Scalable Matching of Industry Models – A Case Study

Brian Byrne[1], Achille Fokoue[2], Aditya Kalyanpur[2], Kavitha Srinivas[2], and Min Wang[2]

[1] IBM Software Group, Information Management, Austin, Texas
`byrneb@us.ibm.com`
[2] IBM T. J. Watson Research Center, Hawthorne, New York
`achille, adityakal, ksrinivs, min@us.ibm.com`

Abstract. A recent approach to the problem of ontology matching has been to convert the problem of ontology matching to information retrieval. We explore the utility of this approach in matching model elements of real UML, ER, EMF and XML-Schema models, where the semantics of the models are less precisely defined. We validate this approach with domain experts for industry models drawn from very different domains (healthcare, insurance, and banking). We also observe that in the field, manually constructed mappings for such large industry models are prone to serious errors. We describe a novel tool we developed to detect suspicious mappings to quickly isolate these errors.
keywords: Model matching.

1 Introduction

The world of business is centered around information. Every business deals with a myriad of different semantic expressions of key business information, and expends huge resources working around the inconsistencies, challenges and errors introduced by a variety of information models. Typically, these information models organize the data, services, business processes, or vocabulary of an enterprise, and they may exist in different forms such as ER models, UML models, thesauri, ontologies or XML schema. A common problem is that these varying models rarely share a common terminology, because they have emerged as a result of several inputs. In some cases, mergers of organizations operating in the same business result in different information models, to express the same exact concepts. In other cases, they may have been developed by different organizational units to express overlapping business concepts, but in slightly different domains.

Irrespective of how these models came about, today's business is faced with many different information models, and an increasing need to integrate across these models, through data integration, shared processes and rules, or reusable business services. In all of these cases, the ability to relate, or map, between different models is critical. Both human attempts to manually map different information models and and the use of tools to automate mappings however are very error prone in the real world. For humans, the source of the error comes from multiple sources:

– The size of these models (typically, these models have several thousand elements each)
– The fact that lexical names of model elements rarely match, or when they do match, its because of the wrong reasons (e.g., a document may have an endDate attribute, as does a claim, but the two endDate reflect semantically different things, although they match at the lexical level).
– Models often express concepts at different levels of granularity, and it may not always be apparent at what level the concept should be mapped. In many real world mappings, we have observed a tendency for human analysts to map everything to generic concepts rather than more specific concepts. While these mappings are not necessarily invalid, they have limited utility in data integration scenarios, or in solution building.

The above points make it clear that there is a need for a tool to perform semi-automated model mapping, where a tool can help suggest appropriate mappings to a human analyst. Literature on ontology matching and alignment is clearly helpful in designing such a tool. Our approach to building such a tool is similar in spirit to the ideas implemented in Falcon-AO [1],[2] and PRIOR ([3]), except that we adapted their techniques to UML, ER and EMF models. Matching or alignment across these models is different from matching ontologies, because the semantics of these models are poorly defined compared to those of ontologies. Perhaps due to this reason, schema mapping approaches tend to focus mostly on lexical and structural analysis. However, existing schema mapping approaches scale very poorly to large models. Most analysts in the field therefore tend to revert to manual mapping, despite the availability of many schema mapping tools.

We however make the observation that in most industry models, the semantics of model elements is buried in documentation (either within the model, or in separate PDF, Excel or Word files). We therefore use techniques described by Falcon-AO and PRIOR to build a generic representation that allows us to exploit the structural and lexical information about model elements along with semantics in documentation. The basic idea, as described in PRIOR is to convert the model mapping problem into a problem of information retrieval. Specifically, each model element is converted into a virtual document with a number of fields that encode the structural, lexical and semantic information associated with that model element. This information is in turn expressed as a term vector for a document. Mapping across model elements is then measured as a function of document similarity; i.e., the cosine similarity between two term vectors. This approach scales very well because we use the Apache Lucene text search engine for indexing and searching these virtual documents.

The novelty in our approach is that we also developed an engine to identify suspicious mappings produced either by our tool or by human analysts. We call this tool a Lint engine for model mappings, after the popular Lint tool which checks C programs for common software errors. The key observation that motivated our development of the Lint engine was that human model mappings were shockingly poor for 3/4 model mappings that were produced in real business scenarios. Common errors made by human analysts included the following:

– Mapping elements to overly general classes (equivalent to *Thing*).
– Mapping elements to subtypes even when the superclass was the appropriate match. As an example, *Hierarchy* was mapped to *HierarchyType* when *Hierarchy* existed in the other model.
– Mapping elements that were simply invalid or wrong.

