Mutual Enhancement of Schema Mapping and Data Mapping

Mingchuan Guo, Yong Yu

Department of Computer Science and Engineering
Shanghai JiaoTong University. Shanghai, China. 200030
{gmc,yyu}@apex.sjtu.edu.cn

Abstract. Schema mapping and data mapping have been two topics widely studied in traditional database related communities. Recently, with the rising interest in the semantical web, the tasks of integrating heterogeneous information sources, both in a schema and a data instance level, are becoming of more practical importance.
Current efforts at resolving these two mapping tasks have been carried out separately. In this paper, we propose a new method that simultaneously attacks these two tasks and achieves a kind of mutual enhancement between them. By applying our method to a *movie-hunting* scenario, we show that precision and recall are both quite high. Our method works well when dealing with numerical-valued attributes. We also show that this method is especially appealing from a semantic web perspective, given its assets of being relatively lightweight and easy to extend.

1 INTRODUCTION

In 2001, T.Berners-Lee proposed the idea of the semantic web [1], in which data have structures, and ontologies(schemas) describe the semantics of the structured data, thus facilitating computers to better understand and process those data.

The idea of building a semantically-rich web raises enormous interest. Yet to put the semantic web into a reality, several major problems have to be solved, with interoperability between heterogeneous information sources being one of them. Given the de-centralized nature of the web, it is certain that there will be large numbers of different information providers, each using their own ontologies(schemas), and there might be overlaps between the instance-level information they provide. It is of practical importance to enable the interoperability between different sources so as to leverage the benefits of the semantic web.

The twin sub-tasks associated with interoperability are:

1. *Schema-level* mapping: The task of discovering the correspondences between different schemas.
2. *Instance-level* mapping: The task of identifying whether two or more instances from different sources actually refer to a single entity in the real world.

For the sake of brevity, hereafter, we simply say *schema mapping* and *data mapping* when referring to the above two sub-tasks respectively.

While the ideas of schema mapping and data mapping are certainly not novelties(as will be briefly discussed in Section 2), it is the semantic web scenario, which puts special emphasis on the importance of data interoperability, that justifies the amount of attention and efforts on these two tasks nowadays. Consider the following real-life scenario:

> *Mike is interested in a movie, say, Finding Nemo. There are many movie web sites and on-line DVD dealers. Mike intends develop a more comprehensive idea of what a movie Finding Nemo is, and at the same time comparing services and prices offered by different cinemas and DVD vendors. With semantic web providing information in a machine-processable way, Mike is alleviated of the somewhat formidable task of manually searching for online movie and DVD dealers/cinemas information, and hopefully, the integration of information and comparison between DVD vendors/cinemas are also automated.*

While such a *movie-hunting* scenario may be hailed as a paradigm application of the semantic web, its feasibility hinges directly upon the two kinds of mapping tasks we just mentioned:

- *Schema mapping.* In order to make comparison meaningful, we must know the correspondences between the schema elements used by different information sources. For example, it makes no sense to compare DVD prices from source A with movie production years from source B.
- *Data mapping.* Mike should always be sure that the information he gets from any specific sources do map to his interested movie, *Finding Nemo*. In the case of a data-level mismatch, the results might be meaningless and even misleading, such as telling Mike that source A's offered price for *Finding Nemo* is lower than source B's offered price for *Finding Fish*, which is a totally different movie.

While both schema mapping and data mapping have been under research for quite some time, these efforts are being carried out somewhat independently.

In this paper,we propose a new method that performs schema mapping by utilizing data mapping information, and at the same time promoting data mapping with the aid of the (partial-)result of schema mapping. We believe, by simultaneously attacking these two tasks, this method will achieve a kind of mutual enhancement between schema mapping and data mapping. Specifically, this paper makes the following contributions:

- Proposes the idea that schema mapping and data mapping might be carried out simultaneously in a mutually-enhancing way.
  To our best knowledge, ours is the first such attempt.
- Lists some desirable characteristics that make our method especially appropriate in the semantic web context.

– Shows how some otherwise intractable mappings can be performed using our method.

