Refined Ontology Matching Methods for Special Data Integration

Dénes Paczolay, András Bánhalmi, Ádám Zoltán Végh, Gábor Antal, Vilmos Bilicki
Department of Software Engineering
University of Szeged
Szeged, Hungary
{pdenes,banhalmi,azvegh,antalg,bilickiv}@inf.u-szeged.hu

***Abstract*— Ontology matching is an important area of research, since it has many applications like semantic webs, information extraction, data mining and reasoning. In most cases, the matching is done between thesauri of hierarchical concepts made for similar domains by different groups. A methodically similar, but technically different task is when ontology matching is used for system integration applied to generated ontologies. For data integration, we created a framework that generates ontologies got from a database schema or from the source code itself, these being called local and global ontologies. To complete the data integration process, only ontology matching has to be done semi-automatically; all the other tasks are carried out automatically. However, the generated ontologies have some special features, namely mixed languages in the name, special abbreviations, and special structures in the names generated. For ontologies like these, the common matching methods which have the best performance on average when these are applied to other tasks, perform much worse in the case of integration. In order to improve the accuracy, we propose novel similarity measuring and ontology matching methods.**

Keywords- semantic knowledge representation; local ontology generation; ontology matching; ontology alignment; ontology; Java to ontology; similarity measure

I. INTRODUCTION

In the past few decades, many ontologies have been created to describe semantically different specific domains. The ontologies created for a similar domain by different groups contain the same or similar concepts, and connections between them in most cases. The fact that there are many similar ontologies implied the need for querying all the ontologies at the same time with the same expression. For this, many research groups proposed to create a mediator schema, a global ontology, which integrates the necessary concepts got from all the so-called local ontologies. In addition, for the purpose of creating a broader semantic search, not only ontologies and RDF (resource description framework) stores, but information from conventional RDBM (relational database management) systems and (nowadays) NoSQL (no structured query language) data stores are also involved. The aim of system integration is similar in some sense: applications with a similar functionality should be able to exchange their data with each other. This task can be resolved by hardcoding the data transformation and calling the services of each other, but this solution is time-consuming, the productivity of development is low, and the reengineering is difficult. For this reason, the concept of semantic integration can be used to integrate the data of the systems with a similar functionality. Some methods and techniques have already been proposed for data integration using ontologies. These solutions may contain one mediator schema only, which integrates all the databases virtually, or contain more ontologies from which one mirrors the concept of a database, the other ontology being responsible for the semantic integration. These are called local and global ontologies. In our system integration framework, the local ontologies are generated using semantic information collected from the database schema itself. The global ontology may also be a generated one; moreover, in one of our use cases it is generated from the source code of the integrating central application itself. The principal problem, however, is that a more robust ontology matching method is needed among the concepts of these specially generated ontologies for the purpose of more efficient and quick data integration development. For the ontology matching problem, many methods have been proposed and implemented, but these methods are too general for handling special issues concerning the rules of the ontology generation and frequent naming cases. To create a more robust ontology matching process, we propose new similarity measures for generated ontologies, and carry out some experiments that apply them in real-world tasks to evaluate their precision.

In the sections below, a detailed overview of related work is provided. Then, we introduce and discuss the ontology generation methods, and many type of problems are examined that are related to ontology matching. After, some similarity methods are proposed, which can improve the performance of ontology matching for generated ontologies. In our experiments, real-world use cases are introduced, and the results of the proposed methods are then compared with those got using the known matching techniques. Lastly, we discuss our experimental results, and make some suggestions for future study.

