Refining Large Integrated Identity Graphs using the Unique Name Assumption

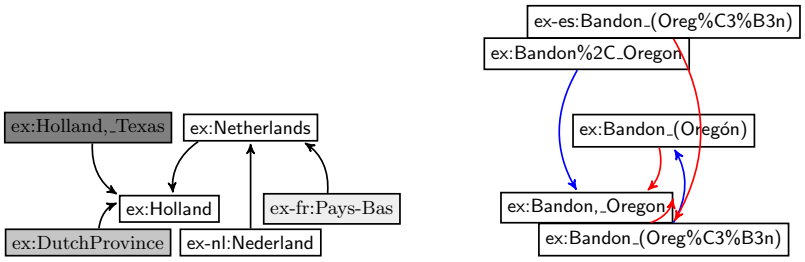
Abstract. The Unique Name Assumption (UNA) supposes that two terms with distinct IRIs from the same dataset do not refer to the same real-world entity. The UNA has been used to refine identity graphs: if the UNA is true, then identity links (e.g. those with relation `owl:sameas`) between two entities defined by the same dataset can be considered erroneous. While this works on a small scale, it has not been validated on the scale of large integrated knowledge graphs, such as the LOD cloud, which contain data from many different sources. We propose a concrete definition of the UNA with tolerance towards exceptions, namely the internal UNA (iUNA). We compare our definition with existing definitions with respect to some real-world datasets. Based on the iUNA, we propose an algorithm to identify erroneous links in an identity graph of half a billion triples extracted from the LOD Cloud. The algorithm employs an SMT solver and takes advantage of the latter's ability to efficiently reason over equality and remove a minor amount of links. Finally, we demonstrate how additional information can improve the accuracy.

1 Introduction

The question "*what is an entity*" and the related question "*when are two entities equal*" are not only longstanding philosophical questions[1] but are also longstanding technical issues in information systems [6]. The Semantic Web, and in its wake, Linked Open Data, has operationalised the notion of an "entity" as an IRI: each is represented as an IRI, and using the same IRI implies referring to the same entity. Although this improves over the ambiguous use of names in everyday language, it still leaves open the other direction of the identity question: can different IRIs represent the same real world entity? The Unique Name Assumption (UNA) supposes that two terms with distinct IRIs do not refer to the same real-world entity. While this works on a small scale, the UNA is an unrealistic assumption on the scale of large integrated knowledge graphs, such as the LOD cloud, which contain data from many different sources.

As a relaxation of the UNA, the `owl:sameAs` predicate allows connected entities with different IRIs that still refer to the same real-world entity [13, 15]. Figure 1a is a fictional example with six entities from three namespace prefixes (`ex`, `ex-fr`, and `ex-nl`) where there are four knowledge bases (four colors). The three entities to the right indicate that the same concept may be defined in different languages. Due to transitivity, the mistake between `ex:Holland` and

[1] `https://plato.stanford.edu/entries/object/`



(a) A fictional example of erroneous `owl:sameAs` links between four different entities named Holland: the country, the former province, and the city in Texas.

(b) An example of redirect links (red) and equivalence under different encodings (blue).

Fig. 1: Examples of subtleties in refining integrated identity graphs.

`ex:Netherlands` was carried further to the two entities to the left. This example shows how erroneous links can accumulate. Previous research estimates the percentage of erroneous identity links to be in the range of 3% [10] to 4% [13], while earlier work estimates this number to be as high as 20% [9].

In this paper, we present a more realistic definition of the UNA for integrated graphs, and we investigate the refinement of identity graphs by detecting and removing erroneous links based on different definitions of UNA.

1.1 Related Work

Existing approaches for detecting errors in identity graphs fall into three categories[15]. *Content-based* approaches exploit the descriptions associated to each resource for evaluating the correctness of an identity link. They typically require textual descriptions for each entity. In practice, such algorithms often do not scale to the size of the LOD Cloud and rely on additional information such as vocabulary alignments, which is not always available [15, 7]. The *network-based* approaches [8, 14] take advantage of graph-theoretical algorithms for the detection of erroneous links (e.g. using the Louvain algorithm [5]). However, these methods are not as accurate as the others. Finally, the *inconsistency-based* approaches [10, 12] hypothesise that owl:sameAs links that lead to logical inconsistencies have higher chances of being incorrect. However, such methods 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, but *without* the need for expert knowledge or axioms. This idea has been explored in [11, 16]. However, the lack of an agreed-upon definition of UNA leads to different conclusions. The CEDAL algorithm [16] uses map-reduce: by collecting nodes into groups that do not violate the UNA and merging groups without violations, the algorithm returns a solution without UNA violation. However, our examination in Section 3.2 shows that it is not always the case that

there is exactly one entity from each source. [11] uses linear program relaxation, but the algorithm does not consider the transitivity of equivalence.

