NSFC-61261130588

ANR-12-IS02-002-01

Lindicle

Linked data interlinking in a cross-lingual environment
跨语言环境中语义链接关键技术研究
Liage des données dans un environnement interlingue

D5.2 QM Similarity computation

Coordinator: Yan Zhang
With contributions from: Juanzi Li, Zhigang Wang, Jérôme Euzenat

Quality reviewer:	Jérôme David
Reference:	Lindicle/D5.2/v4
Project:	Lindicle ANR-NSFC Joint project
Date:	April 3, 2017
Version:	4
State:	final
Destination:	public

Executive Summary

In Deliverable 5.1, we proposed a novel instance matching approach to deal with large scale linked data sets. To reduce the entities to match and reflect the interest of users, this approach postpones instance matching until answers to queries are returned from sources, hence the qualification of query-driven instance matching. The approach firstly computes local similarities between instances in query answers according to their identifying properties and, eventually, existing global similarity. It then propagates the similarities to connected resources. Optionally, user-feedback may be gathered to confirm resulting matches in the query answers. The resulting local similarities are integrated within a global similarity.

Compared with traditional approaches of instance matching, this approach provides the following advantages:

1. Instance matching is performed on the query results in each iteration, and there is no need to load the whole data set at a time.
2. We only get alignments between the instances which appear in the query results. The pay-as-you-go method not only lowers the cost of computation, but reflects users' interest.
3. We reduce the scale of the instances to be aligned by using user queries. So we can use more sophisticated algorithms (such as Similarity Flooding) to match them.

Here, we complete the approach description by detailing the procedure for updating the global QM-similarity based on the current local matching eventually completed by user feedback.

We also design several experiments to verify the effectiveness and the efficiency of our system. In particular, we compare the approach matching data sets globally or piecewise. We also evaluate the contribution of similarity propagation, necessity property filtering and user feedback.

The evaluation experiments compare speed, precision and recall on matching bibliographic data sets from OAEI 2009 with automatically generated queries.

The results first show that the approach is more practical than global matching in terms of speed. They also highlight that all the other features contribute to the quality of the results returned by the approach.

Document Information

Project number	ANR-NSFC Joint project	**Acronym**	Lindicle
Full Title	跨语言环境中语义链接关键技术研究 Linked data interlinking in a cross-lingual environment Liage des données dans un environnement interlingue		
Project URL	`http://lindicle.inrialpes.fr/`		
Document URL			

Deliverable	**Number**	5.2	**Title**	QM Similarity computation
Work Package	**Number**	5	**Title**	Query driven cross-lingual data interlinking

Date of Delivery	**Contractual**	M48	**Actual**	31-12-2016
Status	final		final ☒	
Nature	prototype ☐ report ☒ dissemination ☐			
Dissemination level	public ☒ consortium ☐			

Authors (Partner)	Juanzi Li, Zhigang Wang, Jérôme Euzenat			
Resp. Author	**Name**	Yan Zhang	**E-mail**	z-y14@mails.tsinghua.edu.cn
	Partner	Tsinghua U.		

Abstract (for dissemination)	This report completes the description of our query-driven instance matching approach by detailing how the global $QM - similarity$ is updated. We also design several experiments to test the effectiveness and the efficiency of the proposed solutions. Experimental results show that the query-drive approach is better than the traditional approach which gets the whole result at once. It also compares the various features of the proposed approach showing that they all contribute to better results.
Keywords	data interlinking, linked data, instance matching, query-driven matching, pay-as-you-go

Version Log			
Issue Date	**Rev No.**	**Author**	**Change**
2015-07-03	1	Yan Zhang	Input from QDIM
2016-09-08	2	Jérôme Euzenat	Added summary of D5.1
2017-03-20	3	Jérôme Euzenat	Collected whole content
2017-03-27	4	Jérôme Euzenat	Corrected typos; Added executive summary

Table of Contents

1. Introduction

INSTANCE matching is an important problem in the context of linked data. The query-driven approach to instance matching presented in Deliverable 5.1 aims at overcoming several drawback of existing approaches:

1. The link sets will not be available before the instance matching process is completed over the whole data sources.
2. Given the dynamics of linked open data sets, it is a waste of resources to match them to keep up.
3. Users usually only need links over a small part of the data sets.
4. Given the size of the data sets to match, it is impracticable to process accurate but resource-hungry algorithms on these.

According to the aforementioned problems, an incremental approach, in which instances will be matched as soon and only as they are needed, is naturally suited. Moreover, since such an approach can benefit from the availability of users eager to obtain results, a pay-as-you-go approach [Madhavan et al. 2007] seems possible.