II. RELATED WORK

The chief goal of ontology matching is to collect all the knowledge or information related to a common domain. Because many ontologies were created for a similar domain, an integration method was needed to collect interesting data from them. Integration can be achieved by finding relations among the concepts [1][2]. What these solutions have in common [3]–[6] is that in general, an iterative matching process is performed: after each iteration the result of a matching process can be the input for the next matching process. Also, the results of the matching processes are aggregated. This mechanism can be arranged hierarchically to implement a more complex ontology matching. The basic concept similarity measure used by ontology matching methods is to compare the strings describing the concepts. For this task string matching methods are used [5][7][8], perhaps completed with some additional, but sometimes important complex natural language processing (NLP) methods (translation dictionaries, wordnets, Tf/Idf (term frequency / inverse document frequency) categorization, etc.) [3][9][10]. Besides language processing methods and string matching, many graph-structure based investigations have been proposed to aid the matching process [11][12].

In addition to the above-mentioned ontology merging tasks, other applications also apply the so-called ontologies to describe semantic information about specified objects, and in this way achieve an integrated result. In this area, the integration of Web services should be mentioned [13], and data (or system) integration is also an important application domain [14][15]. In the latter (which is the focus of this paper), the ontologies for a mediation of local and global information are widely used, but we do not have any information about ontology matching methods, which have been developed and focus on generated ontologies, considering, for example, the generation rules, and the naming variations in practice. The widely-used heterogeneous benchmark, which is used for Ontology Alignment Evaluation Initiative (OAEI) [16], does not contain ontologies of this kind [2].

III. USE CASES

An industrial partner suggested that they would like to change their POS printing system to a new one, because their original system had become overly complicated, so adding new clients to the system was hard to implement. To support easy system integration, an ontology-based integration framework was developed. By using different open-source tools with some important modifications added by us, and implementing new modules (e.g., for query rewriting), a well-functioning tool chain was developed. Using this technique for integration, only one task needed human assistance, namely to find the relations among the concepts of the local and the global ontologies. The local ontology was always generated corresponding to the schema of the database. The global ontology was constructed by humans in the "POS Printing" use case, or it was generated from Java source code for the "Event Integration" use case.

A. *POS Printing Data Integration*

The starting point in this application is a PostgreSQL relational database containing about 70 tables to describe products, department stores, promotions, printing templates, etc. The endpoint is a global ontology assembled by two people, which covers the topics of products, promotions and printing. This global ontology was created in two languages, namely English and Hungarian. The local ontology vocabulary was generated from the schema of a relational

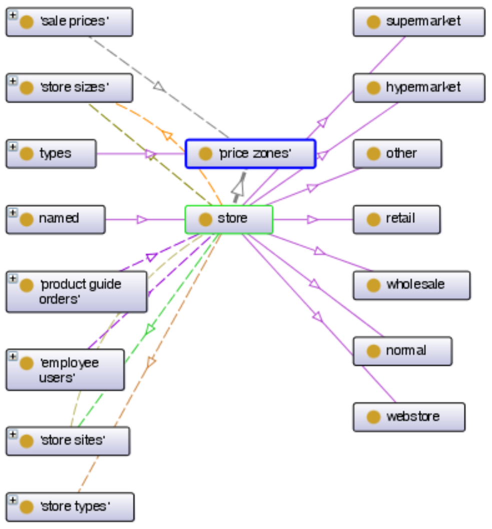

Figure 1. A snippet from the global ontology created for the POS printing use case.

database using the D2RQ generator [17] with some minor modifications which are described here [19]. Fig. 1 illustrates a little snippet of the global ontology created for this use case.

B. *Calendars and events*

In another use case for system integration, we chose calendar integration. This means that it should be possible to collect event-like data got from various applications for the Google Calendar. For this, we searched scheduling, time/event tables, and booking applications with database support on websites such as SourceForge, Google code and Hot Scripts. In the end, we collected 16 applications with their sources. The selected programs were very diverse; some of them used 2-3 tables only, and others had over 15 tables. From the databases, the local ontologies were generated using a modified D2RQ generator [19].

The global ontology for describing the concepts in the Google Calendar was generated from the Java source code of Google Calendar Client API. For this purpose, a Java2RDF application was developed. This application uses QDox [18] to explore class, interface and method definitions.

