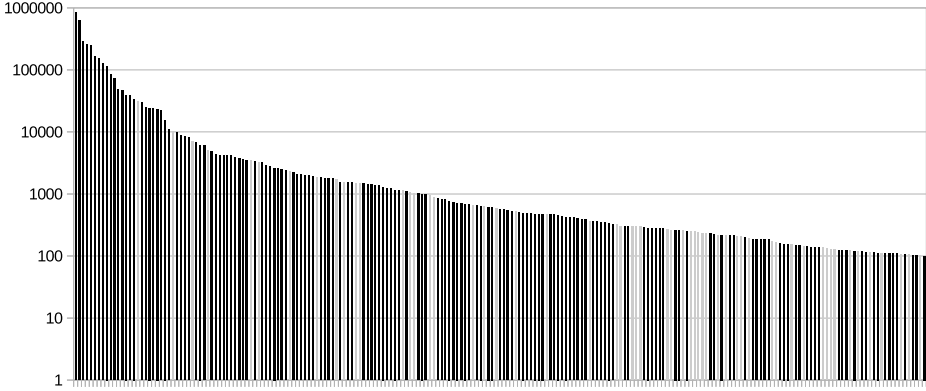


Fig. 3. Distribution of clusters and their sizes at $\epsilon = 4$ and $M = 100$ (note that the y axis uses a logarithmic scale). Black bars denote clusters of explanations using D0, grey bars represent clusters of explanations not using D0.

4.3 Evaluation of the Largest Clusters

The 40 largest clusters cover 3,497,068 inconsistent triples, corresponding to about 96% of all inconsistencies. 36 clusters use axioms from D0, i.e., only four would be found when considering the DBpedia ontology alone. We report here a classification of the 40 top clusters, according to the origin of the inconsistency, and the way(s) to fix it. Each category reveals an *anti-pattern*, i.e. a modeling solution used systematically, which produces unintended consequences.

Firstly, all the clusters contain an *inconsistency anti-pattern*: at least one domain or range restriction for a property is disjoint with at least one of the types declared for the subject or the object of an instance of that property. Formally, assuming the domain EA of explanation axioms, the domain $IA \subset EA$ of inconsistent A-Box axioms, the subsets EA_{a_1, \dots, a_n} , $EA_{a_i} \subset EA$ of explanation axioms for each inconsistent A-Box axiom $a_i \in IA$, with ρ being the object property from a_i , ϕ and ψ being domain and range classes for ρ , and χ , ξ being classes used as types of the individuals from a_i , the following description logic schema and data axiom templates are present in each cluster:

$$\rho \sqsubseteq \phi \times \psi \quad (2)$$

$$\phi \setminus \chi \sqcup \psi \setminus \xi \quad (3)$$

$$\rho(x, y) \wedge (\chi(x) \sqcup \xi(y)) \quad (4)$$

Secondly, specific anti-patterns emerge from the analysis of the systematic errors for the top 40 clusters:

Overcommitment (19). A conflict arises between the schema emerging from data, and the schema from the ontology. In these cases, the ontology provides a typically reasonable intuition on how a certain property should be used, e.g., `dbo#team` is designed as a relation between agents and sports teams,

Table 2. Top 10 disjointness axioms, classes, and properties involved in the inconsistencies detected.