The approach presented in Deliverable 5.1, processes queries against these data sets and matches the results of these queries. This leads to match smaller amount of data that are known to be comparable. To perform this local match the approach uses a global similarity that is accumulated over the successive query performed. It also take advantage of the identification of identifying properties which benefit from the knowledge accumulated through the successive iteration. Finally, the approach can more easily obtain feedback from users as the data is circumvented by their center of interest. This feedback is used to refine identifying properties and the results are aggregated in the global similarity. At any moment, instance alignments between the different data sets can be extracted from the global similarity.

Compared with traditional approaches of instance matching, this approach provides the following advantages:

1. Instance matching is performed on the query results in each iteration, and there is no need to load the whole data set at a time.
2. We only get alignments between the instances which appear in the query results. The pay-as-you-go method not only lowers the cost of computation, but reflects users' interest.
3. We reduce the scale of the instances to be aligned by using user queries. So we can use more sophisticated algorithms (such as Similarity Flooding) to match them.

This report completes the presentation of the query-driven instance matching approach of Deliverable 5.1 by describing how is the global $QM - similarity$ computed. It then provides an evaluation of the various components of the approach.

After recalling the approach (Chapter 2) taken for query-driven instance matching, we detail the methods used to compute global similarity (Chapter 3). Finally, Chapter 4 presents experimental results before concluding (Chapter 5).

2. Summary of the query-driven instance matching approach

We recall here the approach proposed in Deliverable 5.1, were the full description can be found.

The principles of query-driven instance matching can be summarized as follows:

1. A local similarity is established across query answers over the same data sets;
2. This local similarity is based on (a) previously established global similarity, (b) identifying and necessary properties;
3. The local similarity obtained for each pairs of answers is aggregated in a global similarity and propagated through similarity flooding;
4. Identifying and necessary properties used to compute local matching are based on statistical measures, first of property value cardinality and later on instance co-occurence in query answers;
5. Optionally, user-feedback may be used in order to confirm obtained matches on query answers.

Query-driven instance matching aims at finding simultaneously and progressively link sets across several data sources $(d_1, \dots)$ accessible through SPARQL queries $(q_1, \dots q_t)$. Hence it will match query results $(g_{1,t}, \dots)$ instead of whole data sets by taking advantage of the links already found (σ^t_{QM}) in order to improve the link set (σ^{t+1}_{QM}). At any moment, it is possible to extract a link set L_t from σ^t_{QM}.

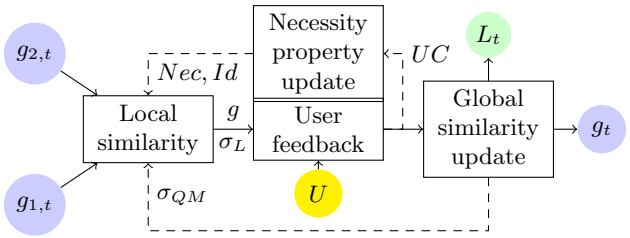


Figure 2.1: Precise view of query-driven instance matching. Dashed lines present the update process whose result is used at the following iteration.

More precisely, at each new query q_t, each pair of answers $g_{k,t}, g_{l,t}$ corresponding to $g_{k,t} = eval(q_t, d_k)$ and $g_{l,t} = eval(q_t, d_l)$ are compared and a local similarity σ_L is computed between the resources appearing in these answers.

From all matches between all the answers of query q_t is computed a global similarity σ_{QM} which is obtained by aggregation of the local matches to the global similarity obtained from the previous queries $(q_1, \dots q_{t-1})$. This similarity is used to extract a match M between them (which can be though of as a local link set) used to build the result g_t.

Optionally, feedback from users can be sought.

Our approach is executed iteratively, and one iteration can be separated into three steps, namely query, local similarity calculation and global similarity calculation. At each iteration, we firstly perform a query on the source ontologies. Then we calculate the local similarity based on the query results. Finally, we use the local results to update the global similarity and get the alignments according to it.

The overall algorithm is presented in Deliverable 5.1. In the next chapter, we first fill the gap left open in this deliverable by answering to the question "How is σ_{QM} computed?".

3. QM-similarity

Briefly speaking, the global query-driven similarity, or *QM-similarity*, is computed by combining the normalised google distance and measuring the ratio of matches between two resources with respect to their number of co-occurence. We first provide a precise definition for these concepts (Section 3.1), illustrate it by an examples (Section 3.2) and finally provide the actual algorithm (Section 3.3).