1.2 Contributions & structure of this paper

This paper focuses on five research questions: **RQ1:** how can we define UNA for large integrated knowledge graphs? **RQ2:** how do we validate the definitions proposed? **RQ3:** can UNA give a reliable indication of identity errors in practise? **RQ4:** can we define an efficient algorithm for the refinement of the identity graphs? **RQ5:** is it possible to improve the results using additional information from the graph?

We present existing definitions of UNA and propose ours in Section 2. Section 3 tests the UNA definitions by validating them over data of the LOD cloud and further examines their reliability for error detection in Section 3.2 and Section 3.3 respectively. We present our refinement algorithm in Section 4 and its implementation details in Section 5.1. Finally, the evaluation metrics and results are included in Section 5, followed by discussion and future work in Section 6.

Our main contributions[2] 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.
2. We design an inconsistency-based refinement algorithm using the iUNA by employing an SMT (Satisfiability Modulo Theories) solver.
3. We publish a gold standard of over 8K manually annotated entities (200K owl:sameAs links), with some additional information about weights of edges, estimated sources of entities, and redirect relations between entities.
4. We introduce new evaluation metrics and provide a benchmark using our gold standard and study how to improve the performance of our algorithm.

2 Preliminaries

2.1 Integrated Knowledge Graphs

A *knowledge graph* is a directed and labelled graph $G = \langle V, E, \Sigma_E, l_E \rangle$, where V is the set of nodes, $E \subseteq V \times V$ the set of edges, and Σ_E is the set of edge labels. A function $l_E : E \to 2^{\Sigma_E}$ assigns to each edge a set of labels from Σ_E. Given a specific relation $R \in \Sigma_E$, we denote its subgraph $G_R = \langle V_R, E_R \rangle$. When R is an equivalence relation I, the subgraph is referred to as the *identity graph*. In this paper, we study only the case of `owl:sameAs`. Due to symmetry, we will use G_I as an undirected graph, and we will ignore reflexive edges. An *integrated knowledge graph* $\mathbf{G} = \langle \mathbf{V}, \mathbf{E}, \Sigma_\mathbf{E}, l_\mathbf{E} \rangle$ is a combination of a set of N knowledge graphs $\{G_1, \ldots, G_N\}$, with $\mathbf{G}$ representing the union of these graphs. An *integrated identity graph* $\mathbf{G}_I$ is a combination of graphs where $\Sigma_\mathbf{E} = \{I\}$.

[2] The code, the datasets, and their description are at `https://figshare.com/s/71b50777d44164209765`.

For an undirected graph G, a *Connected Component* (CC) is a maximal subgraph with any two vertices connected by a path, and G may consist of multiple CCs, denoted G_{cc}. A *gold standard* is a map g of IRIs U to their real-world entities or 'unknown'. An identity link between e_i and e_j is erroneous if $g(e_i) \neq g(e_j)$ and neither is 'unknown'. An *equivalent class* (EC) is a set of vertices corresponding to the same real-world entity. E.g. Figure 1a has an EC containing the three entities on the right, in addition to three singleton ECs. All identity links involving `ex:Holland` are erroneous. A CC may consist of multiple densely inter-connected *clusters*. These clusters are often interlinked by some erroneous edges. Figure 2a shows an example.