IV. METHODS

Here, some novel methods are proposed that focus on the naming rules of the generation process and some other special naming features of attributes in databases. These methods are implemented in the COMA CE framework. It is an open source [20] ontology and schema matching tool, and gave good results on OAEI evaluation campaigns [21]. Using the framework of COMA, first the proposed similarity measures were implemented, then complex matchers aggregated these measures; and at the top, "workflows" were defined. COMA supports the use of synonyms and abbreviations, but it is not transparent how the built-in methods use them. Hence, a new implementation was carried out in the framework that allows one to use some features for computing the proposed similarity measures, such as graph parts (parents, children), data types, synonyms and abbreviations. In the following, the proposed methods will be grouped in terms of their application, and later the experiments used to test their performance will also be grouped in the same way.

A. POS Printing Data Integration

In this case, two kinds of methods are proposed for improving the accuracy of ontology alignment.

1. **STRAT1**: In this similarity measure method, the rules of ontology generation are taken into account. The similarity between any two concepts is calculated by separating the parent class names. This can be done easily, because predefined delimiters were used at the naming using class and attribute names in the ontology generation process. After this separation, when comparing concepts, we will only use the attribute name, and a normal string matching method is applied (edit distance based). In addition, this method and the others as well, contain a type comparison related to ontology: when different typed concepts are compared (class to not class, data type property to not data type property) then only a predefined minimal score is given as the similarity value. It should be mentioned here that this similarity measure is suitable only for concept names in the same language. Hence, our algorithm is the following:
   - Compare the type of the concepts (class, properties)
     - if the types don’t match, then a predefined low constant is the similarity value
   - Create the child names using the predefined delimiters
   - Use string matching method (Edit Distance-based) to determine the similarity value
2. **STRAT2:** This case handles issues when the names of the database tables or attributes do not follow the conventions, so the names can be written in a mixed language perhaps without any conventional separation like uppercase or delimiters. For example, “akciotype” or “aruhaztype” may be mentioned, which mean ‘sale type’ and ‘store type’. Here we solve the problem of accents too, for the Hungarian case. This method contains the following steps:
   - First, compare the type of the concepts (class, properties)
   - Second, the words of global ontology are converted into their non-accented form.
   - Third, the words of a concept in the global ontology written in different languages are collected in a set. However, predefined stop words are not considered.
   - Fourth, the multilingual word set of the global ontology is fitted to the local ontology names, with some restrictions. Only those matched characters are summed for which the length of fitting is at least 2. One character length fitting is not considered.

   The next pseudo code snippet tries to illustrate the essence of this kind of similarity:

```
function getSimilarity (concept1, concept2)
if type(concept1)!=type(concept2)
        return lowValue
name1=toNonAccent(getNames (concept1))
name2=toNonAccent(getNamesMultiLanguage(concept2))
tokens2=tokeniseWithoutStopwords(name2)
tokens2=addSynonyms(tokens2)
foreach T2 in tokens2
        sim=FindTokenGetSimilarityWithGapModel(T2,
name1)
        totalSim=totalSim=sim;
    return totalSim
```

   In this way, the matching process will consider cases where concept definitions contain mixed language expressions or mixed word order.
3. **STRAT3**: The third strategy is a combination of the previous strategies. The combination function simply takes the average of the scores.

B. Event Data Integration

This integration application allowed more improvement possibilities in ontology matching. The reason for this may be that there were 16 different alignments, and this number is sufficient for gaining more experience. One central idea here based on some investigations concerning the child concepts of a class. Graph-based similarity measures have been used since many years [11][12], and they are based on two basic ideas:

- Top-down: if two classes are similar, then it is more likely that among the child concepts there will also be more or less similar ones.
- Bottom-up: if among the child concepts there are many similar concepts, then it is more likely that parent concepts are similar as well.

We propose to fuse these two ideas to get a new method:

1. Beginning with leaves, the similarities of parent concepts are computed by aggregating similarities of child concepts.
2. In the next step, the similarity values of child concepts are computed by aggregating their string similarity (or another one) with the similarity of the parent concept.