3.1 Definition

Intuitively, if i matches j, they would co-occur more in answers and possibly be matched when they co-occur in the answers. Based on this consideration, the QM-Similarity is introduced to measure the similarity between instances in the system, which is described as follows:

$$\sigma_{QM}(i,j) = \sigma_{NQD}(i,j) \times \pi^m_{co}(i,j)$$

where

$$\sigma_{NQD}(i,j) = \frac{1}{1+NQD(i,j)}$$

$$NQD(i,j) = \frac{max(\log(n(i)),\log(n(j))) - \log(n^{co}(i,j))}{\log(n) - min(log(n(i)),log(n(j)))}$$

$$\pi^m_{co}(i,j) = \frac{n^m(i,j)}{n^{co}(i,j)}$$

The n in the NQD formula denotes the total number of queries submitted to the integration system; $n(i)$ denotes the number of queriescontaining the instance i in answers so far. $n^{co}(i,j)$ denotes the number of queries in which i and j co-occur and $n^m(i,j)$ denotes the number of queries in which i and j are matched.

$\pi^m_{co}(i,j)$ is the probability of i and j matching given that they co-occur. $NQD(i,j)$ is similar to the Normalized Google Distance [Gligorov et al. 2007] except that the number of pages containing a term is replaced by the numbers of queries containing an instance.

3.2 Example

Consider the example shown in Figure 2 in Deliverable 5.1. Before the query *The profile and authors of "Ontology Matching"* is evaluated, we assume that the values for the various indicator are as in column 100 of Table 3.1. They will be updated after the query of according to column 101.

Indeed, according to the above formulas:

$NQD(per-1, 1\_per) = \frac{\log(11)-\log(7)}{\log(101)-\log(7)} = 0.168$

$\sigma_{QM}(per-1, 1\_per) = \frac{1}{(1+NQD(per-1,1\_per))} \times \frac{6}{7} = 0.734$

$\sigma_{QM}(pub-1, 1\_per) = \frac{1}{(1+NQD(pub-1,1\_per))} \times \frac{0}{7} = 0$

Because of the alignment between $per-1$ and 1_per in the query result, the QM-similarity $\sigma_{NQD}(pub-1, 1\_per)$ is increased from 0.71 to 0.73. On the other hand, $pub-1$ and 1_per are not matched, so $\sigma_{QM}(pub-1, 1\_per)$ remains equal to 0.

n	100	101
$n(per-1)$	10	11
$n(1\_per)$	6	7
$n(pub-1)$	15	16
$n^{co}(per-1, 1\_per)$	6	7
$n^{m}(per-1, 1\_per)$	5	6
$n^{co}(pub-1, 1\_per)$	6	7
$n^{m}(pub-1, 1\_per)$	0	0
$\sigma_{QM}(per-1, 1\_per)$	.71	.734
$\sigma_{QM}(pub-1, 1\_per)$	0.	0.

Table 3.1: Update of the global similarity σ_{QM}.

3.3 Algorithm

Algorithm 1 takes advantage of data structures for storing similarities:

- $R_{IM}[i,j]$ contains $\sigma_{QM}(i,j)$;
- $n^{co}[i,j]$ contains $n^{co}(i,j)$;
- $n^{m}[i,j]$ contains $n^{m}(i,j)$;

Other notations are described in Table **??**.

Algorithm 1 Computation of the global similarity

```
1: procedure AggregateQM(g, g', M)
Input: g, g': two graph answers
Input: M: matching instances
Output: R_IM: updated similarity structure
2:     for all i ∈ nodes(g) do
3:         for all j ∈ nodes(g') do
4:             n^co[i,j] := n^co[i,j] + 1
5:             if ⟨i,j⟩ ∈ M then
6:                 n^m[i,j] := n^m[i,j] + 1
7:             end if
8:             R_IM[i,j] := σ_QM(i,j)
9:         end for
10:    end for
11:    return R_IM
12: end procedure
```

4. Evaluation

In this chapter, we describe experiments for evaluating the proposed approach and analyse the results in detail.

4.1 Experimental setup

There are instance matching tracks are OAEI. However, they do not involve user queries. The OAEI interactive track involves user feedback, but concerns ontology matching and the feedback is requested by the systems, it is not driven by users through their queries. Hence it is rather difficult to compare with OAEI results. We still take advantage of OAEI datasets (§4.1.1) and references (§4.1.3), but introduce query generation (§4.1.2), and thus do not compare with other systems as mandated by OAEI rules. Instead, we compare components of the proposed solutions (§4.1.4) with usual measures (§4.1.5).