With DOLCE-Zero		Without DOLCE-Zero	
dul#PhysicalObject , dul#SocialObject	3,363,689	dbo#Agent , dbo#Place	60,120
dul#Event , dul#Object	174,917	dbo#Person , dbo#TimePeriod	31,330
dbo#Event , dbo#Person	65,649	dbo#Event , dbo#Person	4,443
dbo#Agent , dbo#Place	62,250	dbo#MeanOfTransp. , dbo#Person	1,521
dul#InformationObject , dul#SocialAgent	51,022	dbo#Building , dbo#Person	245
dbo#Person , dbo#TimePeriod	31,323	dbo#Activity , dbo#Person	34
dul#Abstract , dul#Object	27,663	dbo#Person , dbo#Plant	31
dul#InformationObject , dul#Situation	26,693	dbo#Person , dbo#Tower	14
dul#Situation , dul#SocialAgent	20,594	dbo#Mountain , dbo#Person	9
dul#Concept , dul#SocialAgent	12,498	dbo#Person , dbo#UnitOfWork	1
dbo#Species	1,273,521	dbo#TimePeriod	31,330
dbo#Person	1,182,729	dbo#Agent	28,827
dbo#CareerStation	621,575	dbo#Place	17,555
dbo#MusicalArtist	131,810	Wikidata#Q532	9,539
dbo#Event	124,757	dbo#Event	4,295
dbo#Organisation	63,725	dbo#Organisation	1,832
dbo#Band	53,448	dbo#Astronaut	1,488
dbo#Agent	36,472	dbo#Company	1,366
dbo#TelevisionShow	31,607	dbo#Region	640
dbo#TimePeriod	31,330	dbo#Broadcaster	263
dbo#team	1,520,216	dbo#team	39,065
dbo#family	335,398	dbo#birthPlace	10,285
dbo#order	278,229	dbo#district	7,167
dbo#currentMember	260,325	dbo#owner	4,569
dbo#kingdom	244,427	dbo#leaderName	3,414
dbo#phylum	200,431	dbo#province	3,267
dbo#genus	175,478	dbo#deathPlace	2,272
dbo#associatedMusicalArtist	97,243	dbo#location	1,766
dbo#battle	96,112	dbo#recordedIn	1,728
dbo#class	44,434	dbo#locationCity	1,612

but is often used in ways that conflicts arise with the basic intuition. For example `dbo#team` can be used as a relation between events and teams participating in that event. Since `dul#Agent` is disjoint with `dul#Event`, inconsistencies are detected. The restriction on the domain of `dbo#team` results therefore as an *overcommitment*: the interpretation of a property universe is too specific compared to actual usage. Other examples of overcommitment include instances for the properties `dbo#associatedMusicalArtist`, `dbo#musicalBand`, `dbo#network`, `dbo#militaryBranch`, etc.

Metonymy (11). A conflict arises between two disjoint – but related – interpretations of a same concept. An example appears with `dbo#family`, used in triples expressing relations between species. `dbo#family` has been aligned to the property `dul#specializes`, holding for instances of `dul#Concept`, but the class `dbo#Species` has been aligned to `dul#Organism`, because species in DBpedia include species as well as individual exemplars of a species (for example, famous race horses), i.e. `dbo#Species` is used metonymically in data. Since `dul#Concept` is disjoint with `dul#Organism`, inconsistencies are detected. The metonymy anti-pattern is difficult to resolve, because it is due to ambiguities that seem widespread in human language. Metonymy seems related to human propensity for an economy of means: an interesting cognitive experiment [25] proves the communicative function of ambiguity

(cf. also [21] for a discussion). D0 tries to accommodate this “power of ambiguity” (cf. Section 3) to a certain extent, but relaxing all distinctions would prevent inconsistency checking in general. D0 relaxation has been limited to well known cases of metonymy, and the concept vs. organism metonymy for natural classifications was not considered.⁷

Misalignment (5). A conflict arises because a property (or a class) has been aligned to a wrong D0 entity, which causes inconsistencies in data classification. For example, the property `dbo#commander` has been aligned to `dul#coparticipatesWith`, but its usage in data is actually a case of `dul#hasParticipant`. `dul#coparticipatesWith` holds for instances of `dul#Object` only, but data include a 98% usage between `dul#Event` and `dul#Object`, and only 2% between `dul#Object`, therefore intended usage leans clearly towards the participation pattern.

Since the domain of `dul#coparticipatesWith` is `dul#Object`, and that of `dul#hasParticipant` is `dul#Event` (with `dul#Object owl#disjointWith dul#Event`), inconsistencies are detected. This anti-pattern suggests that DBpedia ontology choices proposed by the crowd, or by infobox reengineering, should also be made based on the actual resulting usage in data.

Version branching (3). A conflict arises between an alignment defined on a version, and a newer version. In these cases, the alignment provided for an older version, may become incoherent in case of a non-conservative change of the ontology in the newer version, e.g. `dbo#team` used to hold between career stations (professional situations of e.g. an athlete) and teams in DBpedia 3.9 ontology, but in DBpedia 2014 it holds between agents and sports teams. Since `dul#Situation` (aligning `dbo#CareerStation`), and `dul#Person` (aligning `dbo#Athlete`) are disjoint, inconsistencies are detected. This anti-pattern suggests that the design of new versions of DBpedia ontology should update the alignments to any change that has been made in the new version. Since this is an interaction problem, our clustering-based approach seems particularly appropriate to scale down the time needed to check all the interactions between the proposed changes in infobox reengineering, crowd modeling, and alignments.

