A User-Guided Approach for Large-Scale Multi-schema Integration

Muhammad Wasimullah Khan[1] and Jelena Zdravkovic[2]

[1] School of Information and Communication Technology,
Royal Institute of Technology
[2] Department of Computer and Systems Sciences,
Stockholm University,
Forum 100, SE-164 40 Kista, Sweden
mwkhan@kth.se, jelenaz@dsv.su.se

Abstract. Schema matching plays an important role in various fields of enterprise system modeling and integration, such as in databases, business intelligence, knowledge management, interoperability, and others. The matching problem relates to finding the semantic correspondences between two or more schemas. The focus of the most of the research done in schema and ontology matching is pairwise matching, where 2 schemas are compared at the time. While few semi-automatic approaches have been recently proposed in pairwise matching to involve user, current multi-schema approaches mainly rely on the use of statistical information in order to avoid user interaction, which is largely limited to parameter tuning. In this study, we propose a user-guided iterative approach for large-scale multi-schema integration. Given n schemas, the goal is to match schema elements iteratively and demonstrate that the learning approach results in improved accuracy during iterations. The research is conducted in SAP Research Karlsruhe, followed by an evaluation using large e-business schemas. The evaluation results demonstrated an improvement in accuracy of matching proposals based on user's involvement, as well as an easier accomplishment of a unified data model.

Keywords: Schema Integration, Business Intelligence, System Interoperability.

1 Introduction

A schema represents a formal structure and can be of many types such as database schema, XML schema, ontology description, or domain conceptual description. The schema matching problem relates to finding *the semantic correspondences* between two or more schemas. Semantic heterogeneity arises from differences in naming, structure and the context in which these schemas are being used. In the database field, schema matching is used to merge different relational schemas to produce a mediated schema. In e-business, it may be used to align business documents with varying data structures. In healthcare, record of a same patient may exists in various hospitals needing alignment in order to present a single view. However, this

K. Sandkuhl, U. Seigerroth, and J. Stirna (Eds.): PoEM 2012, LNBIP 134, pp. 203–217, 2012.

alignment comes at a cost. It takes a number of domain experts to manually inspect schemas in order to perfectly align them. Over time, and especially with the proliferation of the Web, number and size of schemas to be matched increases significantly. The complexity forces companies to find the matches manually often with the help of commercial tools such as [1], [2] and [3]. However, pure manual specification of mappings can both be time consuming and error-prone considering the number and size of schemas to be matched in this information age.

In order to overcome this shortcoming, a lot of research has been done over the last decade. [4] were the first to propose treating schema matching as an independent problem. The focus of the most of the research done in schema and ontology matching is pairwise matching [5,6,7,4,8,9,10]. On the other hand, we do not find many examples where schemas are matched *holistically*. The goal of holistic or multi-schema integration is to integrate more than two schemas at once which result usually in creating a mediated schema where all matching elements are represented only once. *Thus holistic schema matching resembles the pairwise matching in a sense that it generalizes the problem from matching two schemas to n schemas*.

While few semi-automatic approaches have been proposed in pairwise matching to involve user, current holistic approaches rely on statistical information extracted from schemas in order to avoid user interaction. The user interaction is thus largely limited to parameter tuning. These n-way approaches are able to avoid user interaction and still match schemas to considerable degree of accuracy due to the fact that their algorithms operate in domains with quite limited number of distinct concepts [11]. The approaches are considered primarily for matching and integrating Web forms. The schemas are small and simple and consist of list of attributes.

The problem arises when dealing with domains where schemas are large and complex. The example of one such domain is e-business, where exist several standards providing common message structures for organizations to exchange business information. For instance, CIDX is a data exchange standard for a chemical industry and RosettaNet for high-tech industry. Organizations adapt these standards to their specific needs. When organizations define mappings to conduct a business together, those mappings are always their own interpretations of the standard. Thus what is valid in one setting may not be true in other setting.

In this study, we present a user-guided iterative approach for large-scale multi-schema integration. Given n schemas, the goal is to match the elements iteratively and demonstrate that a learning approach results in an improved accuracy with each iteration. This results in a unified data model consisting of a set of schemas and a set of correspondences. For the realization of the unified data model, we complement human activities with a data mining technique for detecting similar structures in a repository of schemas. The purpose of the research is to show a mechanism of user involvement in n-way schema matching, benefits derived from the pairing of users and machine, while addressing the issues that possibly arise from a user involvement.

This paper is structured as follows: Section 2 presents our approach to a user-guided large-scale multi-schema integration. Section 3 is devoted to the evaluation of results. Section 4 reports the works related to ours, and Section 5 summarizes our conclusions and indicates the steps forward.

2 Approach to a User-Guided Schema Integration

This section describes the process of the user-guided iterative approach for a holistic schema matching. The main idea is to leverage the system's growing confidence from the user feedback loop, to better rank the matching proposals with each iteration. Figure 1 shows a sample InvoiceTo schema. An XML schema consists of a root, intermediary elements (optional) and the leaves, together forming a hierarchy. Leaves are where actual information within the schema lies. Rests of the elements are used to structure the schema. Here we make an assumption that leaf correspondences are already given by the user, as it is situation with e-business standards (http://help.sap.com/bp_bniv2604/BBLibrary/Documentation/705_BB_ConfigGuide_EN_DE.doc).