4.1.1 Datasets

We use the AKT-Rexa-DBLP data sets from the OAEI instance matching campaign[1] for the evaluations. It includes three data sets, namely eprints, Rexa and SWETO-DBLP.

1) The eprints data set comes from AKT (Advanced knowledge Technologies) research project and extract the instances using a HTML wrapper from the source website. It is the smallest one of the three data sets.

2) The Rexa data set is extracted from the search results of the search server on `http://www.rexa.info/`.

3) The SwetoDblp data set is a large-size data set focused on bibliography data of Computer Science publications where the main data source is DBLP [**alemanmeza2007a**]. It was created from a XML dump of DBLP and other data sets that are used to add relationships to other entities such as Publishers, Companies and Universities.

Table 4.1 shows the number of instances in these data sets.

Data set	#Instances	#Blank Nodes
Eprints	847	283
Rexa	14771	3721
Sweto-Dblp	1642945	1007887

Table 4.1: Sizes of the 3 data sets.

The three data sets use the same schema, the Opus schema, to represent the data in RDF. This allows to specifically evaluate the query-driven matching approach and not the ontology alignment. The instances in the datasets can be classified into two kinds, *authors* and *documents*.

4.1.2 Query Construction

In the experiment, three instance matching tasks are constructed which are Eprints-Rexa (correspondence between answers from these two data sets), Eprints-Dblp and Rexa-Dblp.

[1] http://oaei.ontologymatching.org/2009/instances/

To imitate user queries, we built 404 SPARQL queries described according to instances of the Eprints data set (because it is the smallest of the three). We use the **CONSTRUCT** query form to get the single RDF graph specified by the graph template, which will be the input of the matching system (see Figure**??**). These queries can be divided into three sets: 1) query an author's documents; 2) query a document's authors; 3) query all the co-authors of an author. We give an example of each query type.

1. Query to discover an author's documents

```
CONSTRUCT {
   ?document opus:author ?author.
   ?document rdf:type ?type.
   ?document rdfs:label ?label.
   ?author rdf:type ?type1.
   ?author foaf:name ?name.
} WHERE {
   ?document opus:author ?authors.
   ?document rdf:type ?type.
   ?document rdfs:label ?label.
   ?authors rdfs:member ?author.
   ?author rdf:type ?type1.
   ?author foaf:name ?name.
   FILTER(regex(?name,"*Nordlander*")
    && regex(?name,"*Tomas*")).
}
```

2. Query to find a document's authors

```
CONSTRUCT {
   ?document opus:author ?author.
   ?document rdf:type ?type.
   ?document rdfs:label ?label.
   ?author rdf:type ?type1.
   ?author foaf:name ?name.
} WHERE {
   ?document opus:author ?authors.
   ?document rdf:type ?type.
   ?document rdfs:label ?label.
   ?authors rdfs:member ?author.
   ?author rdf:type ?type1.
   ?author foaf:name ?name.
   FILTER( regex(?label, "AQUA: \\
    An Ontology-Driven \\
    Question Answering System.")).
}
```

3. Query to obtain all the co-authors of an author

```
CONSTRUCT {
   ?document opus:author ?author1.
   ?document opus:author ?author2.
   ?document rdf:type ?type.
```

```
   ?document rdfs:label ?label.
   ?author1 rdf:type ?type1.
   ?author1 foaf:name ?name1.
   ?author2 rdf:type ?type2.
   ?author2 foaf:name ?name2.
} WHERE {
   ?document opus:author ?authors.
   ?document rdf:type ?type.
   ?document rdfs:label ?label.
   ?authors rdfs:member ?author1.
   ?author1 rdf:type ?type1.
   ?author1 foaf:name ?name1.
   ?authors rdfs:member ?author2.
   ?author2 rdf:type ?type2.
   ?author2 foaf:name ?name2.
   FILTER (regex(?name1,"*Nordlander*")
    && regex(?name1,"*Tomas*")
    && ?author1 != ?author2).
}
```

Measure	Eprints	Rexa	Dblp
min.	0	0	0
avg.	19.7	26	56.9
max.	152	192	435
instances	847	14771	1642945
avg./inst.	2×10^{-2}	1.8^{-3}	3.46×10^{-5}

Table 4.2: The instance numbers for query answers.

Table 4.2 shows the average, maximum and minimum numbers of instances for the query answers. The *instances* measure denotes the number of instances in the data sets. We can notice that the size of query answers are much lower than their source ontologies.

4.1.3 Reference Generation and Evaluation

To evaluate the overall effectiveness of our approach, we evaluated these queries one by one and segmented the whole process into three phases. In each phase, one third of the constructed queries are involved. At the end of each phase, evaluations are performed on the results of the phase according to the references extracted from the Eprints-Rexa-Dblp gold standard. Thus, we can identify the improvement introduced by our methods.