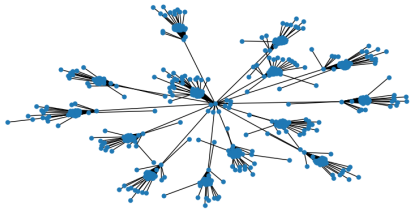
2.2 The Unique Name Assumption

The classical UNA postulates that any two ground terms with distinct names are non-identical [11]. In the scope of integrated knowledge graphs, [16] formalises 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. More formally, we can define $\eta(e_i)$ as the sources of an entity e_i. For example, a source of an entity could be the file where the entity is first introduced, typically with some additional information such as label, comments, abstract, title. An integrated knowledge graph violates the naive UNA if there exists two entities e_i and e_j referring to the same real-world entity s.t. $\eta(e_i) \cap \eta(e_j) \neq \emptyset$. De Melo [11] extends this definition by proposing the use of a quasi unique name constraint for entities, by taking the redirect relations and dead nodes (those that can no longer be resolved) into account. For example, two DBpedia entities from the same dataset/source do not violate the UNA if one redirects to the other, or one is a dead node. We refer to it as the *quasi UNA*, or *qUNA*. In practice, they take this redirect awareness between entities in major hubs as exemptions[3]. These definitions have several drawbacks when applied at Web scale. First, both nUNA and qUNA lack a clear definition of *provenance*. While Valdestilhas et al. rely on LinkLion[4] for computing the provenance of entities, [11] remains unclear, and does not specify how redirection and dead nodes were obtained . Furthermore, due to the lack of gold standard, both definitions were not validated with respect to real-world data before being used for refinement.

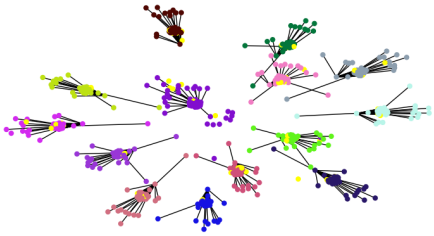
When examining the data in the LOD Cloud, we noticed that the use of identity links varies between linking entities from different languages, versions, or encoding. Moreover, most data publishers follow their own policies to avoid duplicates within their datasets. Therefore, we propose our own definition of the UNA, which we call the internal UNA (iUNA), to take these differences into account. Our iUNA definition assumes that two different IRIs within the same namespace should refer to distinct real-world entities only when they are defined in the same source. Our definition also takes into account the following exceptions as non-violations of the iUNA between two entities e_1 and e_2:

[3] [11] is restricted to only six major hubs: DBLP, English DBpedia, FreeBase, GeoNames, MusicBrainz, and UniProt.

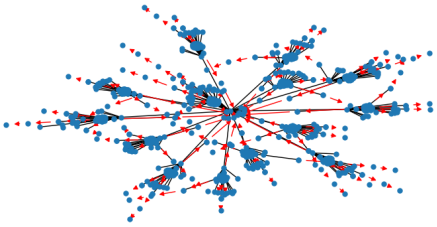
[4] LinkLion (`https://www.linklion.org/`) is no longer available for comparison.



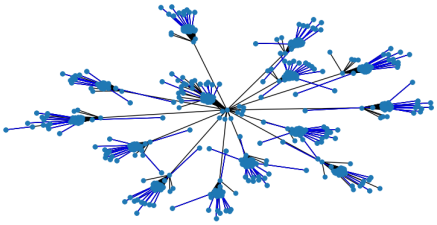
(a) An example CC with 633 nodes and 6011 edges.



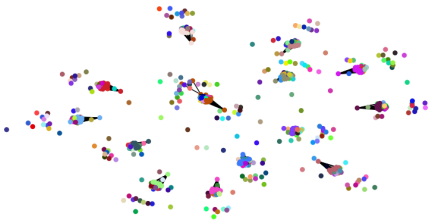
(b) Its gold standard with erroneous edges removed (yellow for unknown)



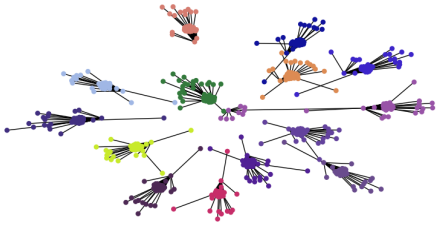
(c) The CC and its (directed) edges of redirection in red. New nodes are IRIs that do not exist in CC.



(d) The CC and its (undirected) edges between nodes differ only by encoding (in purple)



(e) The Louvain Method



(f) Our algorithm (using iUNA-label)

Fig. 2: An example of a connected component (No. 240577), its gold standard, necessary information and a comparison of two solutions.

1. if e_1 redirects to e_2, or vice versa,
2. if e_1 corresponds to the percent-encoding [3] of e_2 or vice versa[5],
3. if e_1 or e_2 are dead nodes, not found, unresolvable or redirects until reaching some error or not found, or has timeout error while resolving.

All the definitions above depend on the provenance of the entities. This is not usually available at Web scale, so we use the following methods to estimate the provenance for an entity e.

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).

