Refining Large Integrated Identity Graphs using the Unique Name Assumption

Shuai Wang[1] ✉[0000−0002−1261−9930], Joe Raad[2][0000−0002−7891−7738], Peter Bloem[1][0000−0002−0189−5817], and Frank van Harmelen[1][0000−0002−7913−0048]

[1] Department of Computer Science, Vrije Universiteit Amsterdam, The Netherlands
{shuai.wang | p.bloem | frank.van.harmelen}@vu.nl
[2] LISN, University of Paris-Saclay, Orsay, France
joe.raad@lisn.fr

Abstract. The Unique Name Assumption (UNA) supposes that two terms with distinct identifiers from the same knowledge base do not refer to the same real-world entity. The UNA can be used to detect errors in large integrated knowledge bases. For example, some identity links can be erroneous if they are in a path that connects two entities (that refer to different real-world objects) defined in the same knowledge base. For large knowledge bases, however, the UNA does not always hold due to redundant IRIs that capture various encodings, languages, namespaces, versions, letter cases, etc. The UNA can still be useful for identifying erroneous links provided good adaption to the exceptions. For this, we propose a concrete definition of the UNA with tolerance towards multiple exceptions, namely the internal UNA (iUNA). To compare the iUNA and other variants of the UNA, we propose a generic algorithm that can be used for refinement. The algorithm employs an SMT (Satisfiability Modulo Theory) solver and takes advantage of the latter's ability to efficiently reason over equality. For evaluation, we identify erroneous links in an identity graph of half a billion triples extracted from the LOD Cloud, and compare our approach against community detection methods (Louvain and Leiden) as well as other identity refinement approaches.

1 Introduction

The question "*What is an entity?*" and the related question "*When are two entities equal?*" are not only longstanding philosophical questions[3] but are also longstanding technical issues in information systems [7]. The Semantic Web, and in its wake, Linked Open Data, have operationalized the notion of an "entity" as an Internationalized Resource Identifier (IRI): each is represented as an IRI, and using the same IRI implies referring to the same entity. Entities are connected

This is an extended paper of the ESWC paper of the same title. RQ5, Section 6.4, and some paragraphs in Section 7 as well as Appendix A and B were removed in the published conference paper due to page limit.

[3] https://plato.stanford.edu/entries/object/

by the identity links (e.g. `owl:sameAs`) to form identity graphs. Many existing approaches for detecting errors in identity graphs require information such as vocabulary alignments, textual descriptions [17, 8] or the presence of a large number of ontology axioms and alignment of the vocabularies [11, 14]. However, such information is often restricted to certain languages or simply not always available [17, 8], thus not appropriate for refinement tasks at web scale. Identity graphs on the web exhibit special properties which must be considered: they are integrated from multiple sources, sources can be multilingual, many suffer from a lack of maintenance and some have multiple encoding schemes.

Since `owl:sameAs` is a symmetric relation, we reduce the directed graph to a simple, undirected graph. In an undirected graph G, a *Connected Component* (CC) is a maximal subgraph with any two vertices connected by a path (Figure 1a). A *gold standard* is the ground truth that maps each node (IRI) to the real-world entity, which can be used for evaluation (Figure 1b). An *equivalence class* (EC) is a set of vertices corresponding to the same real-world entity (may or may not be connected by a path). In an identity graph, a CC is an EC if and only if all its nodes refer to the same real-world entity[4].

The Unique Name Assumption (UNA) supposes that two terms with distinct IRIs do not refer to the same real-world entity. Although the UNA does not always hold due to redundant IRIs that capture various encodings, languages, namespaces, versions, and letter cases, the UNA can still be useful for identifying erroneous links. We design a refinement algorithm that removes a minimal number of edges with good precision (Figure 1f). We compare the results against the Louvain algorithm (Figure 1c and 1d) and the Leiden algorithm (Figure 1e).

This paper focuses on four research questions:

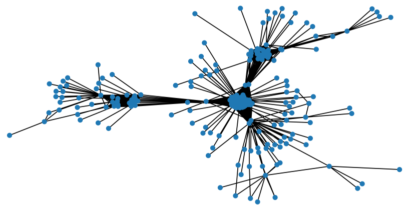
RQ1 How can we define a UNA for large integrated knowledge graphs?
RQ2 How do we validate various definitions of the UNA?
RQ3 Can the UNA give a reliable indication of errors in practice?
RQ4 Can we develop an efficient UNA-based algorithm for refinement?
RQ5 Is it possible to improve the results using additional information?

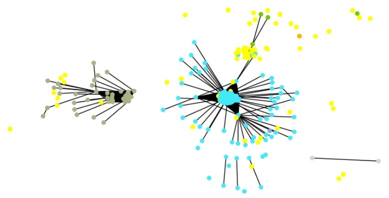
We present existing definitions of the UNA and related work in Section 2. In Section 3, we propose a new definition of the UNA and we test the different UNA definitions and examine their reliability for error detection in Section 4, by validating them over data of the LOD cloud. In Section 5, we present our refinement algorithm. Evaluation results of the algorithm together with its improvement are included in Section 6. Finally, discussion and future work are presented in Section 7. Our main contributions[5] are as follows:

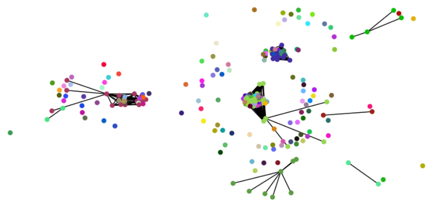
1. We propose a new definition of the UNA, namely the iUNA and check it against a large integrated knowledge graph together with other definitions.

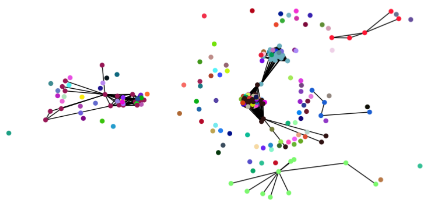
[4] However, when constructing the gold standard by annotating IRIs extracted from the Web, some may be annotated 'unknown' if the subject cannot be established.

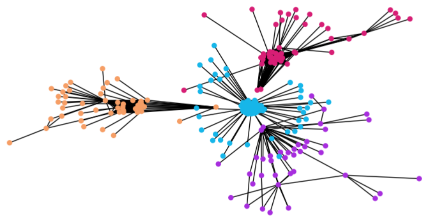
[5] The data is published on Zenodo (`https://zenodo.org/record/7765113`) with DOI `10.5281/zenodo.7765113`.


(a) An example CC with 633 nodes


(b) Its gold standard without erroneous edges (yellow nodes are those labeled 'unknown')


(c) A solution by the Louvain method (resolution = 0.01)


(d) A solution by the Louvain method (resolution = 1.0)


(e) A solution by the Leiden method

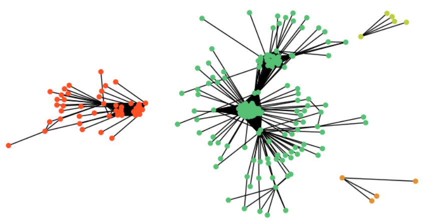

(f) A solution by our algorithm

Fig. 1: An example of a connected component (No. 4170), its gold standard, and solutions by the Louvain algorithm, the Leiden algorithm, and our algorithm.