The rest of the paper is arranged as follows. In Section 2, we give a brief review of related work. Section 3 explores our intuitions and rationales. It is in Section 4 that we present the mechanism of our method. Preliminary experimental results based on real-world data are given in Section 5. Section 6 provides the list of future work. We conclude this paper in Section 7.

2 RELATED WORK

2.1 Schema Mapping

AnchorPrompt[2], Cupid[3] and SimilarityFlooding[4] are well-known schema mapping methods that rely on schema information alone, such as attribute names and structural information, when performing the mapping task.

Many methods also utilize instance information. LSD[5] uses machine-learning to train a set of base learners and a meta learner. When performing the mapping task, base learners' mapping predictions are coordinated by the meta-learner to get at the final result. In [6], by borrowing ideas such as mutual information and conditional entropy from information theory, mapping is made possible when faced with opaque schema attribute names or opaque data values.

While the above methods utilize instances as a whole(serving as training data, etc), several other methods rely on single specific data instances. In [7], ontology mapping is inspired by language games[8]. Data instances known by both ontologies serve as joint attentions that form the basis of the discovery of shared concepts. The ILA system[9], which also resorts to overlapping items to derive schema mapping, discusses issues such as instance selection criteria and a mapping hypothesis evaluation mechanism.

Emphasis is also put on leveraging different kinds of information. Apart from LSD, COMA[10] also uses many matchers, each exploring different properties of schema attributes. Domain specific knowledge [5, 10–12] and historical mapping information[10–12] might also contribute to new mapping tasks.

Among all current methods, iMap[12] is distinguished in that it could also find many kinds of complex mappings. By means of deploying specific searchers, it can search and verify candidate complex mappings. iMap also has features such as the ability to explain predicted mappings. Overlapping instances are also used in iMap to discover equation-like mappings.

2.2 Data Mapping

Data mapping is studied in the database community as data cleaning and deduplication problems. Virtually all incumbent efforts aim to find identical data instances that are in a same table. Common practices([13, 14])are to apply textual similarity functions, and compare the result with a threshold to determine whether two tuples actually refer to a single real world entity.

In the semantic web context it is more probable that we have to map data instances cross different sources. Record linkage[15] is the methodology of bringing together corresponding records from two or more files. In Doan's recent work[16], the PROM solution, a profiler-based approach that performs data mapping cross tables utilizing disjoint attributes, is proposed. In [17], R.Guha innovates by introducing the concepts of Discriminant Descriptions and a bootstrapping process.

Common purpose search engines such as google[1] provide another kind of data mapping, with users specifying the data to be mapped by providing several keywords. It is partially due to the fact that current web infrastructure doesn't support well-structured information presentation that incumbent search engineer's mapping results are usually not satisfying. However, new semantic web-inspired techniques, such as XQuery[2] and XSEarch[18], have already provided us with an inkling of how data mapping might be carried out in a more structured and semantic way.

While some don't insist on data instances to be mapped coming from a single source, all current data mapping methods pre-assume that there must be at least a partial mapping between the schemas of the involved data sources.

To our best knowledge, there has been no attempt to simultaneously consider these two mapping tasks of schema mapping and data mapping.

3 INTUITIONS AND RATIONALES

3.1 Two Practical Requirements

Our research is carried out in a semantic web-oriented way.That is, we would like our method to be especially applicable in (yet not confined to) the semantic web scenario. We discern that two practical requirements, namely, the need to be lightweight and the need to be extensible and self-improving, should be given adequate consideration for this end.

The Need to be Lightweight Given the online, decentralized nature of the semantic web in which most mapping tasks take place, mapping methods being lightweight should be considered as a necessity, rather than a feature.

Firstly, online applications demand online responses. With mapping being a frequently-required applications, users should not be kept waiting for too long for the result to come back.