We encoded 6 different heuristics to flag suspicious mappings, including heuristics that can identify common errors made by our own algorithm (e.g., the tendency to match across elements with duplicate, copied documentation). The Lint engine for model mappings is thus incorporated as a key filter for semi-automated model mapping tool, to reduce the number of false positives that the human analyst needs to examine. A second use of our tool is of course to review the quality of human mappings in cases where the model mappings were produced manually.

Our key contributions are as follows:

– We describe a technique to extend existing techniques in ontology mapping to the problem of model mapping across UML, ER, and EMF models. Unlike existing approaches in schema mapping, we exploit semantic information embedded in documentation along with semantic and lexical information to perform the mapping.
– We describe a novel Lint engine which can be used to review the quality of model mappings produced either by a human or by our algorithm.
– We perform a detailed evaluation of the semi-automated tool on 7 real world model mappings. Four of the seven mappings had human mappings that were performed in a business context. We evaluated the Lint engine on these 4 mappings. The mappings involved large industry specific framework models with thousands of elements in each model in the domains of healthcare, insurance, and banking, as well as customer models in the domains of healthcare and banking. Our approach has therefore been validated on mappings that were performed for real business scenarios. In all cases, we validated the output of both tools with domain experts.

2 Related Work

Ontology matching or the related problem of schema matching is a well studied problem, with a number of different approaches that are too numerous to be outlined here in detail. We refer the reader instead to surveys of ontology or schema matching [4–6]. A sampling of ontology matching approaches include GLUE [7], PROMPT [8], HCONE-merge [9] and SAMBO [10]. Sample approaches to schema matching include Cupid [11], Artemis [12], and Clio [13–16]. Our work is mostly closely related to Falcon-AO [1, 2] and PRIOR [3], two recent approaches to ontology matching that combine some of the advantages of earlier approaches such as linguistic and structural matching incorporated within an information-retrieval approach, and seem well positioned to be extended to address matching in shallow-structured models such as UML, ER and EMF models. Both Falcon-AO and PRIOR have been compared with existing systems in OAEI 2007 and appear to scale well in terms of performance. Because our work addresses matching across very large UML, ER and EMF data models (about 5000 elements), we adapted the approaches described in Falcon-AO and PRIOR to these models. Matching or alignment across these models is different from matching ontologies, because the semantics of these models are poorly defined compared to those of ontologies. More importantly, we report the results of applying these techniques to 7 real ontology matching problems in the field, and describe scenarios where the approach is most effective.

3 Overall Approach

3.1 Matching algorithm

Casting the matching problem to an IR problem Similar to approaches outlined in Falcon-AO [1],[2] and PRIOR ([3]), a fundamental principle in our approach is to cast the problem of model matching into a classical Information Retrieval problem. Model elements (e.g. attributes or classes) from various modeling representations (e.g. XML Schema, UML, EMF, ER) are transformed into virtual documents. A virtual document consists of one or more fields capturing the structural, lexical and semantic information associated with the corresponding model element.

A Vector Space Model (VSM) [17] is then adopted: each field F of a document is represented as a vector in a N_F-dimensional space, with N_F denoting the number of distinct words in field F of all documents. Traditional TF-IDF (Term Frequency - Inverse Document Frequency) values are used as the value of coordinates associated to terms. Formally, let D_F denotes the vector associated with the field F of a virtual document D, and $D_F[i]$ denotes the ith coordinate of the vector associated with the field F of a virtual document D:

$$D_F[i] = tf_i * idf_i \tag{1}$$

$$tf_i = |t_i|/N_F \tag{2}$$

$$idf_i = 1 + log(ND/d_i) \tag{3}$$

where

- $|t_i|$ represents the number of occurrence, in the field F of document D, of the term t corresponding to the ith coordinate of the vector D_F,
- ND corresponds to the total number of documents, and
- d_i is the number of documents in which t appears at least once in F

The similarity $sim(A, B)$ between two model elements A and B is computed as the weighted mean of the cosine of the angle formed by their field vectors.