2. We design an inconsistency-based refinement algorithm that evaluates definitions of the UNA by employing an SMT solver.
3. We publish a gold standard of over 8K manually annotated entities (200K owl:sameAs links) together with some additional information such as disambiguation, weights, redirection, and equivalence under different encoding schemes.
4. We introduce new evaluation metrics and provide a benchmark using our gold standard and algorithm.

2 Related Work

Estimates of the proportion of erroneous identity links in the semantic web range from around 3% [11, 15] to 20% [10]. Existing approaches for detecting errors in identity graphs fall into three categories[17]. *Content-based* approaches exploit the descriptions associated with each resource for evaluating the correctness of an identity link. They typically rely on additional information such as vocabulary alignments and textual descriptions for each entity. However, such information is not always available [17, 8] on open Web datasets, and in practice, such algorithms often do not scale to the size of the LOD Cloud. The *network-based* approaches [9, 16] take advantage of graph-theoretical algorithms for the detection of erroneous links. For instance, [16] rely on the Louvain community detection algorithm for assigning an error degree for each identity link. This error degree is based on the density of the community in which an identity link occurs in, and the weight of the `owl:sameAs` (i.e. reciprocally asserted owl:sameAs have a lower error degree, hence a higher chance of correctness). These error degrees are published online as part of the MetaLink dataset [3]. However, the accuracy of these methods is limited due to a lack of understanding of the underlying semantics. Finally, the *inconsistency-based* approaches [11, 14] hypothesise that `owl:sameAs` links that lead to logical inconsistencies have a higher chance of being incorrect. They typically require the presence of a large number of ontology axioms and alignment of the vocabularies.

The use of the UNA to detect errors in identity graphs is an inconsistency-based approach. This idea has been explored in [12, 19]. Despite that UNA is a well-defined definition in relational database theory (a.k.a. Unique Name Axiom) [18], the lack of an agreed-upon definition of UNA in semantic web leads to different conclusions. The primitive adaption of UNA in semantic web postulates that any two ground terms with distinct names are non-identical [12]. In the scope of integrated knowledge graphs, Valdestilhas et al. [19] formalise this as any two URIs in the same knowledge base cannot refer to the same thing in the real world. We name this definition *naive UNA*, or *nUNA* for short. In practice, an integrated knowledge graph violates the nUNA if at least one of its connected components (from the identity graph) has two entities from the same source.

Figure 2 is a fictional example of six entities from two knowledge bases (corresponding to nodes in light grey and dark grey, respectively). The six entities connected by the black edges form a connected component. The two equivalence classes are about the Netherlands (the three nodes on the right), and a city in Texas named Holland (the two nodes on the left). The node `ex:Holland` can be confusing (could be annotated as "unknown"). The blue arrow is an example how encoding schemes can lead to redundancy. Due to transitivity, the mistake between `ex:Holland,_Texas`, `ex:Holland` and `ex-fr:Pays-Bas` was carried over to other entities such as `ex-nl:Nederland`. This example shows how entities in various languages can be confusing. This connected component violates the nUNA: for the knowledge base of light grey, there are three entities in the connected components. This helps the detection of spurious links. Note that removing the links between `ex:Holland,_Texas` and `ex:Holland` and `ex-fr:Pays-Bas` results in three connected components, which are correct but still violate the nUNA.

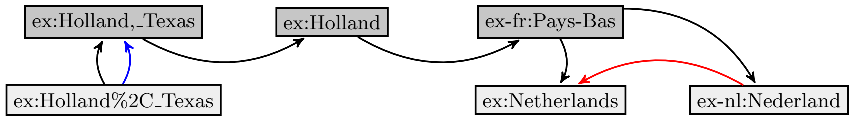


Fig. 2: An example CC with links expressing identity (black), redirection (red), and encoding equivalence (blue) (see also Section 3).

De Melo [12] points out that the Semantic Web is very different from traditional closed scenarios because multiple parties can publish data about the same entity using different identifiers. Thus, they propose to use a quasi-unique name constraint (*quasi UNA*, or *qUNA*) for entities: they use the namespace of an IRI as its source of provenance, with a focus on 6 major hubs including DBLP, DBpedia, FreeBase, GeoNames, MusicBrainz, and UniProt. This definition also takes into account some exceptions: two DBpedia entities from the same dataset/source do not violate the UNA if one redirects to the other, or either is a dead node (those that can no longer be resolved).

These definitions have several drawbacks in practice. First, both the nUNA and the qUNA lack a clear definition of *provenance*, i.e. the source of entities. The algorithm using the nUNA relies on LinkLion[6] for computing the provenance of entities [19]. That of the qUNA takes an entities' namespace as the source by default. As for DBpedia, the paper studied only the namespace `http://dbpedia.org/resource/` for violation and redirection. The algorithm developed based on nUNA outputs only partitions of the identity graph rather than the edges to remove [19]. Despite that the paper proposed to handle cases of DBpedia with exception, qUNA is restricted to awareness of redirect within DBpedia [12]. In fact, recent work estimates that between 45% and 83% of redirection links can be taken as identity link[7] [13]. Furthermore, the work in [12] does not specify how redirection and dead nodes were obtained. In addition, we believe that there are other forms of exceptions that must be considered. For example, the IRIs `wikidata.dbpedia.org/resource/Q6453410`, `www.wikidata.org/entity/Q6453410` and `wikidata.org/entity/Q6453410` are about the same entity but in different versions of Wikidata. Despite issues with the definition, the refinement algorithm using these two UNA definitions takes violations as hard constraints: entities are considered different as long as the UNA is violated. Due to the lack of a gold standard, neither definition was validated on real-world data, or compared with other existing baselines. In this work, we propose a new definition of the UNA that is suited for large integrated graphs on the Web and compare it with the existing UNA variations previously proposed by [12, 19].

	nUNA	**qUNA**	**iUNA**
Definition	Two URIs in the same KB cannot refer to the same thing	Refinement of nUNA, considering exceptions of DBpedia	Refinement of nUNA by considering multiple exceptions and provenance estimations
Provenance	Rely on LinkLion	Namespace (in 6 major hubs)	Three means of provenance
Exceptions	None	Redir. between some DBpedia entities	Encoding variants, redirection, dead nodes
Algorithm (see sections below)	Violation as hard constraint; returns partitions that are contradiction free	Violation as hard constraint; remove links that violate qUNA	Violations as hard and soft constraints; remove fewer identity links
Limitations (see sections below)	No tolerance towards exceptions; relies on an external server for provenance	Not enough exceptions taken into consideration; restricted definition of provenance; violations taken as hard constraints	Not every exception is included or handled explicitly. Can be relaxed by taking violations as soft constraints.