Secondly, data-intensive methods might incur extremely heavy burden on the underlying networks. The perspective of the network being inundated with data transfered/exchanged for miscellaneous mapping tasks is awful. It is desirable that mapping be accomplished with just a few transfers/exchanges of information. Apart from the network overload considerations, this relaxation in

[1] http://www.google.com
[2] http://www.w3.org/TR/xquery/

information demand has the additional benefit of making the method applicable in more scenarios, where more data-intensive methods might fail simply because of the unavailability of large amounts of data.

The Need to be Extensible and Self-Improving By *extensible*, we are referring to method's ability of incorporating many/new information sources to be mapped without significantly impairing the performances[3]. When carried out in the Internet scale, mapping might well be conducted between a huge number of sources. In addition, due to the openness of the Internet environment, it is highly possible that new information sources will come into the scene continually. Semantic web-oriented mapping methods should have the extensibility to incorporate these newly-arrived sources.

Self-improving refers to the mapping methods' ability to improve its performance over time. On the one hand, *self-improving* is the natural requirement of *extensible*–it is through the improvement over time that a method is truly extensible. On the other hand, the method's being *extensible* means that it can learn from extended mapping tasks, thus making *self-improving* possible.

3.2 The Interplay of Schema Mapping and Data Mapping

To begin with, our method is based on the observation that in real-life applications such as the *movie-hunting* scenario, schemas to be mapped *do* have overlaps of data instances(See Section 5.1 for an empirical proof). In fact, so far as searching tasks are concerned, this is an ex-ante requirement.

From *Schema Mapping* To *Data Mapping* This side of the interplay is quite clear: without schema mapping, it is difficult, if not impossible, to achieve data mapping.

First of all, without schema mapping, we would run the risk of mapping instances on essentially different attributes. Consider the following situation:

source_A:

title	pro_year	dvd_year
Hero	*2000*	*2001*
Hero	*2001*	*2002*

source_B:

name	shoot_year
Hero	*2001*

Not knowing that *pro_year*, instead of *dvd_year*, actually maps to *shoot_ year*, we can't tell which movie in source_A should map to the movie in source_B.

Even if we could somewhat overcome the adverse effects of mapping instances on essentially different attributes, the prior knowledge of schema mapping will greatly reduce the computational costs—we can then just focus on the comparisons of mapped attributes, avoiding trying all pairwise combinations.

From *Data Mapping* To *Schema Mapping* If we know beforehand that certain instances make presences in both sources of the two schemas to be mapped,

[3] This is different from most current related literatures, where *extensible* means the easiness of attaching new mapping subroutines in a mapping system

then these instances could serve as the joint attention around which schema attributes relationships could be inferred. This comes in several forms:

– Direct string comparison
  - Shared instances usually have identical values for shared concepts of multiple schemas, offering clues to scheme mappings—attributes on which a single instance takes same(or highly similar) values tend to be a map.
  - String comparison also makes complex mappings such as concatenation possible and somewhat more straightforward.
– Numerical attributes mapping
  - Numerical-valued attributes often participate in equation-like complex mappings, due to different scales used(distances in meters *vs.* in kilometers), currency exchange rates(prices denoted in dollars *vs.* in pounds), etc. Such mappings can be discovered provided that we have overlapping instances from which we can suggest equations.

Consider the following situation:

source_A:

title	pro_year	dvd_year	run_time	MPAA
Hero	*2000*	*2001*	111	R
Hero	*2001*	*2002*	128	R
Shrek	*2001*	*2002*	89	PG
Matrix	*1999*	*2000*	100	G

source_B:

name	shoot_year	rate	hours	mins
Hero	*2001*	MPAA G	2	8
Shrek	*2001*	MPAA PG	1	29
Matrix	*1999*	MPAA G	2	8

Applying the above mentioned rationales, we can arrive at the following schema mapping results:

– $title = name$
– $pro\_year = shoot\_year$
– strcat("MPAA",$MPAA$) = $rate$
– $run\_time = 60{*}hours + minutes$

4 The MUTUAL ENHANCEMENT MECHANISM

The mutual enhancement process composes of 5 sub-routines:*Sel_Query_Ins()* selects query data instances from sources to be mapped; *Pro_Mapped_Ins()* proposes potential mapped instances for an incoming query instance; *Pro_Att_Mappings()* proposes attributes' mapping relationships; *GoOn()* decides whether the mapping process should go on; finally, *Decide_Schema_Mapping()* leverages different proposals to arrive at the final mapping result(s).