Mistyping (1). A conflict arises between a type ϕ declared for some argument (subject or object) of an object property, and an expected type ψ expected for the universe of that property, when $\phi \setminus \psi$. This is typically due to systematic mistyping of individuals, and is not very frequent; an example is `dbpedia#Alfonso_XII_of_Spain dbo#birthPlace dbpedia#Madrid`, where `dbpedia#Madrid` is erroneously typed as `dbo#Agent`. Since `dbo#Agent` is disjoint with `dbo#Place`, which is expected in the range of `dbo#birthPlace`, inconsistencies are detected. Places being typed as agents occasionally occur in DBpedia, with `dbpedia#Korea` and `dbpedia#New Brunswick` being other prominent examples

⁷ D0 was from WordNet requirements [13], and a new axiom – such as `dbo#Species \sqsubseteq dul#Concept \sqcup dul#Organism` – would be justified only if there is a substantial amount of individual exemplars (actual organisms) typed as species in DBpedia.

Table 3. Amount of clusters and inconsistencies found for each anti-pattern in the 40 top clusters, and the expected fix to resolve them.

<i>anti-pattern</i>	<i>D0?</i>	<i>#clusters</i>	<i>#inco</i>	<i>%</i>	<i>fix type</i>
Overcommitment	16 yes, 3 no	19	587962	.168	Schema restriction axioms
Metonymy	yes	11	1277977	.365	Schema or data refactoring
Misalignment	yes	5	133663	.038	Alignment tuning
Version branching	yes	3	1477296	.422	Alignment tuning, workflow change
Mistyping	no	1	10285	.003	Entity typing
Wrong taxonomy	yes	1	9885	.003	Schema taxonomical axioms

Wrong taxonomy (1). A conflict arises between a property or class restriction ϕ , and another restriction from a property or class ψ , where $\phi \equiv \psi$. For example, the inconsistency in triples like `dbpedia#2002%E2%80%93Plymouth_Arghyle_F.C._season dbo#team dbpedia#Plymouth_Arghyle_F.C.` is due to the fact that the property `dbo#team` is `owl#equivalentProperty` `dbo#club`, but a specific domain is only stated for `dbo#club` (i.e. `dbo#Athlete`), while `dbo#team` is used in triples with subject of type `dbo#SportsSeason`, which is aligned to `dul#Situation` \sqcup `dul#TimeInterval`, which are both disjoint with `dbo#Athlete`.

Concerning possible fixes for the inconsistencies, solution patterns apply homogeneously for each of the anti-patterns. In particular, Overcommitment requires refactoring at the property restriction level, and need non-trivial design choices; Metonymy requires refactoring of both the ontology and the data, in order to partition the extension of a metonymical class or property; Misalignment requires tuning the alignments; Version branching requires tuning of the resulting misalignments, but in general a change in the ontology design workflow; Mistyping requires refactoring at the entity typing level; and finally, Wrong taxonomy needs to be solved at the schema level.

As a summary, Table 3 shows the amount of clusters and inconsistencies found for each anti-pattern, and the expected fix to resolve them.⁸

4.4 A Look at the Long Tail

In addition to the analysis of the top clusters, we also looked at the “long tail”, i.e., the infrequent sources of inconsistencies which DBSCAN assigns to the *Noise* cluster. Out of those statements, we drew a random sample of 100 statements, and evaluated them by hand. From those 100 statements, 64 were actually erroneous, 30 were false negatives (i.e., statements which are actually correct), and the remaining six were unclear or questionable. The main sources for the 64 errors are as follows:

Link in longer text (23). If the value of an infobox key-value pair is a complex expression containing several links, all of them are extracted

⁸ A detailed list of findings and proposed solutions is available at <http://dws.informatik.uni-mannheim.de/en/research/dbpedia+dolce/>

into a relational statement with the corresponding subject and predicate. However, this does not always make sense. One example is the statement `dbpedia#Cosmo_Cramer dbo#occupation dbpedia#Bagel`, which is extracted from the corresponding value *Bagel shop worker*, with *Bagel* linked to the corresponding Wikipedia page.