Table 1: Comparing the definition of the UNA

3 The iUNA

When examining the data in the LOD Cloud, we note that identity links are often used to connect the same entity in different language, versions or encodings. Therefore, we propose our own definition of the UNA, which we call the internal UNA (iUNA), to take these differences into account. Our iUNA definition assumes that two different IRIs $e1$ and $e2$ within the same namespace should refer to distinct real-world entities only when: a) they are in the same knowledge base according to certain provenance information, b) they don't satisfy any of the following exceptions:[8]

1. if $e1$ can be percent encoded/decoded into e_2 by one or more steps,[9]
2. if e_1 redirects to e_2 (or vice versa), or both redirect to the same location,
3. if at least one of e_1 and e_2 is a dead node, not found, unresolvable, redirects until reaching some error or has a timeout error while resolving.

To check whether two entities violate the iUNA, condition (a) requires us to check whether they are from the same knowledge base. This requires some form of provenance to determine where an entity is defined. The nUNA relies on the provenance information of LinkLion, which consists of multiple linksets. It is questionable if linksets can in fact be taken as the knowledge base where

[6] LinkLion (`https://www.linklion.org/`) is no longer available.
[7] The uncertainty is due to the presence of a large number of 'unknown' entities
[8] These exceptions are based on our manual examination of the entities in the linksets.
[9] For example, `ex:Bandon_(Oreg%C3%B3n)` and `ex:Bandon_(Oregón)` can be equivalent.

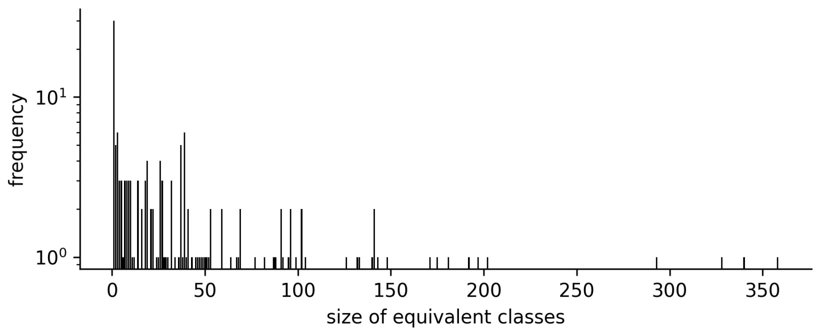


Fig. 3: Size distribution of the equivalence classes in the gold standard.

the entities are defined, not to mention that LinkLion is no longer available. As for the qUNA, it takes the namespace of an entity to define its knowledge base (regardless of the actual knowledge bases where the corresponding identity links are). This can be problematic for popular namespaces: an entity in DBpedia can be defined in one knowledge base but used in other knowledge bases. Authors can specify where an entity is defined using `rdfs:isDefinedBy`, but an ad-hoc examination shows that this information is rare. We therefore propose two additional means for the estimation of provenance of an entity e. Table 1 provides a comparison of the three UNA definitions.

Explicit sources: an explicit source of e is the object in any triple with subject e and predicate `rdfs:isDefinedBy` (or any equivalent or sub-properties).

Implicit label-like sources: an implicit label-like source of e is the RDF file containing triples where e is the subject and `rdfs:label` (or any of its equivalent or sub-properties) is the predicate.

Implicit comment-like sources: an implicit comment-like source of e is the RDF file containing triples where e is the subject and `rdfs:comment` (or any of its equivalent or sub-properties) is the predicate.

4 Testing the UNA

4.1 Dataset & Gold standard

We use the `http://sameas.cc` dataset [4], which provides the transitive closure of 558 million distinct `owl:sameAs` statements. These identity statements were extracted from the 2015 LOD Laundromat crawl [2] that provides more than 38 billion triples from over 650K RDF files. The identity links are distributed over 49 million connected components (CCs), with each CC being associated with a unique ID. We manually annotated all IRIs from 28 CCs with fewer than 1K nodes each. Our gold standard consists of 8,394 manually annotated entities covering a total of 232,311 `owl:sameAs` links. There are 987 entities (11.75%)

annotated as 'unknown'. A total of 209,160 edges (90.02%) are between nodes with the same annotation while 3,678 edges (1.58%) link entities with different manual annotations. The remaining edges involve at least one node annotated as 'unknown'. Based on this manual examination, we estimate the error rate to be between 1.58% and 9.98%. We divide our gold standard randomly into two parts of 14 files each for training and evaluation respectively. To better understand the gold standard, we show their size ECs and their distribution in Figure 3. The plot shows that redundancy is common in the LOD cloud. The majority of ECs contain fewer than 200 nodes, while there could be as many as 358 identifiers referring to the same real-world entity at the right end of the spectrum. This gives a reference for the setting of parameters in our algorithms in Section 5.

4.2 Validating the UNA

Using the gold standard, we validate our definitions (RQ2). For this, we use the sources of entities in our gold standard retrieved also from LOD Laundromat. Our examination shows that only 0.71% of the entities have an explicit source. In contrast, 61.97% of the entities have at least one implicit label-like source and 40.71% have a comment-like source. This indicates that explicit sources are too rare and thus we only use two variants of iUNA in this work: *iUNA-label* and *iUNA-comment* corresponding to label-like sources and comment-like sources respectively.

For each source, we analyze the number of entities in each EC. Although the original work that examines qUNA was restricted to only 6 major hubs' namespace as provenance, it can be easily adapted to any namespace. Thus, we generalize its definition of provenance in the experiments below. Considering that the nUNA lacks a proper definition of provenance, we use the label-/comment-like source defined for iUNA for the sake of comparison. Table 2 provides the proportion of sources with the number of entities in each implicit label-/comment-like source in the equivalence classes. A source follows the UNA if there is only one unique entity in the EC. An estimate of 1,351 out of 1,737 label-like sources follows the nUNA. On the other hand, 14.40% of the sources violate the nUNA by having two entities in at least one equivalence class in the gold standard, and an additional 7.82% of the sources violates the nUNA by having more than two entities. Table 2 shows that the iUNA is better than the nUNA and the qUNA in terms of capturing how the community is implementing the UNA in their knowledge bases. This also shows that taking encoding equivalence and redirection can indeed align the UNA with its use in practice. Thus, the algorithm should not remove all edges that violate the UNA when refining the identity graphs.

4.3 Detecting Errors Using UNA

In this section, we focus on **RQ3:** can the UNA give a reliable indication of identity errors in practice? Our analysis shows that the errors can be classified as two types. The first type are erroneous edges between entities that refer to two real-world entities. The others are edges involving nodes annotated as

		nUNA	qUNA	iUNA
one unique entity	label-like	1,351 (77.78%)	204 (59.48%)	**1,566 (90.15%)**
	comment-like	519 (68.56%)		**670 (88.51%)**
up to two entities	label-like	250 (14.40%)	69 (20.11%)	119 (6.85%)
	comment-like	153 (20.21%)		57 (7.53%)
more than two entities	label-like	136 (7.82%)	70 (20.41%)	52 (2.99%)
	comment-like	85 (11.23%)		30 (3.96%)

Table 2: Analysis of sources of the gold standard that follow the UNA

		Violation (%)	**Lower bound (%)**	**Upper bound(%)**
random		-	47.0	68.1
nUNA	label	61.9	33.4	49.8
	comment	42.5	32.6	46.2
iUNA	label	0.3	8.5	75.9
	comment	0.1	11.7	35.0
qUNA		1.4	16.1	61.3

Table 3: Percentage of pairs violating different definitions of the UNA with the lower/upper bound of their error rates using different sources

'unknown'. Thus, we provide upper and lower bounds of error rates depending on how these edges are treated. First, we study how two random entities in a connected component are identical. For this, in each connected component G in the gold standard, we sample $|V|$ (i.e. the number of nodes) different pairs of entities at random. The estimated error (proportion of non-identical pairs) is between 47.0% and 68.1%, depending on the interpretation of the nodes labeled "unknown" in the gold standard. We use this as our baseline for the analysis below (see the first row of Table 3).