[5] For example, `ex:Bandon_(Oreg%C3%B3n)` and `ex:Bandon_(Oregón)` can be equivalent.

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.

An examination of our gold standard (see below) indicates that only 0.71% of the entities has an explicit source. In contrast, 61.97% of the entities has at least one implicit label-like source and 40.71% has 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 example, in Figure 1a, there can be label-like or comment-like information from different datasets. Indicated by the color, we can reckon that the nodes in white (`ex:Netherlands`, `ex-nl:Nederland`, and `ex:Holland`) have label-like information in source s_1. While `ex-fr:Pays-Bas`, `ex:DutchProvince`, and `ex:Holland,_Taxas` have label-like information in source s_2, s_3, and s_4 respectively. Therefore, only the entities in source s_1 are then tested for the violation of iUNA. The iUNA allows multiple IRIs with different namespaces to refer to the same real-world entity. For example, `http://af.dbpedia.org/resource/Agra` and `http://ca.dbpedia.org/resource/Agra` are both entities published by `dbpedia.org` but are of different languages. The IRIs `http://wikidata.dbpedia.org/resource/Q6453410`, `http://www.wikidata.org/entity/Q6453410` and `http://wikidata.org/entity/Q6453410` are about the same real-world entity but in different versions of wikidata. In addition, we allow IRIs that differ only in their string encoding, or IRIs which ultimately redirect to the same location.

3 Testing the UNA

3.1 Dataset & Gold standard

We use the `http://sameas.cc` dataset [2], which provides the transitive closure of 558 million distinct `owl:sameAs` statements. These identity statements were extracted from the 2015 LOD Laundromat crawl [1] that serves, in a uniform and standards-compliant format, more than 38 billion triples from over 650K RDF files. The identity links are distibuted over 49 million 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 which is annotated as 'unknown'. This leads to an estimate of 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 equivalent classes in the gold standard, we show their size distribution in Figure 3a. This distribution shows that redundancy is common in the LOD cloud. It indicates that the majority of equivalent classes in these CCs contain fewer than 200 nodes, while there could be as many as 358 entities referring to the same real-world entity at the far end of the spectrum. Thirty equivalent classes consist of only one entity. This gives a reference for the setting of parameters in Section 4.

3.2 Validating the UNA

Next, we validate our definitions (RQ2). We analyse how the sources correspond to the number of entities in each equivalent class in the gold standard. Figure 3b presents the number of entities in each implicit label-like source across the equivalent classes. The left-most bars indicate the number of label-like sources that corresponds to only one entity in an equivalent class. An estimate of 21,197 out of 22,671 sources follows the nUNA, amounting to 93.50%. Similarly, 94.43% sources follow the qUNA, and 94.11% sources follow the iUNA. As in Figure 3c, using comment-like information, we estimate that 14,878 out of 15,266 sources has comment-like information for only one entity each, which amounts to 97.46%. Similarly, 96.77% sources follow the qUNA and 97.09% sources follow the iUNA.

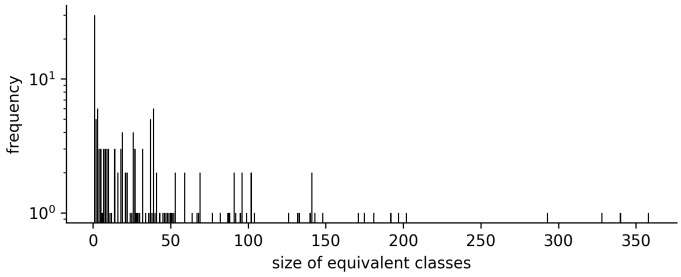
In both qUNA and iUNA, the number of entities in the sources of each equivalent classes reduces due to the removal of dead nodes and awareness of redirects. iUNA has a more general awareness of redirects while qUNA limits its redirect awareness to only six namespaces as explained in Section 2.2. In addition, iUNA takes variance in encoding into consideration.

This shows that despite the redundancy in the LOD cloud, the UNA holds in general. The blue bars on the right of Figures 3b and 3c indicate that there may be exceptions, for instance for datasets that integrate multiple other datasets.