To evaluate the approaches in different phases, we build the reference according to the gold standard provided by OAEI. In order to ensure consistency of the reference in different phases of our query-driven approach, we extract the mappings whose *entity1* appears in the matching result of the query-driven approach at the end of phase 1. Algorithm 2 shows the specific steps of reference generation.

Algorithm 2 Reference generation

```
1: procedure ReferenceGeneration(M_ref, M_s)
Input: M_ref: the matching pairs in the golden standard
Input: M_s: the matching result of phase s
Output: M_exp: the reference to use in phase s
2:     M_exp := ∅
3:     for all (i_p, i_q) ∈ M_i do
4:         for all (i_p, i) ∈ M_ref do
5:             M_exp := M_exp ∪ {(i_p, i)}
6:         end for
7:     end for
8:     return M_exp
9: end procedure
```

4.1.4 Approaches

We implement the following approaches from the solution in Chapter 2 to verify its validity. The thresholds in these approaches are obtained based on observing matching results.

- **IProp:** The approach without Similarity Flooding (at Step 2 of the solution in Section 3.3 of Deliverable 5.1), necessary property filtering (at Step 4) and user feedback (at Step 6).
- **IProp-F:** The IProp approach with Similarity Flooding.
- **IProp-U:** The IProp approach with user feedback.
- **IProp-FU:** The Iprop-F approach with user feedback.
- **Iprop-N:** The Iprop approach with Necessity property filtering.
- **Iprop-FN:** The Iprop-F approach with Necessity property filtering.

We implemented all solutions in Java and experiments are run on a sever with 8-core Intel Xeon E5-2680 processor(@2.7Ghz) and 32GB memory. The operating system is Ubuntu 14.04.

4.1.5 Performance Metrics

We use *Precision*, *Recall* and *F1-Measure* to measure the performance of our proposed solution. They are defined as usual: *Precision* (P) is the percentage of correct discovered matches in all discovered matches; *Recall* (R) is the percentage of correct discovered matches in all correct matches; *F1-Measure* (F1) is the harmonic means of precision and recall.

$$F1 = \frac{2}{1/R + 1/P}$$

4.2 Comparison with global matching

This part compare the IProp and IProp-F approaches with the solution named Global which utilizes the same similarity computing method as IProp to obtain all the correspondences

between the ontologies. The Global approach uses the same configuration, e.g., filter threshold, as IProp. Due to memory constraints, the Global approach cannot run the Similarity Flooding algorithm.

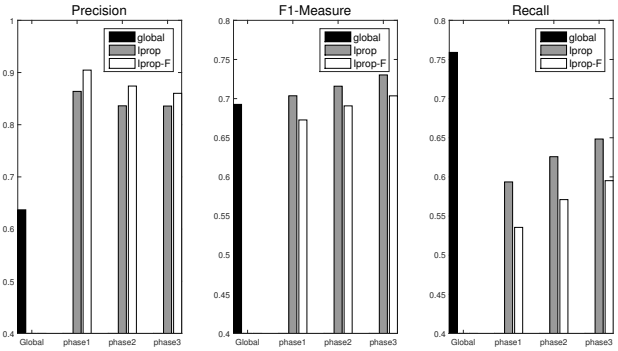


Figure 4.1: Eprints-Rexa.

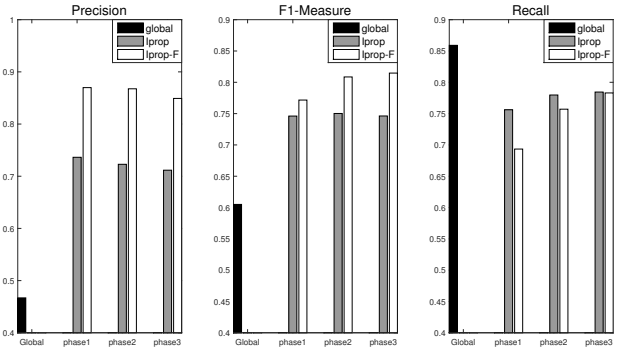


Figure 4.2: Eprints-Dblp.