Formally, let D and D' be the virtual documents corresponding to A and B, respectively. Let q be the number of distinct field names in all documents.

$$sim(A,B) = \frac{\sum_{k=1}^{q} \alpha_k * cosine(D_{F_k}, D'_{F_k})}{\sum_{k=1}^{q} \alpha_k} \tag{4}$$

$$cosine(D_{F_k}, D'_{F_k}) = \frac{\sum_{i=1}^{N_{F_k}} D_{F_k}[i] * D'_{F_k}[i]}{|D_{F_k}| * |D'_{F_k}|} \tag{5}$$

$$|D_F| = \sqrt{\sum_{i=1}^{N_F} (D_F[i])^2} \tag{6}$$

where α_k is the weight associated with the field F_k, which indicates the relative importance of information encoded by that field.

In our Lucene[3]-based implementation, before building document vectors, standard transformations, such as stemming/lemmatization, stop words removal, lowercasing, etc, are performed. In addition to these standard transformations, we also convert camel case words (e.g. “firstName”) into corresponding group of space separated words (e.g. “first name”).

Transforming model elements into virtual documents A key step in our approach is the transformation of elements of a data model into virtual documents. For simplicity of the presentation, we assume that the data model is encoded as a UML Class diagram[4]

The input of the transformation is a model element (e.g. attribute, reference/association, or class). The output is a virtual document with the the following fields:

– *name.* This field consists of the name of the input element.
– *documentation.* This field contains the documentation of the input model element.
– *containerClass.* For attribute, reference and association, this field contains the name and documentation of their containing class.
– *path.* This field contains the path from the model root package to the model element (e.g. for an attribute ”bar” of the class ”foo” located in the package ”example”, the path is /example/foo/bar).
– *body.* This field is made of the union of terms in all fields except path.

While the first two fields encode only lexical information, the next two fields (containerClass and path) capture some of the structure of the modeling elements. In our implementation, when the models to be compared appear very similar, which translates to a very large number of discovered mappings, we typically empirically adjust upwards the weight of the “containerClass” and “path” fields to convey more importance to the structural similarity.

[3] http://lucene.apache.org/java/docs/
[4] Our implementation is able to handle more data model representations, including XML Schemas, ER diagrams, and EMF ECore models.

For the simple UML model shown in Figure 3.1, 5 virtual documents will be created, among which is the following:

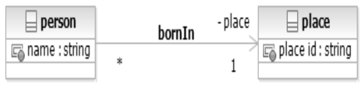

Fig. 1. Simple Model Example

1. Virtual document corresponding to the class "Place":
   - *name* : "Place"
   - *documentation*: "a bounded area defined by nature by an external authority such as a government or for an internal business purpose used to identify a location in space that is not a structured address for example country city continent postal area or risk area a place may also be used to define a logical place in a computer or telephone network e.g. laboratory e.g. hospital e.g. home e.g. doctor's office e.g. clinic"
   - *containerClass*: ""
   - *path*: "/simple test model/place"
   - *body*:"place, a bounded area defined by nature by an external authority such as a government or for an internal business purpose used to identify a location in space that is not a structured address for example country city continent postal area or risk area a place may also be used to define a logical place in a computer or telephone network e.g. laboratory e.g. hospital e.g. home e.g. doctor's office e.g. clinic"
2. Virtual document corresponding to the attribute "Place id":
   - *name* : "place id"
   - *documentation*: "the unique identifier of a place"
   - *containerClass*: "place, a bounded area defined by nature by an external authority such as a government or for an internal business purpose used to identify a location in space that is not a structured address for example country city continent postal area or risk area a place may also be used to define a logical place in a computer or telephone network e.g. laboratory e.g. hospital e.g. home e.g. doctor's office e.g. clinic"
   - *path*: "/simple test model/place/place id"
   - *body*: "place id, the unique identifier of a place, place, a bounded area defined by nature by an external authority such as a government or for an internal business purpose used to identify a location in space that is not a structured address for example country city continent postal area or risk area a place may also be used to define a logical place in a computer or telephone network e.g. laboratory e.g. hospital e.g. home e.g. doctor's office e.g. clinic"