Wrong link (9). These are simply wrong links in the Wikipedia infobox. Following those links does not make sense to a human visitor of Wikipedia, since the link is completely wrong. One example is the statement `dbpedia#Stone_(band) associatedMusicArtist dbpedia#Dementia`, where the object denotes the disease *dementia*, not the artist of the same name.

Redirect (7). Redirects are resolved when building DBpedia. This can result in nonsensical statements. One typical example is the statement `dbpedia#Ben_Casey company dbpedia#Bing_Crosby`. Here, the original infobox value is `Bing_Crosby Productions`, which would be the correct object for the statement. However, the production company has no Wikipedia page on its own, but the object is a redirect to the person *Bing Crosby*, which leads to the error.

Link/anchor text mismatch (6). This is a special case of wrong links – here, the anchor link would actually suggest a different link target. In contrast to wrong links as stated above, following the link in Wikipedia could actually make sense to a Wikipedia user, since there is related information on the website. One example is the statement `dbpedia#Deutschland_sucht_den_Super-star judge dbpedia#MIA.`, where the judge of the TV show is the singer of the band *MIA.*, not the band as such. Nevertheless, the page about the band contains information about the singer.

Metonymy (4). These are a special case of link/anchor mismatches, where the linked resource and the object which was actually meant share their surface form. One example is the statement `dbpedia#Human_Nature_(band) genre dbpedia#Motown`, where the object denotes the record label *Motown*. The term *Motown*, however, is often used as a genre name for the artists signed by the label.

Anchor link (4). When building DBpedia, URI fragments are removed from the link. However, they are used in Wikipedia to point to certain sections on a page, which, in total, is on a different, but related topic. Thus, removing the fragment can sometimes lead to wrong statements in DBpedia. One example is the statement `dbpedia#Pierre_Langlais battle dbpedia#First_Army_(France)`, where the original link had a fragment pointing to the section on the page describing the actual battle.

Multiple Infoboxes (2). If a page contains multiple infoboxes, those can sometimes lead to wrong statements. One example observed is the Wikipedia page *Snooker World Rankings 1978/1979*, which contains infoboxes about individual players.

Unknown (9). In nine out of 64 cases, the authors of this paper could not find the reason for the error to come into existence.

It is noteworthy that in the long tail, i.e., the noise cluster, we could observe only those kinds of errors which are not specific to a certain class or property, but uniformly distributed across DBpedia classes and properties. This shows that the approach actually separates class-specific and class-independent problems, and assigns the latter to a separate cluster.

Furthermore, the analysis of the long tail has revealed four major sources of errors in DBpedia, which can be easily identified during the extraction phase and should obtain further attention in the future: links in longer texts, redirects, anchor links, and pages with multiple infoboxes. Given that the noise cluster contains 13,537 instances, these four sources account for an estimate of at least 4,800 wrong statements in total⁹. A possible treatment strategy could quarantine statements which have one of those characteristics, and treat them with special care, e.g., check them with a reasoner and/or statistical methods such as *SDValidate* [24].

5 Conclusion and Outlook

In this paper, we have shown how mappings to the foundational ontology DOLCE-Zero, which have been introduced in DBpedia from version 3.9 on, can be exploited for finding systematic errors in the construction of the DBpedia knowledge base. The combination of reasoning and clustering of the reasoner’s explanations helps minimizing the human expert’s workload: for the analysis of problems affecting millions of statements, no more than 140 statements (40 representative examples from the top clusters, plus 100 statements from the long tail) – i.e., 0.001% of DBpedia – have been inspected manually.

As a result of the analysis, some of the mappings used for creating DBpedia, as well as a number of assertions in the DBpedia ontology were directly changed. For problems that were not trivial to resolve, bug reports were filed. Furthermore, we have identified some areas where the construction of DBpedia requires additional attention, i.e., the extraction of relational statements from infobox values containing more text than just a plain link or literal, the handling of redirects, and the creation of relational statements from links containing an anchor fragment. Furthermore, our approach has revealed some non-optimal mappings between DBpedia and DOLCE-Zero, in particular where a) the DBpedia2014 ontology has used DBpedia 3.9 alignments, but the basic ontology had changed; and b) some properties are applied ambiguously, which should lead either to a change of the alignment, or of the DBpedia data or ontology.