For ontology matching, the following similarities were implemented:

- S1: edit distance-based similarity with affine gaps
- S2: similar to S1, but if the name of the concept contains the name of the parent, then the parent name is cropped, and just the names of children are compared.
- S3: only child names are compared (if the name contains the name of parent, then it is cropped). Names are then tokenized by delimiters or uppercase letters. The two word series of local and global concepts are then compared, and the maximal similarity is kept. During this matching process, synonym substitution is also applied.
- SP1: Computing a similarity for classes that have children by comparing children. This similarity is

computed by averaging the similarity values of children, which are above a predefined threshold.

- SP2: Similarities of parent classes are computed by aggregating their own similarity values with the similarity of child concepts.
- SP3: Similar to SP2, but the similarity of child concepts is computed by averaging the maximal similarities for each child of the first parent concept. The next 'code' snippet illustrates this similarity:

```
children1=parent1.getChildren()
children2=parent2.getChildren()
for each Ch1 in chidren1
    val=getMaxSimilarityToChildren(Ch1, children2)
    if val>lowThreshold
            totalChSim=TotalChSim+val
            N=N+1
totalChSim=totalChSim/N
aggregatedParentCildSim=nameSim(parent1,parent2)+
                              totalChSim*weightOfChildren
```

- SC1: For child concepts, their previously computed similarity values are aggregated with the similarity value computed for the parent.
- SC2: similar to SC1, but this is computed only for same typed pairs (e.g., object property pair).

Using the above-mentioned similarities, the following matching strategies were developed:

1. **STRAT-GR1**: S1 and SP1 are combined.
2. **STRAT-GR2**: S1, SP2, and SC1 are combined.
3. **STRAT-GR3**: minimal score of STRAT-GR1 and STRAT-GR2
4. **STRAT-GR4**: S3, SP3, and SC2 are combined.

V. EXPERIMENTS

Our experiments were performed on the ontology pairs described above. The reference similarity measures applied were complex strategy matchers of COMA:

- **ComaOpt**: This matcher takes into account graph-based similarities (leaves, parents), path similarities, as well as Levenshtein distance-based string similarities of names.
- **COMA**: This matcher combines name similarities, considers the types of concepts and data, graph-based metrics (leaves, parents, siblings) and path.

The baseline alignments were determined by two different people, and then a decision was made about which ones should be kept and which ones were inaccurate. Some control queries were also defined in our data integration framework to test whether the alignments were suitable. To determine alignments in the POS Printing application, a Java implementation using Alignment API was developed, since here complex alignments were used, described by a "level 2" EDOAL alignment RDF file. In the second experimental set-up, the GUI of COMA was applied to create reference alignments, and "level 0" alignment descriptions were created. In this Event Data integration problem set, two kinds of global ontology were considered. One was the complete ontology generated from Java code of the Google Calendar client API, and the other was a reduced one containing just events and their properties.

The precision of alignments found were measured in terms of the precision, recall, and F-measure, which are commonly used metrics in alignment evaluation (see for this for example Ontology Alignment Evaluation Initiative, OAEI on Internet). Precision means the fraction of retrieved alignments that are correct, recall is the fraction of correct alignments that are found, and F-measure combines precision and recall by a harmonic mean. However, for the Event Data integration task just the F-measure will be presented for reasons of space.

VI. RESULTS

The results of our evaluations are divided into two experiments, and will be discussed separately below.

A. POS Printing Data Integration

This integration task contained a generated local ontology, and a global one created by hand for this area, as described earlier. The results of the proposed matching strategies are listed in Table I.

TABLE I. RESULTS OF THE PROPOSED METHODS FOR THE POS PRINTING TASK.