Adding lexical and semantic similarity between terms The cosine scoring scheme presented above (4) is intolerant to even minor lexical or semantic variations in terms. For example, the cosine score computed using equation (4) for the document vectors (gender: 1, sex: 0) and (gender:0, sex: 1) will be 0 although "gender" mentioned in the first document is clearly semantically related to "sex" appearing in the second document. To address this limitation, we modify the initial vector to add, for a given term t, the indirect contributions of terms related to t as measured by a term similarity metric. Formally, instead of using D_{F_k} (resp. D'_{F_k}) in equation (4), we used the document vector $\widehat{D_{F_k}}$ whose coordinates $\widehat{D_{F_k}}[i]$, for $1 \leq i \leq N_{F_k}$, are defined as follows:

$$\widehat{D_{F_k}}[i] = D_{F_k}[i] + \beta_i * \sum_{j=1 \ \& \ j \neq i}^{N_{F_k}} termSim(t_i, t_j) * D_{F_k}[j] \tag{7}$$

$$\beta_i = \begin{cases} 0 & \text{if, for all } j \neq i, D_{F_k}[j] = 0, \\ \frac{1}{\sum_{j=1 \ \& \ j \neq i \ \& \ D_{F_k}[j] \neq 0}^{N_{F_k}} 1} & \text{otherwise} \end{cases} \tag{8}$$

where

- *termSim* is a term similarity measure such as Jaccard or Levenshtein similarity measure (for lexical similarity), a semantic similarity measure based on WordNet [18] [19], or a combination of similarity measures. $termSim(t_i, t_j) * D_{F_k}[j]$ in (7) measures the contribution to the term t_i of the potentially related term t_j.
- β_i is the weight assigned to indirect contributions of related terms.

For efficiency, when comparing two document vectors, we only add in the modified document vectors, the contributions of terms corresponding to at least one non-zero coordinate of any of the two vectors.

The equation (7) applied to the previous example transforms (gender:1, sex :0) to (gender: 1, sex: termSim("sex", "gender")) and (gender: 0, sex: 1) to (gender: termSim("gender", "sex"), sex: 1). Assuming that termSim("sex", "gender"), which is the same as termSim("gender", "sex"), is not equal to zero, the cosine score of the transformed vectors will obviously be different from zero, and will reflect the similarity between the terms "gender" and "sex".

For the results reported in the evaluation section, only the Levenshtein similarity measure was used. Using a semantic similarity measures based on wordnet significantly increasing the algorithm running time with a marginal improvement of quality of the resulting mappings. The running time performance of semantic similarity measures based on WordNet, was still unacceptable after restricting related terms to synonyms and hyponyms.

Our approach provides a tigher integration of cosine scoring scheme and a term similarity measure. In previous work, e.g. Falcon-AO[2], the application of the term similarity measure (Levenshtein measure in Falcon-AO) is limited to names of model elements, and the final score is simply a linear combination of the cosine score and the measure of similarity between model element names.

4 Evaluation of Model Matching Algorithm

To evaluate the model matching algorithm, we accumulated industry models and customer data models from IBM architects who regularly build solutions for customers. The specific model comparisons we chose were ones that IBM architects need mapped in the field. In four cases out of 7 model matching comparisons, the matching had been performed by IBM solutions teams manually. We tried to use these as a 'gold standard' to evaluate the model matching algorithm, but unfortunately found that in 3 of 4 cases, the quality of the manual model matching was exceedingly poor. We address this issue with a tool to assess matching quality in the next section.

As shown in Table 1, the industry models we used in the comparisons included BDW (a logical data model for financial services), HPDM (a logical data model for healthcare), MDM (a model for the IBM's solution for master data management), RDWM (a model for warehouse solutions for retail organizations), and IAA (a model for insurance). Model A in the table is a customer ER model in the healthcare solutions space, model B is a customer logical data model in financial services, and model C is customer logical data model in retail. To evaluate our model matching results, we had two IBM architects assess the precision of the best possible match produced by our algorithm. Manual evaluation of the matches was performed on sample sizes of 100 in 5 of 7 cases (all cases except the IAA-BDW and A-HPDM comparisons). For IAA-BDW, we used a sample size of 50 because the algorithm produced less than 100 matches. For A-HPDM, we relied on previously created manual mappings to evaluate both precision and recall (recall was at 25%). The sizes of these models varied from 300 elements to 5000 elements.

We make two observations about our results:

- (a) The results show a great deal of variability ranging from cases where we had 100% precision in the top 100 matches, to 52% precision. This reflected the degree to which the models shared a common lineage or common vocabulary in their development. For example, RDWM was actually derived from BDW, and this is clearly reflected in the model matching results. IAA and BDW target different industries (and therefore do not have much in common), and this is a scenario where the algorithm tends to make more errors. We should point out that although IAA and BDW target different industries (insurance and banking respectively), there is a real business need for mapping common or overlapping concepts across these disparate models, so the matching exercise is not a purely academic one.
- (b) Even in cases where the precision (or recall) was low, the IBM architects attested to the utility of such a semi-automated approach to model matching, because their current process is entirely manual, tedious and error prone. None of the model mapping tools available to them currently provide results that are usable or verifiable.