For these same sampled pairs, we test the error rate and the UNA violation percentage for the three UNA definitions. The second row in Table 3 shows that when using label-like sources, 61.9% of the sampled pairs violate the nUNA, the estimated error is between 33.4% and 49.8%. In contrast, only 0.3% sampled pairs violates iUNA, with an error rate between 8.5% and 75.9%. Recall that 11.75% nodes were annotated "unknown". This analysis also indicates that such nodes are heavily involved in pairs violating the UNA. More pairs violate the UNA when using label-like sources than when using comment-like sources. In all cases, the lower bounds of error reduce when compared against that of randomly sampled pairs. Using iUNA with comment-like sources reaches the lowest error rate for the lower bound. These selected pairs are then used in the algorithm to identify erroneous edges in the paths that connect them.

Next, we study the impact of redirection. There are in total 13,922 nodes in the graphs that capture redirect relations[10]. We find that 3,072 out of 8,394

[10] Redirection was tested with the *requests* Python package using the *get* function with a max timeout of 5 seconds for connection and 25 seconds for reading.

entities were redirected. Among them, 5,528 correspond to new IRIs that are in the extended graph but not in the original graphs. There are in total 6,991 edges in the redirect graphs. Among them, 546 are between entities in the original graph with 504 correct ones and 8 erroneous ones. That is, the error rate is between 1.47% and 7.69%. In addition, we have 12,531 pairs of entities that redirect to the same entity in the extended graph. The error rate is between 4.29% and 6.32%.

Next we study the equivalent entities suffering from different encodings (recall the example given in Figure 2). We have 1,818 pairs of entities in the gold standard.[11] Among them, there are edges between 1,130 pairs in the original identity graphs with an error rate between 2.21% and 8.50%. We discovered 688 new pairs that differ only by encoding with an error rate between 1.16% and 14.83%. Finally, there is a pair of entities whose IRIs in alternative encoding are the same but they actually refer to different real-world entities. We conclude that though the exception do not always hold, they are often useful.

5 Algorithm Design

We limit the scope of refinement algorithms in this paper to removing erroneous identity links and forego identifying erroneous entities or adjoining additional links. The intuition is that for two inter-connected clusters, if there is more force pushing them apart than holding them together, then some edge(s) should be removed to split the clusters apart. The "force" that pushes the clusters apart are between pairs of entities violating the UNA. These pairs might not be directly connected, but they can be connected through multiple paths. The removed edges as the output of the algorithm is a *cut* for the graph. Computing an optimal cut whose removal makes the graph consistent within each CC is APX-hard (i.e. where there are polynomial-time approximation algorithms) [12]. We can encode this problem (as soft and hard clauses) to an optimization problem and employing an SMT solver [5]. The goal is to maximise the sum of weights over all soft clauses while satisfying all the hard clauses. We choose this approach because it enables fast reasoning over weighted constraints of relations of equality and inequality and it returns a sub-optimal answer in case of timeout.

5.1 Algorithm using UNA

Since the iUNA/nUNA requires the same parameters, we present the algorithm using the iUNA. That of qUNA can be derived simply by removing the parameters of redirect graphs and that of encoding equivalence. Algorithm 1 takes as input a graph G, the corresponding redirect graph G^R, the graph of equivalence under various encodings G^E, and a weighting scheme w. As a first step, we load H_{css} with the connected components of G. We obtain the corresponding subgraphs H^R_{cc}, H^E_{cc} from G^R, G^E respectively. G_{ccs}, together with G^R_{cc}, G^E_{cc} and the

[11] We used the *parse* function in the *rfc3987* and *urllib* Python library.

Algorithm 1: partition

```
1  Input: an identity graph $G$, a weighting scheme $w$, a graph of redirect $G^R$, a
     graph of equivalence under various encodings $G^E$
   Result: status $s$, a set of edges removed $A$, the graph of partitions $G_P$
2  initiate $A$ as an empty set (to store removed edges);
3  initiate $H_{ccs}$ as a set of the connected components of $G$;
4  while $|A|$ is increasing (no new edge to remove) and $H_{ccs}$ is not empty do
5      foreach $H_{cc} \in H_{ccs}$ do
6          (optional: obtain the corresponding subgraphs $H^R_{cc}$, $H^E_{cc}$ from $G^R$, $G^E$);
7          $(N_{ccs}, A') = \text{partition\_iter}(H_{cc}, w, H^R_{cc}, H^E_{cc})$;
8          $A := A \cup A'$;
9          remove $H_{cc}$ from $H_{ccs}$;
10         add new graphs $N_{ccs}$ that are not singleton to $H_{ccs}$.
11 remove $A$ from $G$ to get $G_P$;
12 return $(A, G_P)$.
```

weighting scheme is then taken as the input of Algorithm 2. The removed edges are collected in A. The algorithm stops when no more edges can be removed.

In the while-loop of Algorithm 1, there is a repeated call to Algorithm 2 that examines each graph of a connected component in H_{ccs} (line 7). Algorithm 2 takes advantage of an SMT solver's power of reasoning over weighted relations of equality and returns a solution within a given time bound. We first randomly sample some pairs of nodes. We keep those that violates the iUNA, denoted P (line 2). If there is at most one pair in graph G_{cc} that violates the iUNA, we keep the graph as it is (line 4). Otherwise, we initiate an SMT solver (line 5). For each node, we introduce a integer variable. We encode two hard clauses to ensure the values to be between 0 and M in the model m. These integer variables will eventually be assigned an integer value in the model m after solving.

Next, we explain how the soft clauses are generated. For each pair (s, t) in P, we obtain a clause $\text{NOT}(I_s = I_t)$ and associate it with a weight according to the weighting scheme w (line 10). Instead of taking all the edges of G_{cc}, we take the edges of its minimum spanning forest and a small sample of the edges to reduce the load on the SMT solver. In line 11, we obtain the minimum spanning forest F. For efficiency, we keep a set of edges in B (line 12) for the backpropagation process of SMT's internal algorithm design. The edges of $F \cup B$ forms the set of edges in G_{cc} to examine this round (line 11-14). Recall that in Section 4.3, our analysis showed that it provides relatively reliable information when considering redirection and equivalence under different encoding. Therefore, we encode the edges of the redirection (line 15-18) as soft clauses. The undirected graph is used for the checking of convergence of redirection of two entities (line 15, 17).