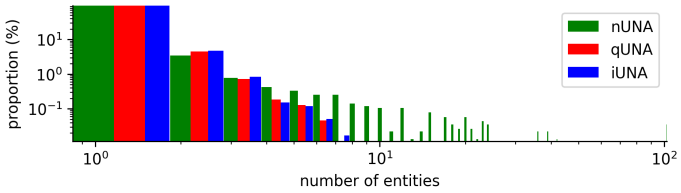
3.3 Reliability of the UNA

In this section, we focus on **RQ3:** Can the UNA give a reliable indication of identity errors in practice? As a baseline, we sample for each connected component G, $|V|$ different pairs at random. An estimate of the error (proportion of non-identical pairs) is between 0.47 and 0.68, depending on the interpretation of the unknown nodes.

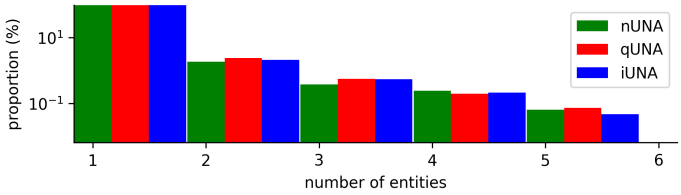
Next, we sample the same amount of pairs and test the error rate of the pairs that violate each of the three UNA definitions. The statistics in this section are the average of three runs. 61.79% of the sampled pairs violates the nUNA, the estimated error is between 33.31% and 49.89%. In contrast, an average of 49346.3 (41.23%) pairs violates the qUNA with an estimated error rate between 33.28% and 51.87%. Finally, on average, only 934.3 (0.78%) pairs on average violates iUNA, with the lowest error rate between 6.10% and 36.69%. In summary, the most amount of pairs violates nUNA. qUNA captures the least amount of pairs and has high error rate in comparison with iUNA.



(a) A histogram of the size of ECs in the gold standard.



(b) Frequency of entity counts in label-like sources



(c) Frequency of entity counts in comment-like sources

Fig. 3: Distribution of entities in equivalent classes

When using comment-like sources, the estimated error for the nUNA is between 33.07% and 46.77%. That of the qUNA has an error rate between 32.80% and 49.08%. Finally, on average 356 (0.30%) pairs violate the iUNA with an error rate between 8.89% and 21.18%. This shows that the pairs violating the iUNA have the lowest error for both label-like and comment-like sources. Finally, we observe that, for the iUNA, when using label-like sources, 2.62 times more pairs can be captured than that of comment-like sources.

The reliability of the iUNA also depends on the three exemptions. Figure 3b and 3c already took dead nodes into account. Next, we estimate the impact of the first two exemptions. Figure 1b presents a fictional example with redirect edges in red. The equivalence relation due to different encodings are captured by blue edges. There are in total 13,922 nodes in the graphs that capture redirect relations [6] Our experiment shows that 3,072 out of 8,394 entities were redirected.

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

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%. An example is in figure 1b where the entity `ex-es:ex:Bandon_(Oreg%C3%B3n)` and `ex:Bandon_(Oregón)` both redirects to `ex:Bandon,_(Oregon)`.

Next we study the equivalent entities suffering from different encodings. Figure 1b illustrates two examples: `Bandom%2C_Oregeon` and `ex:Bandon,_Oregon`; `ex:Bandon_(Oreg%C3%B3n)` and `ex:Bandon_(Oregón)`. We have 1,818 pairs of entities in the gold standard[7]. 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 differs 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 exemptions do not always hold, they are often useful.

4 Algorithm Design

The algorithm follows the intuition 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 pairs of entities violating the UNA. The "force" that holds the clusters together are the existing edges between the clusters, edges of redirection and the edges in the graph of equivalence under different encodings. The algorithm balances the two forces to identify a minimal amount of erroneous edges so that the two clusters can be told apart.

Computing an optimal cut whose removal makes the graph consistent within each CC is APX-hard [11]. We can, however, find sub-optimal solutions in polynomial time by using SMT methods [4]. We choose this approach for two reasons: to perform fast reasoning over relations of equality and inequality; and to obtain a sub-optimal solution for weighted constraint solving in polynomial time.