So far, we have used a rather naive distance function for clustering the explanations, i.e., Manhattan distance on binary axiom occurrence vectors. More sophisticated similarity measures can be thought of, but are still subject to research [2].

The approach presented in this paper can be transferred to other datasets whose ontology is mapped to DOLCE-Zero. For the future, we are interested

⁹ Since not all problematic statements rooted in one of those four problems lead to an inconsistency, we expect the number to be even higher.

in testing the approach also on other knowledge bases, such as YAGO [30] or NELL [3], and compare the findings. From such experiments, we expect further insights in the prevalent challenges for building large-scale knowledge bases.

Acknowledgments. The work presented in this paper was supported by RapidMiner in the course of the RapidMiner Academia program.

References

1. Acosta, M., Zaveri, A., Simperl, E., Kontokostas, D., Auer, S., Lehmann, J.: Crowd-sourcing linked data quality assessment. In: Alani, H., et al. (eds.) ISWC 2013, Part II. LNCS, vol. 8219, pp. 260–276. Springer, Heidelberg (2013)
2. Bail, S., Parsia, B., Sattler, U.: Declutter your justifications: determining similarity between OWL explanations. In: 1st International Workshop on Debugging Ontologies and Ontology Mappings, pp. 13–24 (2012)
3. Carlson, A., Betteridge, J., Wang, R.C., Hruschka Jr., E.R., Mitchell, T.M.: Coupled semi-supervised learning for information extraction. In: International Conference on Web Search and Data Mining, pp. 101–110. ACM (2010)
4. Chellas, B.F.: Modal logic: an introduction, vol. 316. Cambridge Univ Press (1980)
5. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. *Kdd* **96**, 226–231 (1996)
6. Fleischhacker, D., Paulheim, H., Bryl, V., Völker, J., Bizer, C.: Detecting errors in numerical linked data using cross-checked outlier detection. In: Mika, P., et al. (eds.) ISWC 2014, Part I. LNCS, vol. 8796, pp. 357–372. Springer, Heidelberg (2014)
7. Gangemi, A., Pisanelli, D., Steve, G.: An ontological framework to represent norm dynamics. In: Proceedings of the 2001 Jurix Conference, Workshop on Legal Ontologies (2001)
8. Gangemi, A., Fisseha, F., Keizer, J., Lehmann, J., Liang, A., Pettman, I., Sini, M., Taconet, M.: A core ontology of fishery and its use in the fishery ontology service project. In: CEUR Proceedings, vol. 118 (2004)
9. Gangemi, A., Guarino, N., Masolo, C., Oltramari, A.: Sweetening WordNet with DOLCE. *AI Magazine (Fall)* (2003)
10. Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., Schneider, L.: Sweetening ontologies with DOLCE. In: Gómez-Pérez, A., Benjamins, V.R. (eds.) EKAW 2002. LNCS (LNAI), vol. 2473, pp. 166–181. Springer, Heidelberg (2002)
11. Gangemi, A., Mika, P.: Understanding the semantic web through descriptions and situations. In: Meersman, R., Schmidt, D.C. (eds.) CoopIS 2003, DOA 2003, and ODBASE 2003. LNCS, vol. 2888, pp. 689–706. Springer, Heidelberg (2003)
12. Gangemi, A., Navigli, R., Velardi, P.: The OntoWordNet project: extension and axiomatization of conceptual relations in WordNet. In: Meersman, R., Schmidt, D.C. (eds.) CoopIS 2003, DOA 2003, and ODBASE 2003. LNCS, vol. 2888, pp. 820–838. Springer, Heidelberg (2003)
13. Gangemi, A., Nuzzolese, A.G., Presutti, V., Draicchio, F., Musetti, A., Ciancarini, P.: Automatic typing of DBpedia entities. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 65–81. Springer, Heidelberg (2012)
14. Genesereth, M.R., Fikes, R.E.: Knowledge interchange format-version 3.0: reference manual. Tech. rep. (1992)