While not every soft clause is true in the model, all the hard clauses must be satisfied. The goal is to maximise the sum of weights over all soft clauses while satisfying all the hard clauses. Note that if an SMT solver fails to get an optimal

Algorithm 2: partition_iter

```
1  Input: a graph of connected component $G_{cc}$, a weighting scheme $w$, a graph
     of redirect $G^R_{cc}$, a graph of equivalence under various encodings $G^E_{cc}$
   Result: a set of graphs of connected components $N_{ccs}$, edges removed $A_{cc}$
2  obtain random pairs of nodes, select only those that violates the iUNA, as $P$;
3  if $|P| \leq 1$ then
4      return ($G_{cc}$, $\emptyset$).
5  initiate an SMT solver $o$;
6  foreach entity $e$ in $G_{cc}$ do
7      introduce an integer variable $I_e$ in the SMT solver;
8      assert hard clauses $(0 \leq I_e)$ and $(I_e \leq M)$ in $o$.
9  foreach pair $(s, t)$ in $P$ do
10     assert in $o$ a soft clause NOT($I_s == I_t$) with weight according to $w$.
11 let $F$ be the minimum spanning forest of $G_{cc}$;
12 sample a small amount of additional edges from $G_{cc}$ as $B$;
13 foreach pair $(s, t)$ in $F \cup B$ do
14     assert in $o$ a soft clause $(I_s == I_t)$ with weight according to $w$.
15 obtain $G'^R_{cc}$ the undirected graph of the (directed) graph $G^R_{cc}$;
16 foreach pair $(s, t)$ in $G'_{cc}$ do
17     if there is a path between $s$ and $t$ in $G'^R_{cc}$ then
18         initiate/update the weight of a soft clause $c_r$ in $o$ according to $w$.
19 foreach pair $(s, t)$ in $G^E_{cc}$ do
20     initiate/update the weight of a soft clause $(I_s == I_t)$ in $o$ according to $w$.
21 let $m$ be the model of $o$ after solving;
22 extract the removed edges $A_{cc}$ from $m$;
23 remove $A_{cc}$ from $G_{cc}$;
24 compute $N_{ccs}$ as the connected components without singletons;
25 return ($N_{ccs}$, $A_{cc}$).
```

solution within the timeout, it will return the best sub-optimal solution (line 21). The edge (s, t) remains if and only if I_s equals I_t in the model m (line 22).

The weighting scheme w consists of a series of functions that map clauses to weights: $w = (f_G, f_R, f_E, f_P)$. We used the training dataset to fine-tune the weighting scheme. For a soft clause c_e corresponding to an edge e, the weight is $f_G(c_e) + f_R(c_e) + f_E(c_e) + f_P(c_e)$. The first weighting scheme w_1 consists of four functions: f_G assigns the clause of each edge in the $F \cup B$ a weight of 5, the rest 0; Similarly, f_P assigns the clauses corresponding to pairs in P a weight of 2. f_R and f_E both increase the weight by 1 for that of G'^R_{cc} and $G^E_c c$ respectively. After some manual tuning, we provide an alternative weighting scheme w_2 with the corresponding values being 31, 16, 5, and 5, respectively. Other parameters and hyper-parameters were set according to Section 4.1 and fine-tuned. The upper bound M was set to 2+$|G_{cc}|$/50. A random selection of 12% of the edges from the original graph were kept in B. Finally, based on our experience with Z3, the timeout bound for SMT solving was set to $(|G_{cc}|/100 + 0.5)$ second.

6 Evaluation

6.1 Implementation

We used the *networkx* Python package[12] for the computation of the connected components and the minimum spanning forests. For the manual annotation of the entities, we used ANNit[13]. We used the implementation of the Leiden algorithm and the Louvain algorithm in CDlib[14]. As for SMT solver, we employed Z3[15] and used its Python binding [5]. We published all the code as an open source project[16]. All our experiments were conducted on the LOD Labs machine. It has 32 64-bit Intel Xeon CPUs (E5-2630 v3 @ 2.40GHz) with a RAM of 264GB.

6.2 Evaluation Metrics

While precision and recall are commonly used in evaluation metrics [17], the presence of 'unknown' annotations makes them less suitable for this task since no edge involving entity of 'unknown' counts toward precision or recall. Thus, precision and recall do not adequately capture the qualities. Moreover, we noticed that 11 graphs in our gold standard have no erroneous edges except those with nodes labeled "unknown". Therefore, we provide an additional metric. In its design, we focus on two properties that the equivalence classes should possess within the CCs resulting from refinement: (a) the equivalence class should not be separated over multiple CCs; (b) two equivalence classes should not share the same CC. This leads to the following metric for the graph G' that results from applying a refinement algorithm to G:

$$\Omega(G') = \sum_{C \in G'_{ccs}} \sum_{Q_e \in E(C)} \frac{|Q_e|}{|V|} \frac{|Q_e|}{|O_e|} \frac{|Q_e|}{|C|}.$$

Here, C iterates over all connected components in G', and $E(C)$ is a partitioning of the nodes in C by equivalence class, so that Q always represents the set of nodes within a given C that refers to the same real-world entity e. V represents the total number of vertices, and O_e is the set of all entities in G' referring to e.

[12] `https://networkx.github.io`

[13] ANNit is a user-friendly interface for fast annotation of entities and triples. See `https://github.com/shuaiwangvu/ANNit` for details.

[14] Community Discovery Library is a meta-library for community discovery in complex networks: `https://pypi.org/project/cdlib/`.

[15] `https://github.com/Z3Prover/z3`

[16] The code and implementation details are at `https://github.com/shuaiwangvu/sameAs-iUNA` together with the results of several parametric settings.

Within the summation, there are three factors. The first, $|Q_e|/|V|$ is the proportion of the current set of vertices to the total. This turns $\Omega(G')$ into a weighted sum over all subsets $|Q|$, with the weights summing to the total proportion of nodes not annotated "unknown". The second, $|Q_e|/|O_e|$, is 1 if all references to e are in C, and lower if there are more references in other connected components. This penalizes deviating from (a). The third, $|Q_e|/|C|$, is 1 if all nodes in C refer to e and lower if the connected component is shared with nodes referring to other entities. This penalizes deviating from (b). Note that if the graph contains no "unknown" nodes, the max. of Ω is 1.

6.3 Evaluation Results

We compare our algorithm using two variants of sources (implicit label-like and comment-like sources) with two weighting schemes (w_1 and w_2, as defined in Section 5) against the Louvain algorithm [6], the Leiden algorithm [1], as well as the result of MetaLink with two threshold values [3, 16]. Table 4 presents the results of the average of 5 runs for each method with best results highlighted. The Louvain algorithm removes the most amount of edges. It has the highest recall but relatively low precision. Recall the example in Figure 2, the results of Louvain can be smaller isolated components. This problem also exhibits in our evaluation, due to the significant amount of edges removed, its Ω values are low despite varying its resolution parameter from 0.01 to 1.0. Compared with Louvain, the result of the Leiden algorithm shows obvious improvements. There are fewer edges removed while the precision and Ω have improved for both the training set and the evaluation set. As for Metalink, we run the algorithm with two thresholds: 0.9 and 0.99 (only links with an error degree higher than the threshold are considered erroneous). There are fewer edges removed in both cases, with higher Ω values compared against that of Leiden and Louvain.