4.1 Algorithm using UNAs

Algorithm 1 takes the identity 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, it calls Algorithm 2 to test if the problem is beyond the capability of solving. If solved, the algorithm removes the edges identified (line 7), resulting in G'. Next, we obtain the graphs for each connected component of G_{ccs}. We obtain the corresponding subgraphs G^R_{cc}, G^E_{cc} from G^R, G^E respectively. G_{ccs}, together with G^R_{cc}, G^E_{cc} and the weighting scheme is then taken as input for

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

Algorithm 1: partition

```
1  Input: an identity graph G, a graph of redirect G^R, a graph of equivalence
     under various encodings G^E, a weighting scheme w
   Result: status s, a set of edges removed A, the graph of partitions G_P
2  initiate A as an empty set;
3  let status s, removed_edges A = partition_iter(G, G^R, G^E, w);
4  if s is error then
5      return ('error', ∅, G)
6  while |A| is not increasing (no new edge to remove) do
7      let H be the new graph of G with A removed;
8      obtain H_ccs, the graphs for each connected component of G' ;
9      foreach H_cc ∈ H_ccs do
10         obtain the corresponding subgraphs H^R_cc, H^E_cc from G^R, G^E
             respectively;
11         (s', A') = partition_iter(H_cc, H^R, H^E, w);
12         if s' is not 'error' then
13             A := A ∪ A'
14         else
15             return ('error', A)
16 remove A from G to get G_P;
17 return ('success', A, G_P).
```

Algorithm 2 (line 11). If successfully solved, the edges are collected in A. The algorithm stops when it encounters an error or has no more edges to remove.

In the while-loop of Algorithm 1 (line 11), there is a repeated call to Algorithm 2 that examines each graph of a connected component. More specifically, it takes advantage of an SMT solver for the reasoning over weighted relations of equivalence and returns a sub-optimal solution within a given time bound. Here, we use the minimum spanning forest F instead of the graph G_{cc} to reduce the load on the SMT solver. For an edge, or a pair of entities $(s,\ t)$, we introduce integer variable I_s and I_t respectively. A soft clause $c_{s,t}$ is $(I_s = T_t)$ combined with a weight (line 14 and 17) with reference to the weighting scheme w. The hard clauses are $(0 \leq I_s)$ and $(I_s \leq M)$ where M is the upper bound of the integer values. While not all soft clauses are 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. The SMT solver returns 'unknown' when it runs out of time. Otherwise, it returns a model where each each entity is assigned an integer. The edge $(s,\ t)$ remains if and only if I_s equals I_t in the model m.

In Algorithm 2, the use of a minimum spanning forest reduces the number of edges but removes some graph properties. To keep some original graph structure in F and to guide the back-propagation in SMT solving, we sample 15% of the edges B and add them to F. This is based on our observation that over 90%

Algorithm 2: partition_iter

```
1  Input: a graph of connected component G_cc, a graph of redirect G^R_cc, a graph
     of equivalence under various encodings G^E_cc, a weighting scheme w
   Result: s, a set of edges removed A_cc
2  obtain a set of pairs P violating the UNA;
3  if |P| ≤ 1 then
4      return ('success', ∅)
5  initiate an SMT solver o;
6  # hard clauses ;
7  foreach entity s in G_cc do
8      we introduce an integer variable I_s and assert hard clauses (0 ≤ I_s) and
         (I_s ≤ M)
9  # soft clauses ;
10 let F be the minimum spanning forest of G_cc;
11 randomly sample a small portion of B edges in G_cc and add to F;
12 obtain G'^R_cc the undirected graph of the (directed) graph G^R_cc;
13 foreach pair f = (s, t) in F ∪ P do
14     initiate a soft clause c_f according to w
15 foreach pair r = (s, t) in G'^R_cc ∪ G^E_cc do
16     if s and t are reachable then
17         initiate/update the weight of a soft clause c_r according to w
18 add all soft and hard clauses to o;
19 let s be the result of o after solving and m be the model (if any);
20 if status s is 'unknown' then
21     return ('error', ∅)
22 else
23     check each edge f of F, against the model m;
24     let A_cc be the set of all removed edges of F;
25     return ('success', A_cc).
```

of the edges are transitive closures between DBpedia's multilingual entities (see Section 6) for more details. We sample at most $|V_{cc}|$ pairs and filter out those do not violate UNA to form P to get soft clauses (line 9). If $P \leq 1$, no edge will be removed and the algorithm returns an empty set.

The weighting scheme w consists of a series of functions that map clauses to weights: $w = (f_G, f_R, f_E, f_P)$. 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 a weight of 5, the rest 0; f_R, f_E increases the weight by 1 respectively while f_P reduce the weight by 1. We used the training dataset to fine-tune the second weighting scheme w_2, we let f_G assign the clause of each edge in F a weight of 31, the rest 0. That of f_R and f_E are both 5 while f_P reduces the weight by 16.