Matching strategy	F-Measure	Precision	Recall
ComaOpt	0.0694	0.0735	0.0658
STRAT1	0.2209	0.2069	0.2368
STRAT2	0.3681	0.3448	*0.3947*
STRAT3	*0.4444*	*0.5085*	*0.3947*

As can be seen, the performance of the reference alignment (ComaOpt) is rather low, so it is not surprising that using some additional knowledge about the structure of generated ontologies, and exploiting the fact that the global ontology is labeled in two languages can greatly increase its performance results. It is also seen that the STRAT2 strategy has a better F-measure value than that for STRAT1. The STRAT3 matching strategy, which is a combination of STRAT1 and STRAT2, raises the precision value. This means that in this application, handling the mixed language and mixed word order effects (STRAT2) is more important than handling the effect of ontology generation rules (STRAT1). However, the most robust solution is a combination of them (the precision of STRAT3 is roughly the sum of those of STRAT1 and STRAT2).

B. Event Data Integration

The results of the Event Data Integration problem set are summarized in tables II and III. Table II contains F-measures of alignments corresponding to local ontologies and the full generated global ontology, while Table III contains those for the restricted global ontology.

Here, both the global and local ontologies are generated, so the naming rules are similar. This means in practice that comparing just the words of the generated names will boost the performance, but not so dramatically as in the previous use case.

Examining Table II first, we see that STRAT-GR1 and STRAT-GR3 can greatly improve the performance in some cases, and it is very interesting that STRAT-GR2 can give a fairly positive result when all the other measures give zero. On average, the STRAT-GR1 and STRAT-GR3 are the most promising methods, while STRAT-GR2 and STRAT-GR4 currently like the COMA baseline ones.

Table III also shows performance improvements in most cases, but these improvements are spread over the proposed methods, and there is no method that is an absolute winner. On average, however, the order of the best is different in Table II: the best one becomes STRAT-GR3, while STRAT-GR1 is the next best one. STRAT-GR2 is also better than the baseline cases.

It should be mentioned again that STRAT-GR1 is a complex matcher that combines string similarity with a one-step child-parent rescoring method, while STRAT-GR3 is similar, but it aggregates the scores of parent similarities with the similarities of children in two steps in a different way. We see that taking into consideration the similarity of descendants is important in most cases when concepts are compared. Other improvements could be achieved by including knowledge in the similarity measures concerning the ontology generation process.

VII. Summary and Future Work

Here, novel similarity methods were proposed for a special case of ontology matching; namely, when typically generated ontologies are the targets of the alignment process. To demonstrate the validity of our concept, experiments were also carried out to verify improvements in the performance for each method applied.

In the future, we plan to create a graphical user interface that supports the notation of complex alignments as well. Moreover, an automatic matcher is planned to help find these complex relations (e.g., inverse, composition, restriction). To evaluate the precision of automatic ontology matching in this case, a modified evaluation process, which takes into account the type of complex alignments will also be needed. Another interesting area might be to adaptively improve complex matching models by tuning their parameters, which means applying and customizing adaptive learning techniques to our particular case.

VIII. Acknowledgements

This work has been partially supported by the European Union and the European Social Fund through project FuturICT.hu (grant no.: TAMOP-4.2.2.C-11/1/KONV-2012-0013). Also, the work has been supported by the European Union, co-financed by the European Regional Development Fund (grant no.: GOP-1.1.1-11-2011-0052).

IX. References

[1] A. K. Alasoud, “A Multi-Matching Technique for Combining Similarity Measures in Ontology Integration,” Phd Thesis, Concordia University Montréal, Québec, Canada, 2009.

[2] P. Shvaiko and J. Euzenat, “Ontology Matching: State of the Art and Future Challenges,” IEEE Trans. Knowl. Data Eng., vol. 25, no. 1, IEEE Educational Activities Department Piscataway, NJ, USA, 2013, pp. 158–176.

[3] D. H. Ngo and Z. Bellahsene, “YAM++ - Results for OAEI 2012,” in International Semantic Web Conference, United States, 2012.

[4] D. Engmann and S. Maßmann, “Instance Matching with COMA++,” in BTW 2007 Workshop: Model Management und Metadaten-Verwaltung, Aachen, Germany, 2007, pp. 28–37.