Models Compared	Number of matches	Precision
A-HPDM	43	67%
B-BDW	197	74%
MDM-BDW	149	71%
MDM-HPDM	324	54%
RDWM-BDW	3632	100%
C-BDW	3263	96%
IAA-BDW	69	52%

Table 1. Model matching results

4.1 Lint Engine

We turn now to another aspect of our work, which is to somehow measure the quality of ontology matching in the field. As mentioned earlier, we initially started our work with the hope of using manual matchings as a gold standard to measure the output of our matching algorithm, but were surprised to find a rather large number of errors in the manually generated model mappings. A lot of these errors were presumably due to the ad hoc nature of the manual mapping process, leading to poor transcription of names, e.g., changes in spaces, appending package names etc. when writing mapping results in a separate spreadsheet; specification of new classes/attributes/relationships to make up a mapping, when the elements did not exist in the original models etc. Also, there were cases in which mappings were made to an absurdly generic class (such as *Thing*) which rendered them meaningless.

In order to deal with the above issues, and also improve the accuracy of our mapping tool, we decided to write a Lint Engine to detect suspicious mappings. The engine runs through a set of suspicious mapping patterns, with each pattern being assigned a severity rating and a user-friendly explanation, both specified by the domain expert. We have currently implemented the following six mapping patterns based on discussions with a domain-expert:

– *Element not found*: The pattern detects mappings where one or more elements involved does not exist in any of the models. This pattern is assigned a high severity since it indicates something clearly suspicious or wrong.
– *Exact name mismatches*: Detects mappings where a model element with an exact lexical match was not returned. This does not necessarily indicate an incorrect mapping, however does alert the user of a potentially interesting alternative that may have been missed.
– *Duplicate documentation*: Detects mappings where the exact same documentation is provided for both elements involved in the mapping. This may arise when models or portions of models are copy/pasted across.
– *Many-to-1 or 1-to-Many*: Detects cases where a single element in one model is mapped to a suspiciously large number elements in another model. As mentioned earlier, these typically denote mappings to an absurdly generic class/relation.

– *Class-Attribute proliferations*: Detects cases when a single class' attributes/relations are mapped to attributes/relations of several different classes in the other model. What makes this case suspicious is that model mappings are a means to an end, typically used to specify instance transformations. Transformations can become extremely complex when class-attribute proliferations exist.
– *Mapping without documentation*: Detects cases where all the elements involved in the mapping have no associated documentation. This could arise due to lexical and structural information playing a role in the mapping, however the lack of documentation points to a potentially weaker match.

We applied our Lint engine to the manual mappings to see if it could reveal in more detail the defects we had observed. The results are summarized in the Tables 2 - 5 below.

Total number of mappings	306
Total number of suspicious mappings	151 (51 %)
One To Many Mappings	143 (46 %)
Mapping Without Documentation	40 (25 %)
Exact Name Not Match	13 (8 %)
Duplicate Documentation	2 (1 %)

Table 2. Evaluation of B-BDW manual mappings using our Lint Engine

Total number of mappings	702
Total number of suspicious mappings	702 (100 %)
Name Not Found in Models	702 (100 %)
Mapping Without Documentation	702 (100 %)
Exact Name Not Match	30 (4 %)
One To Many Mappings	312 (44 %)

Table 3. Evaluation of BDW-MDM manual mappings using our Lint Engine

Total number of mappings	117
Total number of suspicious mappings	95 (81 %)
Mapping Without Documentation	95 (100 %)
One To Many Mappings	10 (10 %)
Duplicate Documentation Checker	9 (9 %)
Name Not Found in Models	2 (2 %)

Table 4. Evaluation of A-HPDM manual mappings using our Lint Engine

Total number of mappings	748
Total number of suspicious mappings	748 (100 %)
Mapping Without Documentation	741 (99 %)
Name Not Found in Models	459 (61 %)
Class Attribute Mapping Proliferation	472 (63 %)
Duplicate Documentation Checker	378 (50 %)
One To Many Mappings	321 (42 %)
Exact Name Not Match	33 (4 %)

Table 5. Evaluation of MDM-HPDM manual mappings using our Lint Engine

The results are quite shocking, e.g., in the BDW-MDM case, all 702 mappings specified an element that did not exist in either of the two models. The only explanation for this bizarre result is that mapping exercises, typically performed in Excel etc, are hideously inaccurate - in particular, significant approximation of the source and target elements is pervasive. Another point to note is that humans like to try and cheat and map at a generic level, and this practice seems to be quite pervasive, as such mappings were discovered in almost all the cases. Finally, the lack of, or duplication of documentation can be identified in many ways (e.g. products such as SoDA from Rational[5]) - but surfacing this during the mapping validation is very helpful. It helps present an estimation of the degree of confidence in the foundation of the mapping - the understanding of the elements being mapped.