The *query-propose-decide* mechanism is as follows:

The input parameter *s1,s2* and *iSet1,iSet2* are the two sources' schemas and instance sets respectively.

There are many options as to the implementations of each of the 5 subroutines. Following is a brief description of our current implementations:

– *Sel_Query_Ins()*: Query instances are always sent from the source having the fewer instances[4], they are randomly selected and sent in an exponential way—starting from 5 instance, then 10,20,40... No instance is selected twice as a query instance.

[4] Information such as the size of the instance repository is usually easy to obtain

Algorithm 1 Mutual Enhancement

```
procedure MUTUAL_ENHANCEMENT($s1, s2, iSet1, iSet2$)
    while GoOn() do
        $qIns \leftarrow Sel\_Query\_Ins()$
        for all $ins \in qIns$ do
            $mIns \leftarrow Pro\_Mapped\_Ins(ins)$
            $pMappings += Pro\_Att\_Mappings(ins, mIns)$
        end for
    end while
    $schemaMapping \leftarrow Decide\_Schema\_Mapping(pMappings)$
    return schemaMapping
end procedure
```

– *Pro_Mapped_Ins()*: To be accepted as a potential map, an instance should meet the following two requirements:
  1. Having the largest number of identical values as appearing in the query instance; and
  2. Agreeing in value with the query instance on previously-proposed mapped attributes as much as possible.
– *Pro_Att_Mappings()*: All potential attribute mappings, implied by value correspondences, are proposed. A single mapping can thus be proposed many times by different (proposed)mapped instance pairs. Equation discovery begins when we have more than 3 mapped instance pairs, and is fine-tuned with newly-accepted mapped instances.
– *GoOn()*: The *query-and-propose* process stops when all instances have been sent as query instances, or when there is unlikely to be any more attribute mapping. In our preliminary implementation, we think there is unlikely to be more attribute mapping when no attribute mapping is proposed by 3 consecutive proposed mapped instance pairs.
– *Decide_Schema_Mapping()*: All proposed attribute mappings are retained as a part of the final schema mapping if they don't conflict with others. In the case of conflicts, such as *production year* being proposed to map to both *shooting year* and *release year*, the one supported by more proposals is retained.

5 EXPERIMENTATION AND DISCUSSION

5.1 Experiment Background

Our experiments were carried out with schemas and instances extracted from 6 movie web sites[5]. Since the web sites do not provide ready schemas, schemas

[5] *www.imdb.com, www.allmovie.com, www.hollywood.com, www.eonline.com, www.movies.com, and www.movieweb.com*

are manually constructed by extracting whatever might be meaningful in describing a movie from these web sites. Schema sizes vary from 12 attributes for *MOVIEWEB* to 25 attributes for *ALLMOVIES*, averaging at 20 attributes.

To assess the extent of instance overlap, we got random movies from each of the 6 web sites, and then queried the remaining 5 web sites with that randomly-chosen movie. Results show that there is a significant level of overlap between different sources, averaged at 44.3%, ranging from the low of 27.8% between *HOLLYWOOD* and *MOVIES*, to the high of 83.3% between *IMDB* and *ALLMOVIE*[6]. This is a testimony to our observation in Section 3.2 that real-life scenarios might have high levels of instance overlaps.

We tested with the 6 sources populated with beforehand-downloaded data instances, instead of using all instances available on the web sites. Thus we were able to evaluate performances under different levels of instance overlaps.

To evaluate the precision and recall of our method, we had to decide on what the correct mappings are. Volunteers were asked to list the mapping relationships, and their opinions were then leveraged to arrive at what we thought ought to be the correct schema mapping. About one half of all attributes appearing in the 6 sources participate in schema mappings, leaving attributes particular to a specific source, such as *sound mix*, unmapped. Data mappings are, however, reasonably decided by the two attributes that appear in all of the 6 movie schemas, *title* and *production year*. We assert that the combination of these two attributes serves as a primary key, and suffices for data mapping purposes.