[5] Y. Kalfoglou and B. Hu, “CMS: CROSI Mapping System - Results of the 2005 Ontology Alignment Contest,” K-Cap’05 Integrating Ontologies workshop, Banff, Canada,, 2005, pp. 77–85.

[6] Q. Ji, P. Haase, and G. Qi, “Combination of Similarity Measures in Ontology Matching using the OWA Operator,” in: Proceedings of the 12th International Conference on Information Processing and Management of Uncertainty in Knowledge-Base Systems, june 22-27, Malaga, 2008.

[7] N. Choi, I.-Y. Song, and H. Han, “A survey on ontology mapping,” Journal: SIGMOD Record, vol. 35, no. 3, ACM New York, NY, USA, 2006, pp. 34–41.

[8] G. Stoilos, G. Stamou, and S. Kollias, “A string metric for ontology alignment,” in Proceedings of the 4th international conference on The Semantic Web, Berlin, Heidelberg, 2005, pp. 624–637.

[9] M. Espinoza, A. Gómez-Pérez, and E. Mena, “LabelTranslator - a tool to automatically localize an ontology,” in Proceedings of the 5th European semantic web conference on The semantic web: research and applications, Berlin, Heidelberg, 2008, pp. 792–796.

[10] J. Euzenat and P. Shvaiko, Ontology Matching, (book) 1st ed. Springer Publishing Company, Incorporated, 2010.

[11] N. F. Noy and M. A. Musen, “Anchor-PROMPT: Using Non-Local Context for Semantic Matching,” in Proceedings of the workshop on ontologies and information sharing at the international joint conference on artificial inteligence (IJCAI), Seattle, Washington, USA , 2001, pp. 63–70.

[12] M. H. Seddiqui and M. Aono, “An efficient and scalable algorithm for segmented alignment of ontologies of arbitrary size,” Web Semantics: Science, Services and Agents on the World Wide Web, vol. 7, no. 4, 2009, pp. 344 – 356.

[13] S. A. McIlraith and D. L. Martin, “Bringing semantics to Web services,” Intelligent Systems, IEEE, vol. 18, no. 1, pp., IEEE Educational Activities Department Piscataway, NJ, USA, 90 –93, Feb. 2003.

[14] O.. Curé, M. Lamolle, and C. L. Duc, “Ontology Based Data Integration Over Document and Column Family Oriented NoSQL stores,” in The 7th International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS2011), Bonn, Germany, 2011.

[15] C. Kavitha and G. S. Sadasivam, “Ontology Based Semantic Integration of Heterogeneous Databases,” in European Journal of Scientific Research, 2011, vol. Vol.64 No.1, pp. 115–122.

[16] http://oaei.ontologymatching.org/ (2013)

[17] C. Bizer, “D2RQ - treating non-RDF databases as virtual RDF graphs,” in In Proceedings of the 3rd International Semantic Web Conference (ISWC2004, poster presentation), Hiroshima, Japan, 2004.

[18] http://qdox.codehaus.org/ (2014)

[19] A. Banhalmi, D. Paczolay, A. Z. Vegh, G. Antal, and V. Bilicki, “Development of a Novel Semantic-Based System Integration Framework,” in Engineering of Computer Based Systems (ECBS-EERC), 2013 3rd Eastern European Regional Conference on the Engineering of Computer Based Systems, Budapest, Hungary, 2013, pp. 18–24.

[20] D. Aumueller, H.-H. Do, S. Massmann, and E. Rahm, “Schema and ontology matching with COMA++,” in Proceedings of the 2005 ACM SIGMOD international conference on Management of data, New York, NY, USA, 2005, pp. 906–908.

[21] J. Euzenat at al., “Results of the ontology alignment evaluation initiative 2011,” in Proc. 6th ISWC workshop on ontology matching (OM), Bonn (DE), pages 85–110, 2011.