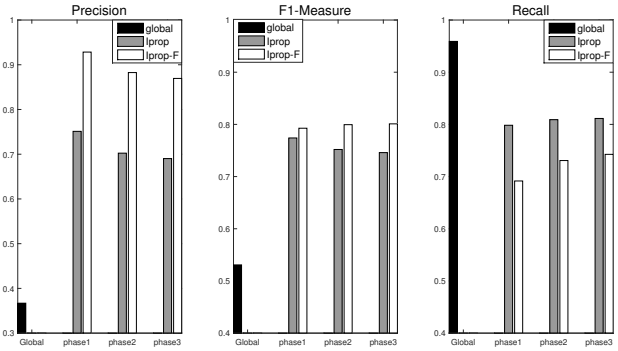


Figure 4.3: Rexa-Dblp.

Figure 4.1, 4.2, 4.3 and Table 4.3 show the comparison between IProp, IPropF and Global in Precision, Recall, F1-Measure and elapsed time respectively. From these results, we can observe that:

1. The elapsed timed of the query-driven approach (IProp) is much less and the F1-Measure is higher than the Global's in all three datasets, IProp improve the efficiency and effectiveness significantly.

2. Considering F1-Measure values in different phases of IProp and IProp-F, with more and more queries being posted into the system, the whole performance of the query-driven approaches increases.

3. The recall of Global is always higher than that of IProp since some matching pairs do not appear together in query results.

4. The precision of IProp decreases over phases. This is because more matching pairs are added into results incrementally as more and more queries are being executed.

5. The precision of IProp-F in Figure 4.1, 4.2 and 4.3 is higher than IProp, but recall is always less. This is because Similarity Flooding will modify the similarities by propagating them among the instances, and then obtain more accurate results. The results of the three experiments show that the effectiveness of Similarity Flooding depends on the specific data set. IProp-F get a better performance only on the Eprints-Dblp data set.

Task	Global	IProp	IProp-F
Eprints-Rexa	89.7	6.4	8.2
Eprints-Dblp	8757.8	12.3	15.6
Rexa-Dblp	310255.4	21.56	26.5

Table 4.3: The elapsed time of the compared methods(s).

4.3 User feedback

In this part, we compare IProp with IProp-U and IProp-F with IProp-FU to evaluate the efficiency of user feedback. In IProp-U and IProp-FU, after the QM-similarities are calculated, only the elements with the maximum similarity among the current matching results are recommended to the user. Table 4.4 reveals the precision of the recommended matches. The results indicate the accuracy of the proposed approaches (the overall precision reaches 0.93). Figure 4.4, 4.5 and 4.6 show the comparative results. From these figures, we can note that in all the comparisons, the approaches with user feedback are higher in precision and F1-Measure and equivalent in recall to the non-feedback counterparts.

Approach	Eprints-Rexa	Eprints-Dblp	Rexa-Dblp	avg.
IProp	0.96	0.92	0.95	0.94
IProp-F	0.89	0.89	0.96	0.92
avg.	0.93	0.91	0.96	0.93

Table 4.4: The precision of the recommended matches.

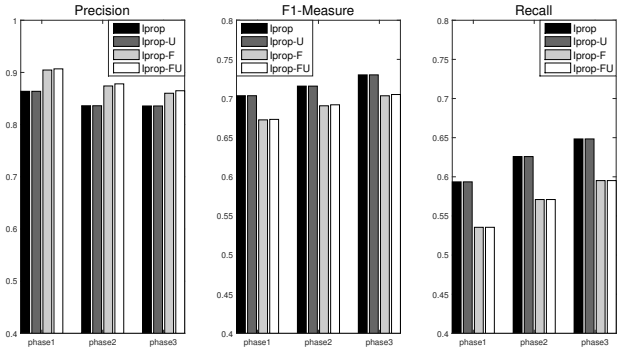


Figure 4.4: Eprints-Rexa with user feedback.

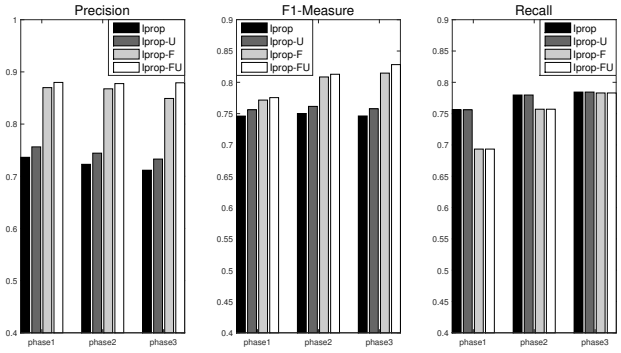


Figure 4.5: Eprints-Dblp with user feedback.