In almost all cases, using comment-like sources results in better precision values while having fewer edges removed. The difference of Ω between using label-like sources and comment-like sources is minor. In general, fewer links were removed when using the UNA and Metalink for refinement. Comparing the nUNA with the iUNA, we can see that using the nUNA results in more edges removed with a lower precision. When comparing the qUNA with the iUNA, we find as well that the qUNA removes a larger amount of edges, which leads to a slightly higher recall. In almost all settings, using the iUNA results in higher precision, which could be the benefit of better modeling using exceptions. The best Ω values in both sets are obtained using the qUNA, while using the iUNA results in better precision with similar Ω values. Compared with Metalink, our algorithm shows higher precision and better Ω values. Overall, our evaluation indicates that different algorithms have different advantages, but using the UNA shows clear benefits.

As for time efficiency, the Louvain and Leiden algorithm completes processing both the training and evaluation sets within 40 seconds. For the algorithm using the UNA, it takes around 8 minutes to process the training set in contrast to up to 27 minutes for the evaluation set. In addition, we note that up to three

		Training set				**Evaluation set**			
		precision	recall	Ω	$\|A\|$	precision	recall	Ω	$\|A\|$
Louvain	res=0.01	0.020	**0.803**	0.091	39,471.4	0.042	**0.727**	0.087	42,424.2
	res=1.0	0.020	0.778	0.087	39,226.2	0.042	0.660	0.084	43,610.0
Leiden		0.249	0.198	0.377	3,398.4	0.068	0.323	0.439	2,782.6
MetaLink	t=0.9	0.076	0.029	0.522	241	0.086	0.032	0.524	337
	t=0.99	0.036	0.004	0.591	58	0.013	0.001	0.635	99
nUNA	label, w1	0.126	0.150	0.590	406.2	0.042	0.063	0.597	684.6
	label, w2	0.153	0.181	0.591	529.0	0.061	0.075	0.580	697.4
	comment, w1	0.201	0.146	0.595	263.0	0.098	0.040	0.618	356.4
	comment, w2	0.209	0.178	0.597	360.2	0.063	0.036	0.606	431.2
qUNA	w_1	0.258	0.152	**0.641**	492.0	0.058	0.036	0.662	706.4
	w_2	0.227	0.174	0.640	566.6	0.101	0.054	**0.671**	634.2
iUNA	label, w1	**0.333**	0.127	0.606	78.0	0.122	0.013	0.652	236.8
	label, w2	0.204	0.118	0.616	125.8	**0.136**	0.028	0.647	235.0
	comment, w1	0.267	0.090	0.598	63.8	0.097	0.002	0.636	141.2
	comment, w2	0.258	0.117	0.607	133.2	0.117	0.003	0.638	173.8

Table 4: Evaluation of the Louvain algorithm with two resolution values, the Leiden algorithm, MetaLink with two threshold values, and our algorithm using different UNA and settings.

graphs in the evaluation set can suffer from timeout using our algorithm[17]. When there is a timeout, the SMT solver returns a sub-optimal solution. Our manual examination shows that some "harder" and larger graphs were distributed to the evaluation set when constructing the two sets.

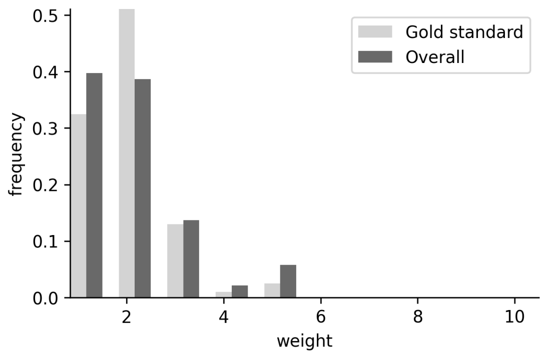


Fig. 4: Weight distribution of the `owl:sameAs` links in the LOD Laundromat.

[17] These are connected components with the IDs 14872, 4635725, and 37544.

6.4 Improving the Results

Next, we study if we can use additional information to improve results. We can define the weight of an edge as the number of datasets in which the corresponding triple can be found (not to be confused with the weight of a soft clause in the algorithm). Out of the 650K available LOD Laundromat files, the `owl:sameAs` links are distributed over 7,024 files. Thus, the weight of an edge can vary between 1 and 7,024. Figure 4 compares the weight distribution in the gold standard and across the entire (undirected) identity graph. The bar chart shows that most of the triples are associated with weights less than 5.

		Training set				**Evaluation set**			
		precision	recall	Ω	$\|A\|$	precision	recall	Ω	$\|A\|$
qUNA	w_1	0.258	0.152	0.641	492.0	0.058	0.036	0.662	706.4
	w_1^S	0.231	0.192	0.642	446.0	0.052	0.046	0.658	683.4
	w_1^D	0.302	**0.306**	0.642	637.6	0.044	0.051	0.683	738.4
	w_1^{SD}	0.275	0.218	0.640	523.2	0.039	0.068	0.662	682.2
	w_2	0.227	0.174	0.640	566.6	0.101	0.054	0.671	634.2
	w_2^S	0.236	0.209	0.644	638.0	0.042	0.034	0.668	645.6
	w_2^D	0.244	0.211	**0.645**	601.6	0.107	**0.077**	**0.675**	658.8
	w_2^{SD}	0.218	0.193	0.642	658.6	0.060	0.064	0.666	694.2
iUNA	label, w_1	0.333	0.127	0.606	78.0	0.122	0.013	0.652	236.8
	label, w_1^S	0.216	0.111	0.601	89.4	0.095	0.020	0.639	251.8
	label, w_1^D	**0.404**	0.269	0.606	271.8	0.106	0.057	0.661	242.4
	label, w_1^{SD}	0.327	0.122	0.608	150.4	0.070	0.092	0.661	262.2
	label, w_2	0.204	0.118	0.616	125.8	0.136	0.028	0.647	235.0
	label, w_2^S	0.141	0.094	0.607	133.6	0.120	0.026	0.649	228.4
	label, w_2^D	0.278	0.138	0.617	163.0	**0.143**	0.035	0.661	200.6
	label, w_2^{SD}	0.218	0.114	0.610	150.4	0.117	0.070	0.664	295.6
	comment, w_1	0.267	0.090	0.598	63.8	0.097	0.002	0.636	141.2
	comment, w_1^S	0.162	0.068	0.584	67.6	0.106	0.011	0.626	126.2
	comment, w_1^D	0.351	0.135	0.593	211.4	0.123	0.046	0.639	193.0
	comment, w_1^{SD}	0.336	0.083	0.598	134.2	0.120	0.054	0.631	134.8
	comment, w_2	0.258	0.117	0.607	133.2	0.117	0.003	0.639	173.8
	comment, w_2^S	0.248	0.088	0.596	99.8	0.086	0.014	0.634	192.2
	comment, w_2^D	0.261	0.110	0.611	120.4	0.127	0.033	0.640	166.0
	comment, w_2^{SD}	0.187	0.091	0.593	117.0	0.109	0.057	0.637	191.2

Table 5: Evaluation results of the algorithm using additional information with extended weighting schemes.