15. Jain, P., Hitzler, P., Yeh, P.Z., Verma, K., Sheth, A.P.: Linked data is merely more data. In: AAAI Spring Symposium: Linked Data Meets Artificial Intelligence, vol. 11 (2010)
16. Lehmann, J., Bühmann, L.: ORE - a tool for repairing and enriching knowledge bases. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part II. LNCS, vol. 6497, pp. 177–193. Springer, Heidelberg (2010)
17. Lehmann, J., Gerber, D., Morsey, M., Ngonga Ngomo, A.-C.: DeFacto - deep fact validation. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 312–327. Springer, Heidelberg (2012)
18. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web Journal* (2013)
19. Ma, Y., Gao, H., Wu, T., Qi, G.: Learning disjointness axioms with association rule mining and its application to inconsistency detection of linked data. In: Zhao, D., Du, J., Wang, H., Wang, P., Ji, D., Pan, J.Z. (eds.) CSWS 2014. CCIS, vol. 480, pp. 29–41. Springer, Heidelberg (2014)
20. Mika, P., Oberle, D., Gangemi, A., Sabou, M.: Foundations for service ontologies: aligning OWL-S to dolce. In: Staab, S., Patel-Schneider, P. (eds.) *Proceedings of the World Wide Web Conference (WWW2004)* (2004)
21. Nuzzolese, A.G., Gangemi, A., Presutti, V., Ciancarini, P.: Type inference through the analysis of wikipedia links. In: LDOW (2012)
22. Paulheim, H.: Identifying wrong links between datasets by multi-dimensional outlier detection. In: *International Workshop on Debugging Ontologies and Ontology Mappings* (2014)
23. Paulheim, H., Bizer, C.: Type inference on noisy RDF data. In: Alani, H., et al. (eds.) ISWC 2013, Part I. LNCS, vol. 8218, pp. 510–525. Springer, Heidelberg (2013)
24. Paulheim, H., Bizer, C.: Improving the quality of linked data using statistical distributions. *International Journal on Semantic Web and Information Systems (IJSWIS)* **10**(2), 63–86 (2014)
25. Piantadosi, S.T., Tily, H., Gibson, E.: The communicative function of ambiguity in language. *Cognition* **122**(3), 280–291 (2012)
26. Pisanelli, D., Gangemi, A., Steve, G.: An Ontological Analysis of the UMLS Metathesaurus. *J. of American Medical Informatics Association* **5** (1998)
27. Schmachtenberg, M., Bizer, C., Paulheim, H.: Adoption of the linked data best practices in different topical domains. In: Mika, P., et al. (eds.) ISWC 2014, Part I. LNCS, vol. 8796, pp. 245–260. Springer, Heidelberg (2014)
28. Shearer, R., Motik, B., Horrocks, I.: Hermit: A highly-efficient OWL reasoner. In: OWLED, vol. 432 (2008)
29. Sheng, Z., Wang, X., Shi, H., Feng, Z.: Checking and handling inconsistency of DBpedia. In: Wang, F.L., Lei, J., Gong, Z., Luo, X. (eds.) WISM 2012. LNCS, vol. 7529, pp. 480–488. Springer, Heidelberg (2012)
30. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: 16th International Conference on World Wide Web, pp. 697–706 (2007)
31. Töpper, G., Knuth, M., Sack, H.: DBpedia ontology enrichment for inconsistency detection. In: *Proceedings of the 8th International Conference on Semantic Systems*, pp. 33–40. ACM (2012)

32. Waitelonis, J., Ludwig, N., Knuth, M., Sack, H.: Whoknows? evaluating linked data heuristics with a quiz that cleans up dbpedia. *Interactive Technology and Smart Education* **8**(4), 236–248 (2011)
33. Weaver, G., Strickland, B., Crane, G.: Quantifying the accuracy of relational statements in wikipedia: a methodology. *JCDL* **6**, 358–358 (2006)
34. Wienand, D., Paulheim, H.: Detecting incorrect numerical data in DBpedia. In: Presutti, V., d’Amato, C., Gandon, F., d’Aquin, M., Staab, S., Tordai, A. (eds.) *ESWC 2014*. LNCS, vol. 8465, pp. 504–518. Springer, Heidelberg (2014)