5.2 Experiment Result

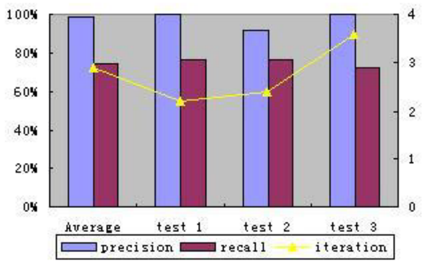
Overall Performance Fig.1 (a) presents our method's schema mapping performances, in terms of precision, recall and iteration numbers. The *Average* columns denote the average performances of all pair-wise combinations of the 6 sources. We list 3 of such combinations(see Table 1). Here we use all the instances we download from the 6 web sites, ranging from 786 to 2419 movies for each web sites respectively, and their levels of overlap roughly reflect the true situation.

It is worth noting that since results might be affected by the querying instances used, for each pair-wise combination, our method is run 5 times, and the results shown here are the mean of the 5 individual runs' results.

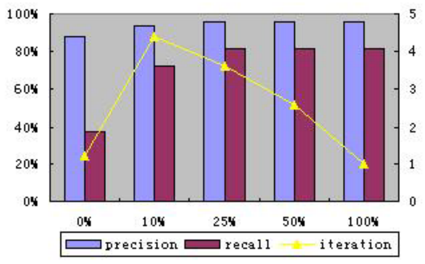
Precision is unanimously high, averaging at 98.3%. This is a natural outcome, in that schema mapping is discovered by comparing overlapping instances' attribute values, and it is rarely the case that many instances all take on same values for two or more different attributes so as to mislead the mapping. The ability to discover equation-like mappings further strengths the precision.

[6] Since query instances are always selected from the smaller repository, the overlap ratio we use here is defined as:

$$\frac{\text{number of overlapping instances}}{\text{size of the smaller repository}}$$



(a)



(b)

Fig. 1. (a)Overall Performances (b)Stability Analysis

Table 1. Selected Results

	sourc1	size1	source2	size2	overlap	precision	recall	iteration
test 1	IMDB	2419	ALLMOVIE	896	83.3%	100%	76.4%	2.2
test 2	EONLINE	1740	MOVIES	1320	60.8%	94%	81.8%	2.4
test 3	MOVIEWEB	1139	HOLLYWOOD	786	41.4%	100%	72.3%	3.6

While precisions of test 1 and test 3 are both 100%, it is only 94% for test 2. This is because that while both *EONLINE* and *MOVIES* have a *Release Date* attribute(which forms a map), the first schema also has an *In Theater Date* attribute that usually takes on the same value as *Release Date*. Thus this attribute is often proposed, wrongly, to be mapped to *Release Date* attribute of *MOVIES* by the *Pro_-Att_Mappings()* sub-routine. If proposed the same number of times as with the correct mapping, the *Decide_Schema_Mapping()* sub-routine will just retain this false mapping, resulting in the lower precision.

Compared with precision, recall for schema mapping is relatively low, averaging at 74.6%. This can be explained by two major causes:

1. Some shared attributes, such as *review* and *movie plot*, are unlikely to take on same values for an identical movie in different sources. Our current implementation just can't find mappings of these attributes.
2. Information on some films might be incomplete. Querying instances all having no value on a particular attribute might probably miss the mapping of that attribute(Following we will give a detailed illustration).

It is somewhat difficult to evaluate the data mapping part of our method. For one thing, so far as the *movie-hunting* scenario is concerned, there are only negligible increases in data mapping in terms of the ratio of the actual mapped instance appearing in the first 3 proposed mapped instances. This is due to the fact that our testing data offers few settings, like the one discriminating between *pro_year* and *rel_year* elaborated in 3.2, that could vividly show the enhancement schema mapping brings to data mapping. For another thing, a major benefit that schema mapping result brings to data mapping is the computational efforts saved, which is hard to quantify.