In addition, we find a significant amount of entities that correspond to disambiguation pages in Wikipedia. Of the 3,678 edges in the gold standard that are identified as erroneous, 1,395 edges (38%) involve at least one entity about disambiguation[18]. This can not only make the results less stable but also confirm our finding that the precision-recall is not suitable for this problem. For all links in the gold standard, the error rate is between 27.27% and 71.81% when disambiguation entities are involved. This is significantly higher than the average error rate in the gold standard. In addition, we noticed that after removing 501 disambiguation entities, the largest connected component is reduced from 177,794 to 82,685 entities (a reduction of 53.4%).

As a primitive experiment, we extend the weighting scheme by increasing the weight of the corresponding soft clause by 2 when the weight of the edge is ≥ 2 (denoted w_S). We take also such disambiguation into account. If a clause is about an edge involving at least one disambiguation entity, its weight is reduced by 5 (denoted w_D). When both conditions apply, we denote w_{SD}. Table 5 shows that taking disambiguation entities into account can improve both precision and recall noticeably in most cases. However, there is little improvement in Ω. This addresses the importance of taking semantics into consideration in the refinement of the identity graphs. However, using solely the weights of the edges or both conditions results in no significant improvement.

7 Discussion and Future Work

In this paper, we studied three definitions of UNA and proposed a UNA-based identity refinement approach. RQ1 was answered by defining the iUNA that considers certain exceptions that are common in large integrated graphs. For RQ2 and RQ3, we created a gold standard and compared the reliability of iUNA against the qUNA and the nUNA. For RQ4, we proposed an identity refinement algorithm and evaluated its performance on different definitions of UNA. Finally, we explored the use of additional information to improve the results to answer RQ5.

Strictly speaking, our gold standard is not large enough for an accurate estimate of the error rate of the entire identity graph. Using our sample, we found that among the 3,678 erroneous edges, only 5 entities have multiple label-like or comment-like sources. This indicates that redundancy is not the direct cause of the error. This contradicts the conclusion of [12] (see type 2 error: consistency and conciseness error). However, based on this gold standard, it is possible to generate synthetic data for the evaluation of future methods that can handle larger connected components. It can also be used to trace the source of error, which may inspire future algorithms. Appendix A shows how the gold standard can be used for the evaluation of other identity relations.

The performance of our algorithm is sensitive to the parameters and hyperparameters. For example, the upper bound for each integer value M can significantly influence the results if too small. Future work includes studying how our algorithm scales with different time limits, automatic tuning of the parameters,

[18] The disambiguation entities were identified where there is a triple with the relation `dbp:wikiPageUsesTemplate` and the object `dbr:Template:Disambiguation` or about a select of 17 multilingual relations with similar meaning.

and extending the gold standard. The results of some other parametric settings are included in the supplementary material in the repository.

The performance of MetaLink is comparable with the best outcome of our algorithms. However, our analysis shows that no more than 10% edges removed are shared between Metalink and our algorithms in various settings. It could be promising to explore a hybrid approach in future work. Since our evaluation confirms the superiority of the communities detected using the Leiden algorithm compared to Louvain, it is also reasonable to quest how far the results can be improved if MetaLink uses Leiden's outputs for calculating its error degree.

The identity graph we study contains a large number of connected components of size two, as well as two very large connected components. The biggest CC in this dataset has 177,794 entities and 2,849,426 edges (No. 4073). The second biggest has 21,191 entities and 101,269 edges (No. 142063). The rest are significantly smaller with no more than 5076 nodes. Some past attempts using SMT solvers have also discovered the bottleneck in scalability [20, 21]. In future work, we plan to design scalable algorithms following a divide-and-conquer approach for the handling of large connected components using pairs of entities that violate the UNA as heuristics. More specifically, by removing 501 nodes identified captured by the first two relations about disambiguation mentioned above, we manage to break the largest component (No. 4073) into a sequence of smaller connected components with 89,215 entities, 2,258 entities, and some smaller components. We are interested in how the removal of such entities reduces the size of connected components. Following that, we are interested in designing algorithms that can scale to the entire identity graph.

Our analysis shows that 211,348 (90.98%) out of 232,311 edges in the gold standard are about DBpedia entities between different languages[19]. Moreover, among 3,678 erroneous edges, 3,029 (82.35%) involve multilingual DBpedia entities. These links could be automatic generated using transitive closure or inherited as DBpedia enriches. Our analysis shows that these edges have not only made the identity graph more complex, but also less usable as the errors propagate through the graph due to the transitivity of `owl:sameAs`. Lessen such edges can benefit the correctness of the identity graph and improve the efficiency of refinement algorithms.

In our work, we noticed that a significant amount of entities are dead nodes, some are equivalent under various encodings, and some are redirect of each other. In future work, it worth exploring how the identity graph and the results of refinement would change after removing dead nodes and merging some entities.

In contrast to graph-based methods, our algorithm takes the logical properties into account. In future work, we would like to take advantage of the unsat core of the SMT solver and provide some essential explanation for the removal of each edge.

Our analysis shows that a significant amount of 11.75% entities were annotated 'unknown' in the gold standard. Our manual assessment shows that 526

[19] We identify the language of an entity by its namespace. For example `http://ru.dbpedia.org/resource/` is assumed to be in Russian.

(53.29%) of them are leaf nodes. Among the remaining 461, 179 (38.83%) of them have neighbours with more than one annotations other than ‘unknown’. This could be a reason for the wide range of error rate in Section 4.3. 2 of them have only neighbours annotated ‘unknown’. The remaining 280 (60.74%) of them have exactly one annotation other than ‘unknown’.

Fig 1 shows that some resulting graphs consist of some singletons. In this work, we did not add any new edges. It makes sense to consider adding some edges to entities that form singletons in the output of the refinement algorithm. Some primitive analysis is included in Appendix B.

We noticed that given longer processing time, the returned result of SMT solver can be improve with fewer cases of timeout. While our algorithm uses the SMT solver as a standalone tool, it is possible to take full advantage of the SMT solver by calling its internal functions.

All the algorithms except the Louvain algorithm presented in this paper suffer from low recall (see Table 4). This is partially due to the optimisation criteria that limits the number of edges removed. Table 5 demonstrates the potential to use additional information to further improve the results. The Ω value indicates that some correct edges to remove could be adjacent to the edges removed by the algorithms. Future work includes taking multilingual labels into account to further refine the result by removing alternative adjacent edges instead.

These experiments and analysis of the corresponding evaluation results show the necessity of developing accurate and scalable hybrid refinement algorithms for integrated identity graphs. The problem boils down to reasoning over equality with optimization, which could be a use case of future algorithms in graph theory as well as applications that use multiple sources of information. Our analysis also addresses the importance of verifying identity links before consuming the corresponding linked open data in real-world scenarios in future work.

Acknowledgement

The project was funded by the NWO TOP grant. The authors would like to thank Jacco van Ossenbruggen, Victor de Boer, Emma Beauxis-Aussalet, and colleagues for their advice. The final version of the paper has taken into account the comments by the anonymous reviewers, which we appreciate.