TABLE II. RESULTS OF REFERENCE AND PROPOSED MATCHING METHODS IN TERMS OF THE F-MEASURE, WHEN A LARGE GLOBAL ONTOLOGY IS USED IN THE EVENT INTEGRATION USE CASE.

Application/Matching strategy	**Coma-Opt**	**COMA**	**STRAT-GR 1**	**STRAT-GR 2**	**STRAT-GR 3**	**STRAT-GR 4**
Basic-php-events-lister2.04	0.25	0.25	*0.29*	0.20	0.26	*0.29*
calendar	*0.21*	0.25	0.00	0.13	0.00	0.00
calendar_v2.0_en	0.14	0.17	*0.27*	0.20	0.21	0.23
calendar_ws	0.00	0.00	0.00	*0.25*	0.00	0.00
calendarix_0_8_20080808	0.13	0.05	0.13	0.17	0.20	0.11
calendartechnique-2.0.2RC4	0.19	0.23	*0.30*	0.12	*0.32*	0.07
cmappCalendar_1.1	0.19	0.19	*0.82*	0.13	0.75	0.59
Fullcalendar	0.00	0.00	0.00	*0.29*	0.00	0.00
luxcal273	*0.24*	0.17	0.20	0.10	0.17	0.11
maian_events	0.21	0.24	*0.31*	0.19	0.25	0.20
mapcal-0.2.1	*0.36*	*0.36*	0.35	0.27	0.30	*0.36*
openbookings.org_v0.6.4b	0.00	0.00	0.00	*0.35*	0.00	0.00
PHPCalendar.Basic.2.3	0.19	0.07	0.00	0.00	0.00	0.00
supercali-1.0.7	0.15	0.20	0.15	0.17	*0.21*	0.10
vcalendar_php_1.1.5	0.22	0.25	*0.53*	0.20	0.43	0.17
webical-0.4.1	0.43	0.46	0.45	0.39	*0.60*	*0.62*
Average	0.18	0.18	*0.24*	0.20	*0.23*	0.18

TABLE III. RESULTS OF REFERENCE AND PROPOSED MATCHING METHODS IN TERMS OF THE F-MEASURE, WHEN A RESTRICTED GLOBAL ONTOLOGY IS USED IN THE EVENT INTEGRATION USE CASE.

Application/Matching strategy	**Coma-Opt**	**COMA**	**STRAT-GR 1**	**STRAT-GR 2**	**STRAT-GR 3**	**STRAT-GR 4**
Basic-php-events-lister2.04	0.40	0.36	0.44	0.46	0.44	*0.54*
calendar	*0.33*	0.00	0.17	0.27	*0.33*	0.00
calendar_v2.0_en	0.40	0.40	*0.53*	0.44	0.44	*0.53*
calendar_ws	0.20	0.00	0.20	*0.67*	0.20	0.20
calendarix_0_8_20080808	0.30	0.27	0.39	0.33	0.39	0.22
calendartechnique-2.0.2RC4	0.27	0.28	*0.40*	0.30	*0.40*	0.15
cmappCalendar_1.1	0.56	0.56	*0.88*	0.47	*0.88*	0.80
Fullcalendar	0.60	0.00	0.55	*0.71*	0.55	0.40
luxcal273	*0.52*	0.40	0.46	0.35	0.45	0.44
maian_events	0.53	0.40	0.63	0.57	0.63	*0.67*
mapcal-0.2.1	0.61	*0.64*	0.50	0.42	0.48	0.57
openbookings.org_v0.6.4b	0.00	0.00	0.00	*0.24*	0.00	0.00
PHPCalendar.Basic.2.3	0.33	0.32	*0.53*	0.33	0.44	0.25
supercali-1.0.7	0.67	0.67	0.53	0.59	*0.71*	0.53
vcalendar_php_1.1.5	0.64	0.55	0.72	0.48	*0.73*	0.50
webical-0.4.1	0.56	0.58	0.65	0.67	*0.73*	0.69
Average	0.43	0.34	0.47	0.46	*0.49*	0.40