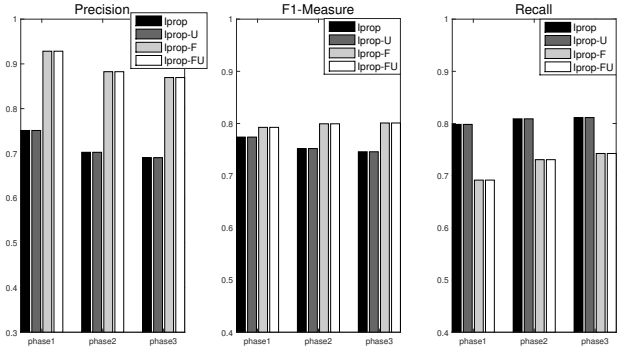


Figure 4.6: Rexa-Dblp with user feedback.

4.4 Identifying and necessity properties

As mentioned before, in the experimental data sets, there are two kinds of instances: the authors typed by the *foaf:Person* class and the documents. The instances of *foaf:Person* have the property *foaf:name*. According to the method in Chapter 4 of Deliverable 5.1, the identifying degree d_i^p of this property is 0.99. Therefore, it can be regarded as an identifying property of *foaf:Person*. For the document classes existing in all the three ontologies, Table 4.5 shows the d_i^p values of their properties where the *Article_P* denotes the

class *Article_in_Proceedings*. From this table, the *rdfs:label* property can be selected as an identifying property of both classes.

Because only one single valid property is contained in the *foaf:Person* class, we merely consider the necessity property in the document classes. Table 4.6 shows the necessity degrees of the properties in the aforementioned document classes calculated based on the user confirmed matches generated by the IProp-U approach on all the queries. Similar results can be obtained through the IProp-FU approach. Therefore, in this experiment, the *opus:pages* property is chosen as the necessity property in IProp-N and IProp-FN approaches.

Figure 4.7, 4.8, and 4.9 give a comparison between IProp and IProp-N and IProp-F and IProp-FN respectively to show the effectiveness of the necessity properties. By comparing the results, the approaches can accurately filter out the match errors, through the necessity properties, and thus improve the precision, nearly retain the recall and consequently raise their F1-Measure. It is because the necessity degree of the property *opus:pages* is high enough (0.98 for Article_P and 1 for Article), this filter process can hardly reduce the recall.

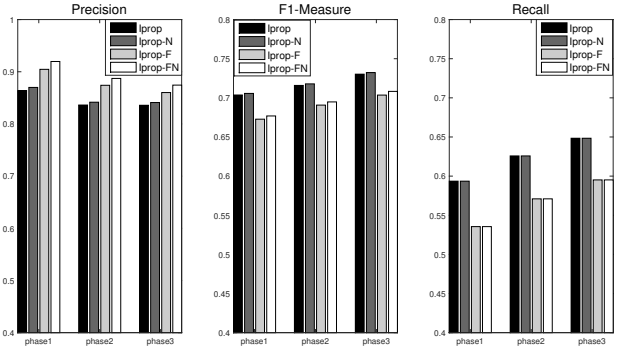


Figure 4.7: Eprints-Rexa with necessity properties filter.

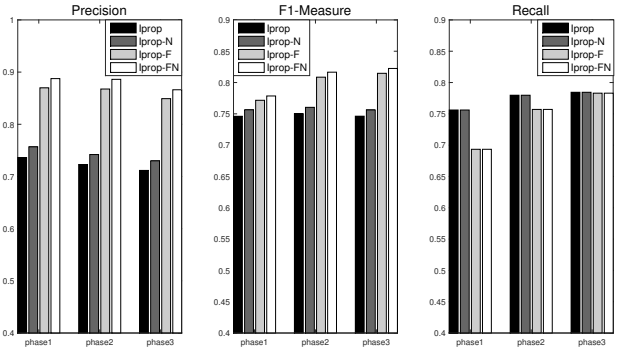


Figure 4.8: Eprints-Dblp with necessity properties filter.

Class	label	pages	book_title	year	volumn
Article_P	0.99	0.07	0.01	0.00	/
Article	0.98	0.13	/	0.00	0.00

Table 4.5: Identifying values of the properties of the document classes.

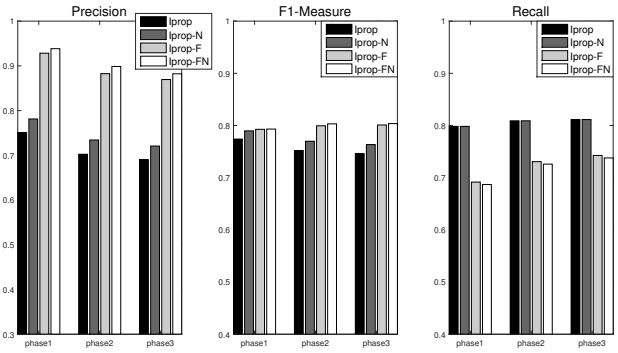


Figure 4.9: Rexa-Dblp with necessity properties filter.