Stability of the method To assess how our method is affected by the instance overlap level, we tested with instance overlap ratios being 10%, 25%, 50% and 100%. We manipulated the two sides (*EONLINE* and *MOVIES*) to get the desired overlapping ratio, while the absolute size of the two sources were kept unchanged(500 instances for each source). The result, shown in Fig.1(b), is the mean of 5 individual runs.

It can been seen that so far as there *is* instance overlap, schema mapping results in terms of precision(94% $\sim$ 96%) and recall(72.3% $\sim$ 81.9%) are rather stabilized.

The iteration numbers are, understandably, inversely related to the instance overlap level. When there is an 100% overlap, the 5 query instances of the first round suffices the schema mapping task. When overlap ratio is low, it takes more rounds so that enough mapped instances have been identified and these pair-wise instances no longer suggest new schema mappings. Our implementation of sending query instances in an exponential way, while avoiding blindly sending unnecessary large numbers of instances, ensures that the iteration won't be too prolonged. Results show that it works quite well. When the overlap ratio is only 10%, an average of 4.4 iterations is all it takes.

This verifies that our method is quite lightweight. Even when there is relatively low instance overlaps, our method is quick at arriving at the final results, with high precisions and recalls.

An interesting discovery is that, even when there is no instance overlap, some schema mapping still could be found. Some attributes, such as *rating* and *genre*, can only take a quite limited number of different values, so it is highly possible that different instances from two sources take same values on such attributes. Such instances are then proposed as being identical, resulting in the mapping of these attributes. We call such mapping *twisted* mapping. Results also show that when there is no instance overlap, the iteration ends rather promptly(1.2 rounds in our test). This is explained by the fact that counting on attributes such as *rating* as instance mapping criteria usually turns out many *twistedly-mapped* instances, yet these instances could rarely come up with further attribute mappings. So the iteration ends rather promptly.

Table 2 lists each run's specific results between *EONLINE* and *MOVIES* so as to assess our method's sensitivity to different query instances used.

Table 2. Results of Individual Runs

	precision	recall	iteration
1st run	90%	63.6%	2
2nd run	90%	81.8%	3
3rd run	90%	72.7%	2
4th run	100%	81.8%	3
5th run	90%	81.8%	2

The 1st and 3rd runs end up missing schema mapping relationships that other runs do find. The missed mappings are for attributes *running time* and *release date.* A closer look at the instances suggests that many instances of *EONLINE* don't have values on these 2 attributes. If none of the instances involved in the mapping process has values on such attributes, there is no way to figure out these attribute mappings. The number of iteration is varied somewhat for the same reason: If, repeated, incomplete instances are involved, then the iteration ends sooner.

Each of the 4 runs that has precision of 90% turns out 10 attribute mappings, one of which is the false mapping of *Release Date* and *In Theater Date*.

Extendable and Self-Improving While currently, experiments to test the extendable and self-improving characteristics of our method is still under way, here we'd like to show the major ideas of achieving extendability and self-improvement.

As we have shown, our method's performances, especially in terms of the iteration numbers, is somewhat decided by how we select query instances. The more overlapped instances in the query instances, the better. Instead of selecting query instances randomly, we think it is possible to learn from previous mapping tasks about which instances tend to be shared among different sources. For example, previously successfully mapped instances could be recorded so that they could be used as query instances in future mapping tasks, with great chances of reducing the number of iterations.

Another kind of self-improvement stems from our observation that a pair of mapped instances may have different yet similar values on mapped attributes. A movie may be have *"comedy"* as value for *genre* attribute in one web site, yet in another web site, it may be labelled as *"humor"* on attribute *style*. Data mapping may be hampered by such different albeit same-meaning values. If we know that the two moives in fact refer to a single movie, and that attributes *genre* and *style* are mapped, then we can conclude that value *"comedy"* and *"humor"* probably has the same meaning. This information in term may be helpful for future data mapping tasks.