A Reusing the gold standard for other relations

The gold standard can also be used for the evaluation of other identity relations such as `skos:closeMatch`, `skos:exactMatch`, etc. Similarly, it is also possible to test that of `owl:differentFrom` and `rdfs:seeAlso`. We retrieve from the LOD-a-lot knowledge graph the edges corresponding to the entities in the gold standard. Our examination shows that there is no overlapping edge in the gold standard with other identity relations except `rdfs:seeAlso`. For the corresponding graph, there are 6,185 edges and 3,178 nodes. Further analysis using the gold standard of `owl:sameAs` shows that between 2.70% and 9.73% of the pairs of `rdfs:seeAlso`, are not equivalent. Recall that the error rate of `owl:sameAs` is between 1.58% and 9.98%. This could be useful additional information for future work.

B Singletons in the resulting graphs

Table 6 shows that the methods in evaluation result in some singletons after the removal of erroneous edges. We examine these singletons in three measures: the proportion of singletons not connected to major CC (PS); the proportion of singletons annotated 'unknown' (PU); the proportion of singletons with unique annotation only for themselves (i.e. no other entity of the same annotation, PT). Generally speaking, the more edges removed, the more singletons there are. The numbers in the column of PS indicate that adding identity links that connect singletons with their corresponding major connected components could be beneficial for the result. In fact, this would improve the Ω value. This analysis shows that adding some edges could be a beneficial additional step (e.g. using string matching) in the refinement algorithm.

		#singletons	PS (%)	PU (%)	PT (%)
Louvain	res=0.01	4947.0	84.93	13.03	2.05
	res=1.0	4499.0	81.62	11.22	71.57
MetaLink	t=0.9	127	41.73	15.75	42.51
	t=0.99	57	17.54	12.28	70.17
nUNA	label, w1	486.2	79.94	15.35	4.71
	label, w2	473.2	80.40	17.09	2.51
	comment, w1	112.2	95.74	3.97	0.29
	comment, w2	103.2	94.27	4.10	1.64
qUNA	w1	226.4	54.73	43.07	2.20
	w2	202.6	43.44	54.62	1.93
iUNA	label, w1	116.4	41.27	54.24	4.50
	label, w2	111.6	32.60	62.92	4.49
	comment, w1	33.2	82.96	1.75	15.29
	comment, w2	32.0	87.92	5.39	6.69

Table 6: Singletons and their semantics.

References

[1] Vincent A. Traag et al. "From Louvain to Leiden: guaranteeing well-connected communities". In: *CoRR* abs/1810.08473 (2018). arXiv: `1810.08473`. URL: `http://arxiv.org/abs/1810.08473`.

[2] Wouter Beek et al. “LOD laundromat: a uniform way of publishing other people’s dirty data”. In: *International semantic web conference*. Springer. 2014, pp. 213–228.

[3] Wouter Beek et al. “MetaLink: A Travel Guide to the LOD Cloud”. In: Lecture Notes in Computer Science. Springer, 2020, pp. 481–496. ISBN: 9783030494605.

[4] Wouter Beek et al. “sameAs.cc: The closure of 500M owl:sameAs Statements”. English. In: *The Semantic Web - 15th International Conference, ESWC, Proceedings*. Lecture Notes in Computer Science. Springer/Verlag, 2018, pp. 65–80.

[5] N. Bjørner. “Engineering theories with Z3”. In: *Asian Symposium on Programming Languages and Systems*. Springer. 2011, pp. 4–16.

[6] Vincent D Blondel et al. “Fast unfolding of communities in large networks”. In: *Journal of statistical mechanics: theory and experiment* 2008.10 (2008), P10008.

[7] Peter Pin-Shan Chen. “The entity-relationship model—toward a unified view of data”. In: *ACM transactions on database systems (TODS)* 1.1 (1976), pp. 9–36.

[8] John Cuzzola et al. “Filtering Inaccurate Entity Co-references on the Linked Open Data”. In: Sept. 2015, pp. 128–143. ISBN: 978-3-319-22848-8. DOI: 10.1007/978-3-319-22849-5_10.

[9] Christophe Guéret et al. “Assessing linked data mappings using network measures”. In: *Extended Semantic Web Conference*. Springer. 2012, pp. 87–102.

[10] Harry Halpin et al. “When owl:sameAs Isn’t the Same: An Analysis of Identity in Linked Data”. In: *The Semantic Web – ISWC 2010*. Ed. by Peter F. Patel-Schneider et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 305–320.

[11] Aidan Hogan et al. “Scalable and distributed methods for entity matching, consolidation and disambiguation over linked data corpora”. In: *Journal of Web Semantics* 10 (2012). Web-Scale Semantic Information Processing, pp. 76–110. ISSN: 1570-8268. DOI: https://doi.org/10.1016/j.websem.2011.11.002. URL: https://www.sciencedirect.com/science/article/pii/S1570826811000813.

[12] Gerard de Melo. “Not Quite the Same: Identity Constraints for the Web of Linked Data”. In: *AAAI*. 2013.

[13] I. Nasim et al. “What does it mean when your URIs are redirected? Examining identity and redirection in the LOD cloud”. In: *Workshop on Managing the Evolution and Preservation of the Data Web (MEPDaW)*. 2022.

[14] Laura Papaleo et al. “Logical Detection of Invalid SameAs Statements in RDF Data”. In: *EKAW*. 2014.

[15] Joe Raad. “Identity Management in Knowledge Graphs”. doctoral dissertation. PhD thesis. University of Paris-Saclay, 2018.

[16] Joe Raad et al. "Detecting erroneous identity links on the web using network metrics". In: *International Semantic Web Conference (ISWC)*. Springer. 2018, pp. 391–407.

[17] Joe Raad et al. "The sameAs Problem: A Survey on Identity Management in the Web of Data". In: *CoRR* abs/1907.10528 (2019). arXiv: `1907.10528`. URL: `http://arxiv.org/abs/1907.10528`.

[18] Raymond Reiter. "Towards a Logical Reconstruction of Relational Database Theory". In: *On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases, and Programming Languages*. Ed. by Michael L. Brodie, John Mylopoulos, and Joachim W. Schmidt. New York, NY: Springer New York, 1984, pp. 191–238. ISBN: 978-1-4612-5196-5. DOI: `10.1007/978-1-4612-5196-5_8`. URL: `https://doi.org/10.1007/978-1-4612-5196-5_8`.

[19] Andre Valdestilhas et al. "CEDAL: Time-Efficient Detection of Erroneous Links in Large-Scale Link Repositories". In: (Aug. 2017). DOI: `10.1145/3106426.3106497`.

[20] S. Wang et al. "Refining Transitive and Pseudo-Transitive Relations at Web Scale". In: *The Semantic Web - 18th International Conference, ESWC 2021, Proceedings*. Lecture Notes in Computer Science. Springer Science and Business Media Deutschland GmbH, 2021, pp. 249–264. DOI: `10.1007/978-3-030-77385-4_15`.

[21] S. Wang et al. "SUBMASSIVE: Resolving subclass cycles in very large knowledge graphs". In: *Workshop on Large Scale RDF Analytics*. 2020.