5 Evaluation

5.1 Implementation

We used the *networkx* Python package[8] for the computation of the connected components, the Louvain algorithm [5], and the minimum spanning forests. For the manual annotation of the entities, we used ANNit[9]. We use Z3[10] as SMT solver [4]. We published all the code as an open source project[11]. All experiments were conducted on a 2.2 GHz Quad-Core i7 laptop with a 16GB memory running Mac OS. All reflexive edges were eliminated in preprocessing.

Based on our experience with Z3, the timeout for SMT solving was set to $|G_{cc}|/100 + 0.2$ second for each G_{cc}. In this setting, there is no timeout and the solver does not return 'unknown' for even the biggest graph in our experiment. The scalability is discussed in Section 6.

5.2 Evaluation Metrics

While precision and recall are commonly used in evaluation metrics [15], the presence of 'unknown' annotations makes them less suitable for this task since no edge involving entity of 'unknown' counts toward precision or recall. Take the results in figure 2e and 2f as example. The precision and recall are both 0 for our algorithm because all the edges removed involve some entities annotated "unknown". In contrast, the Louvain algorithm achieves a better precision and recall despite removing significantly more edges. This shows that precision and recall do not adequately capture the qualities we look for in a refinement algorithm.

We provide an additional metric, that we hope captures the desired qualities better. In its design, we focus on two properties that the equivalent classes should possess within the connected components resulting from refinement: (a) the equivalent class should not be separated over multiple components; (b) the equivalent class should not share the connected component it is in with other equivalent classes.

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 equivalent class, so that Q always represents the set of nodes within a given C that refer to the same entity e. V represents the total

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

[9] ANNit is a user-friendly interface for fast annotation of entities and triples. See [url and doi withheld for anonymity] for details.

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

[11] The code and implementation details are at [url and doi withheld for anonymity].

number of vertices in the graph, and O_e is the set of all entities in G' referring to e.

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.

5.3 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 4) against the Louvain algorithm. Table 1 presents the results of the average of three runs for each method. The Louvain algorithm removes a significant amount of edges in comparison with our algorithm. It has a higher recall but a lower precision. As for time efficiency, the Louvain Method completes processing both the training and evaluation sets within two minutes, while it takes around 23 minutes for our algorithms to finish. There are six CCs in the training set and and five in the evaluation set where where no edge is erroneous (except edges linking entities annotated as 'unknown' that can possibly be erroneous). While Louvain removes edges from all CCs, our algorithm removes no edges from 4.93 and 3.93 graphs respectively on average. In addition, we noticed that up to two CCs in the evaluation set can suffer from timeout using our algorithm. The precision drops significantly in the evaluation set. Our manual examination shows that some "harder" cases were distributed to the evaluation set including the example in figure 2a. It takes around 5 minutes to process the training set in contrast to around 17 minutes for the evaluation set.

Using qUNA removes fewer edges while using iUNA gives higher Ω but not significantly better. In more cases, using label-like sources gives better Ω than comment-like sources. We therefore take the settings of the best performance on Ω and $|A|$ for the next section.

5.4 Improving the Results

RQ4: Can we use additional information (weights over edges of the input graph) to improve the performance?

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

	training set				evaluation set			
	precision	recall	Ω	$\|A\|$	precision	recall	Ω	$\|A\|$
Louvain algorithm	0.020	**0.759**	0.084	39,302	0.039	**0.660**	0.083	43,642
qUNA-label-w_1	0.300	0.061	0.587	**14**	**0.417**	0.006	0.607	57
qUNA-label-w_2	0.237	0.083	0.618	88	0.167	0.004	0.576	53
qUNA-comment-w_1	**0.324**	0.031	0.595	**14**	0.244	0.004	0.562	**24**
qUNA-comment-w_2	0.236	0.104	0.614	91	0.199	0.021	0.591	79
iUNA-label-w_1	0.186	0.077	0.605	101	0.086	0.026	0.585	35
iUNA-label-w_2	0.168	0.108	**0.619**	262	0.065	0.016	**0.617**	175
iUNA-comment-w_1	0.187	0.053	0.609	91	0.146	0.009	0.575	42
iUNA-comment-w_2	0.084	0.003	0.618	114	0.072	0.026	0.610	130

Table 1: Comparison of the precision, recall, the Ω metric and the number of removed edges $|A|$ for the Louvain algorithm, and for our approach based on different UNA and provenance definitions.