Class	label	pages	book_title	year	volumn
Article_P	0.86	0.98	0.35	0.98	/
Article	0.91	1.0	/	0.95	0.70

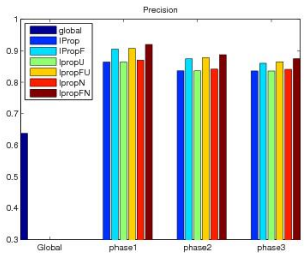
Table 4.6: Necessity values of the properties of the document classes based on the IProp-U.

5. Conclusion

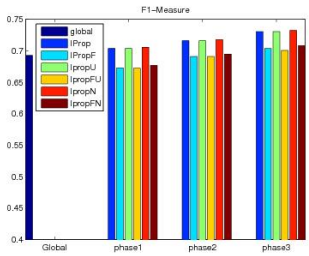
In Deliverable 5.1, we have proposed a novel instance matching approach to improve the performance of larger scale instance matching in semantic integration systems. This approach iteratively elaborate a global similarity between data sources by taking advantage of relatively small query answers matched against each others and user feed back of the resulting match. The iteratively built similarity can be use to export, at any time, an instance alignment across data sets.

Here we further presented the way the global $QM-similarity$ is computed. We evaluated the proposed method against OAEI instance matching data sets. The merits of the various presented strategies have been assessed and their benefits are clearly highlighted.

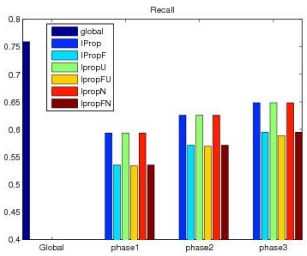
A. Experiment results in one diagram



Precision

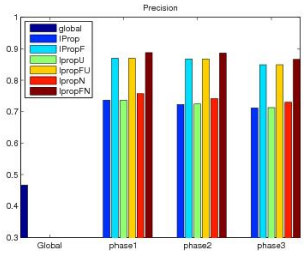


F-measure

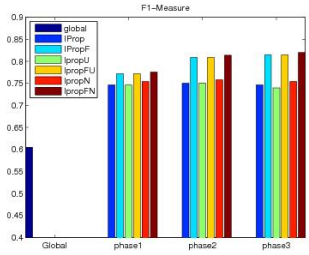


Recall

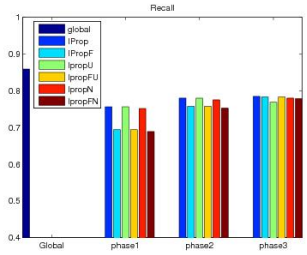
A.1. Eprints-Rexa.



Precision

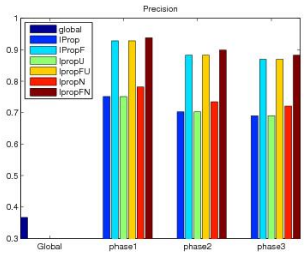


F-measure

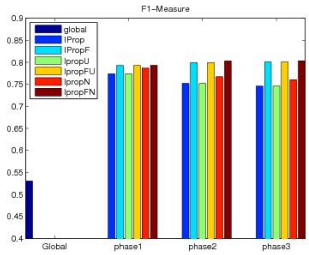


Recall

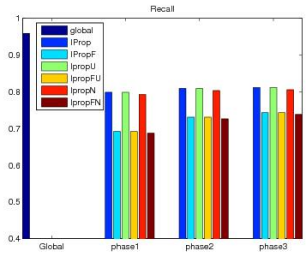
A.2. Eprints-Dblp.



Precision



F-measure



Recall

A.3. Rexa-Dblp.

Bibliography

Gligorov, Risto, Warner ten Kate, Zharko Aleksovski, and Frank Van Harmelen (2007). "Using Google distance to weight approximate ontology matches". In: *Proc. 16th international World Wide Web conference (WWW)*. ACM, pp. 767–776 (cit. on p. 7).

Madhavan, Jayant, S Jeffery, Shirley Cohen, Xin Dong, David Ko, Cong Yu, and Alon Halevy (2007). "Web-scale data integration: You can only afford to pay as you go". In: *Proc. 3rd Conference on Innovative Data Systems Research (CIDR), Asilomar (CA US)*, pp. 342–350 (cit. on p. 5).