The results were analyzed in detail by a domain expert who verified that the accuracy and usefulness for the suspicious mappings was very high (in the B-BDW case, only 1 suspicious mapping produced by Lint was actually correct). The fact that the lint engine found roughly less than 1 valid mapping for every 10 suspicious ones is an indication of the inefficiency of manual mapping practices. What the engine managed to do effectively is to filter from a huge pool of mappings, the small subset that need human attention, while hinting to the user what may be wrong by nicely grouping the suspicious mappings under different categories.

References

1. Jian, N., Hu, W., Cheng, G., Qu, Y.: Falcon-ao: Aligning ontologies with falcon. In: Proceedings of K-CAP Workshop on Integrating Ontologies. (2005)
2. Qu, Y., Hu, W., Cheng, G.: Constructing virtual documents for ontology matching. In: Proceedings of the 15th international conference on World Wide Web, Edinburgh, UK (2006)
3. Mao, M., Peng, Y., Spring, M.: A profile propagation and information retrieval based ontology mapping approach. In: Proceedings of the 3rd International Conference on Semantics, Knowledge and Grid (research track), Xian, China (2007)
4. Noy, N.F.: Semantic integration: a survey of ontology-based approaches. SIGMOD Rec. **33**(4) (2004) 65–70

[5] http://www-01.ibm.com/software/awdtools/soda/index.html

5. Kolaitis, P.G.: Schema mappings, data exchange, and metadata management. In: Proceedings of the 24th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Baltimore, Maryland (2005)
6. Bernstein, P.A., Melnik, S.: Model management 2.0: manipulating richer mappings. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Beijing, China (2007)
7. Doan, A., Madhavan, J., Dhamankar, R., Domingos, P., Halevy, A.: Learning to match ontologies on the semantic web. The VLDB Journal **12**(4) (2003) 303–319
8. Noy, N.F., Musen, M.A.: Prompt: Algorithm and tool for automated ontology merging and alignment. In: Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence, Austin, Texas, USA (2000)
9. Kotis, K., Vouros, G., Stergiou, K.: Capturing semanticstowards automatic coordination of domain ontologies. In: AIMSA. (2004) 22–32
10. Lambrix, P., Tan, H.: Sambo-a system for aligning and merging biomedical ontologies. Web Semant. **4**(3) (2006) 196–206
11. Madhavan, J., Bernstein, P.A., Rahm, E.: Generic schema matching with cupid. In: VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (2001) 49–58
12. Castano, S., De Antonellis, V., De Capitani di Vimercati, S.: Global viewing of heterogeneous data sources. IEEE Trans. on Knowl. and Data Eng. **13**(2) (2001) 277–297
13. Miller, R.J., Haas, L.M., Hernández, M.A.: Schema mapping as query discovery. In: Proceedings of 26th International Conference on Very Large Data Bases, Cairo, Egypt (2000)
14. Miller, R.J., Hernández, M.A., Haas, L.M., Yan, L.L., Ho, C.T.H., Fagin, R., Popa, L.: The clio project: Managing heterogeneity. SIGMOD Record **30**(1) (2001)
15. Bernstein, P.A., Ho, H.: Model management and schema mappings: Theory and practice. In: Proceedings of the 33rd International Conference on Very Large Data Bases, University of Vienna, Austria (2007)
16. Hernández, M.A., Popa, L., Ho, H., Naumann, F.: Clio: A schema mapping tool for information integration. In: Proceedings of the 8th International Symposium on Parallel Architectures, Algorithms and Networks, Las Vegas, Nevada, USA (2005)
17. Raghavan, V.V., Wong, S.K.M.: A critical analysis of vector space model for information retrieval. Journal of the American Society for Information Science **37**(5) (January 1999) 279–287
18. Jiang, J.J., Conrath, D.W.: Semantic similarity based on corpus statistics and lexical taxonomy. CoRR **cmp-lg/9709008** (1997)
19. Lin, D.: An information-theoretic definition of similarity. In: ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (1998) 296–304