	training set				evaluation set			
	precision	recall	Ω	$\|A\|$	precision	recall	Ω	$\|A\|$
iUNA-label-w_2	0.168	0.108	**0.619**	262	0.065	0.016	0.617	175
iUNA-label-w_2+weight	0.217	0.108	0.610	233	0.050	0.015	0.614	162
iUNA-label-w_2+disambiguation	0.221	0.135	0.615	264	0.098	**0.030**	**0.642**	191
qUNA-comment-w_1	0.324	0.031	0.595	**14**	**0.244**	0.004	0.562	**24**
qUNA-comment-w_1+weight	0.159	0.016	0.579	17	0.111	0.002	0.575	27
qUNA-comment-w_1+disambiguation	**0.412**	**0.163**	0.573	209	0.133	0.005	0.578	43

Table 2: Evaluation results with additional information

gold standard and across the entire identity graph. The bar-chart shows that most of the triples are associated with a weight less than 5.

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[12]. This can not only make the results less stable but also con-

[12] 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.

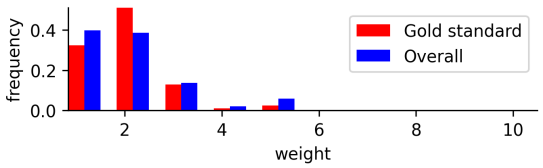


Fig. 4: Weight distribution of the owl:sameAs links in the LOD Laundromat.

firm 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 increase the weight of the corresponding soft clause by 2 when the weight of the edge is ≥ 2. 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. Table 2 shows that considering disambiguation entities may improve the Ω and the recall while only using the weight of the edges results in no consistent improvement.

6 Discussion and Future Work

RQ1 was answered by defining the iUNA for large, integrated graphs. For RQ2 and RQ3, we validated against our gold standard and compared the iUNA against the qUNA and the nUNA. We proposed and evaluated an algorithm (RQ4) and further studied the effects of edge weights and disambiguation nodes on the results (RQ5). Strictly speaking, our sample is still too small for an accurate estimate of the error rate of the entire identity graph. Using the gold standard, we found that among the 3,678 erroneous edges, only 5 have different label-like or comment-like sources. This indicates that UNA can be used for refinement but redundancy is not the direct cause of error. This is against the conclusion of [11] (type 2 error: consistency and conciseness error). This observation also explains the slow performance of the SMT solver: it has to perform sequences of back-tracking to "carry" inequality assertions through the edges.

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. We plan to study how our algorithm scales if given longer time, and compare it against other types of methods that were mentioned in Section 1.1 to provide a complete benchmark.

In the gold standard, 211,348 out of 232,311 edges (90.98%) are about DBpedia entities between different languages[13]. Our further analysis shows that among 3,678 erroneous edges, 3,029 involve DBpedia entities of different languages. This indicates that 82.35% of the errors come solely from relations between such entities. These links could be the result of automatic generation of transitive closure or inherited errors as DBpedia enriches. These edges have not only made the identity graph bigger (largest CC contains 177K entities), but also directly unusable as the errors propagate through the graph due to the transitivity of owl:sameAs. These experiments show the necessity of developing accurate and scalable refinement algorithms, and the need to be applied before consuming Linked Open Data in real-world scenarios.

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

References

[1] 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.

[2] Wouter Beek et al. "sameAs.cc: The Closure of 500M owl:sameAs Statements". English. In: *The Semantic Web - 15th International Conference, ESWC 2018, Proceedings*. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 15th International Conference on Extended Semantic Web Conference, ESWC 2018 ; Conference date: 03-06-2018 Through 07-06-2018. Springer/Verlag, 2018, pp. 65–80. ISBN: 9783319934167. DOI: 10.1007/978-3-319-93417-4_5.

[3] Tim Berners-Lee, Roy Fielding, Larry Masinter, et al. *Uniform resource identifiers (URI): Generic syntax*. 2005.

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

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

[6] 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.

[7] John Cuzzola, Ebrahim Bagheri, and Jelena Jovanovic. "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.

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

[9] 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.

[10] 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.

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

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

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

[14] Joe Raad et al. "Detecting erroneous identity links on the web using network metrics". In: *International semantic web conference*. Springer. 2018, pp. 391–407.

[15] 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.

[16] Andre Valdestilhas, Tommaso Soru, and Axel-Cyrille Ngonga Ngomo. "CEDAL: Time-Efficient Detection of Erroneous Links in Large-Scale Link Repositories". In: Aug. 2017. DOI: 10.1145/3106426.3106497.