6 FUTURE WORK

At the time of this writing, we have only conducted some preliminary experiments. Following are several experiments we are contemplating to carry out:

– To analyze how incorrect/imcomplete information might affect the performances. Our goal is to alleviate their adverse effects, and to further study how *twisted mapping* could be of help to suggest correct mappings;
– To reduce the number of exchanges of instances so as to make our method more lightweight.
– To further test the extendability and self-improvement of our method.

Currently, when proposing attribute mappings, previously-proposed mapping information is not taken into account. Utilizing such information in an earlier stage, in stead of at the final stage of deciding which mappings to retain, might contribute to performances.

Our method somewhat precludes the discovery of non-leaf attribute mappings, in that non-leaf attributes don't directly take on values. Future efforts are needed to work around this drawback.

While we have shown that schema mapping and data mapping can be carried out in a mutually-enhancing way, we admit that our current implementation is biased toward schema mapping. In future work, we will pay more attention to how data mapping could benefit from schema mapping.

7 CONCLUSIONS

In this paper, we proposed a method that, by simultaneously attacking the twin problems of schema mapping and data mapping, achieves a kind of mutual enhancement between them. Our method is based on the observation that in real-life scenarios, instance-level overlapping level tend to be high. We have shown that this method is well-suited for the semantic web scenario in that it is relatively lightweight and extensible. Preliminary experimental results turned out to rather inspiring. Ongoing efforts are being made to achieve better performances.

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American (2001) 279
2. Musen, M.A., Noy, N.F.: Anchor-prompt: Using non-local context for semantic matching. (2001)
3. Madhavan, J., Bernstein, P.A., Rahm, E.: Generic schema matching with cupid. In: Proc. 27th VLDB. (2001) 49–58
4. Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching. In: Proc. 18th ICDE, San Jose, CA (2002)
5. Doan, A., Domingos, P., Halevy, A.Y.: Reconciling schemas of disparate data sources: a machine-learning approach. SIGMOD (2001) 509–520
6. Kang, J., Naughton, J.F.: On schema matching with opaque column names and data values. In: Proc. of the SIGMOD 2003. (2003) 205–216
7. Wiesman, F., Roos, N., Vogt, P.: Automatic ontology mapping for agent communication. In: Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems. (2002) 563–564
8. Ma, C., Steels, L., Vogt, P., Amyot, R., Press, T.M.: Grounding adaptive language games in robotic agents. (2001)
9. Etzioni, O.: Category translation: Learning to understand information on the internet. (2000)
10. Do, H.H., Rahm, E.: Coma–a system for flexible combination of schema matching approaches. In: Proc. of the 28th VLDB. (2002) 610–621
11. Madhavan, J., Bernstein1, P., Chen, K., Halevy, A., Shenoy, P.: Matching schemas by learning from a schema corpus. In: Proceedings of of the IJCAI-03 Workshop on Information Integration. (2003)
12. Dhamankar, R., Lee, Y., Doan, A., Halevy, A., Domingos, P.: imap: Discovering complex semantic matches between database schemas. In: Proceedings of the ACM SIGMOD Conference on Management of Data. (2004)
13. Cohen, W.W.: Integration of heterogeneous databases without common domains using queries based on textual similarity. In: Proceedings of ACM SIGMOD 1998. (1998) 021–212
14. Sarawagi, S., Bhamidipaty, A.: Interactive deduplication using active learning. In: Proceedings of the 8th ACM SIGKDD Conference. (2002)
15. E.Winkler, W.: The state of record linkage and current research problems. In: Proceedings of of the Survey Methods Section. (1999) 73–79
16. Doan, A., Lu, Y., Lee, Y., Han, J.: Object matching for information integration: A profiler-based approach. In: Proceedings of the IJCAI-03 Workshop on Information Integration on the Web. (2003)
17. R.Guha: Object co-identification on the semantic web. In: Proceedings of the 8th ACM SIGKDD Conference). (2002) 350–359
18. Cohen, S., Mamou, J., Kanza, Y., Sagiv, Y.: Xsearch: A semantic search engine for xml. In: Proceedings of of the 29th VLDB Conference, Berlin, Germany. (2003)