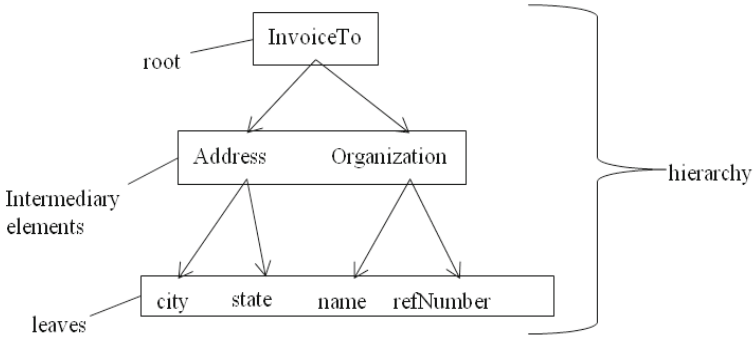

Fig. 1. A sample schema

Based on input schemas and leaf correspondences, system makes proposals to the user with varying degree of confidence as we move up the hierarchy of a schema. For elements higher in hierarchy, less information is available, so system will be less confident than for those which are lower in hierarchy. As user judges proposals and new matchings are found, this information gets added to the system repository and is used for making decisions in subsequent iterations. Consequently, system confidence increases with each iteration and it is likely to produce higher quality proposals. Figure 2 below shows the iterative process.

In the first step, hierarchical input schemas are transformed to linear input of data mining. Then these schemas along with given leaf correspondences act as input to Closed Frequent Itemset Mining (CFIM) [12]. CFIM was originally used for market basket analysis as to detect common patterns such as: 'Shoppers who buy oatmeal and sugar also buy milk.' In the context of this paper, CFIM is applied for detecting similar structures in a repository of electronic business document schemas.

The result of mining is *redundancy groups* (step 4) which are presented to the user for judgment and are therefore also referred to as 'proposals' shown in Figure 3. Each redundancy group is composed of redundant transactions. They are called redundant because the transactions in these groups share the same set of items.

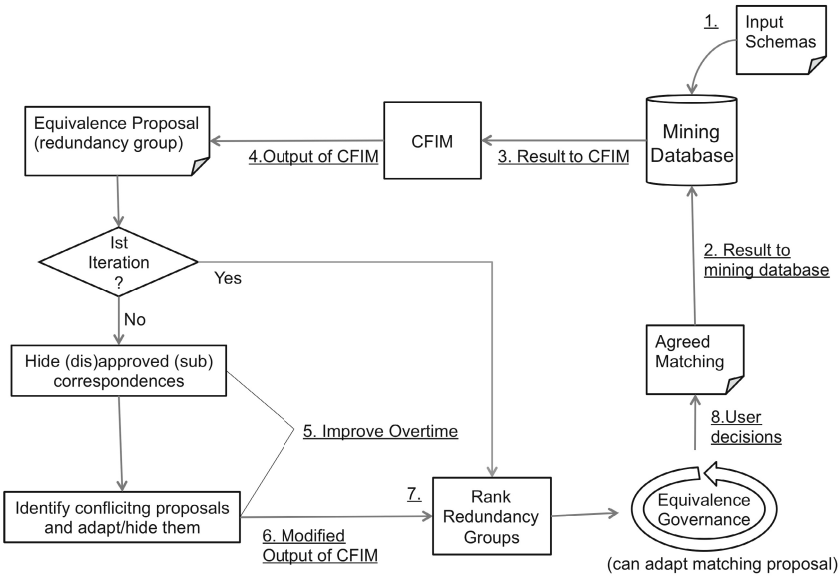

Fig. 2. Iterative approach to finding UDM

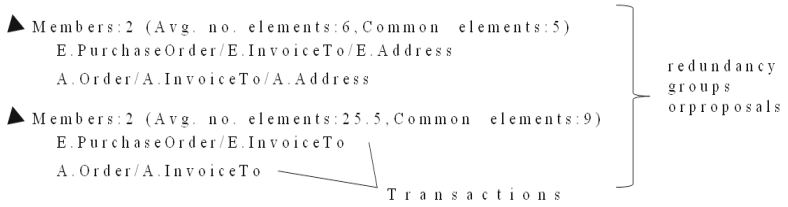

Fig. 3. Output of mining - Redundancy groups (proposals)

Transaction and item are the abstract terms used in frequent itemset mining literature. Input to frequent itemset mining is a set of transactions. Each transaction is associated to a set of items. In XML schemas, we only have elements that form the trees of the type definitions. “Elements” are mapped to the notions of “transaction” and “item” to apply frequent itemset mining. If we want to know the similarity between two elements e1 and e2 of two different schemas S1 (containing e1) and S2 (containing e2), then e1 is a transaction and all inferior elements in S1 are e1’s items, and e2 is another transaction with all inferior elements of e2 in S2 being e2’s items.

The very first iteration will result in presenting mining results as proposed, to the user. However, in subsequent iterations user feedback is taken into account to improve the results.

The ranked results are then presented to the user (Figure 3). The user can judge about these proposals. Judging may involve acknowledging or approving a proposal, disapproving a proposal, or making corrections to a proposal. The user judgment produces agreed matchings (correspondences). The process repeats with user agreed

matchings in previous iteration added to the set of leaf correspondences. The process is repeated until no new agreed matching is found.

In a learning approach, it is natural for a user to expect that judgment will result in increased quality of proposals while the effort required to construct a correspondence will decrease. Thus, we take some measures shown in step 5 of Figure 2 based on user judgment in order to meet the user expectations. These measures include *hiding*, *adaptation of correspondences* and *backtracking*.

Hiding correspondences

The hiding of proposals needs to take place because of the fact that user does not want to go through long list of proposals every iteration, especially the proposals which are already (dis)approved by the user. Thus it helps us reduce the number of proposals presented to a user in each iteration. The hiding takes place in two cases: when a user either approves a proposal or disapproves it. Disapproval of a proposal will result in hiding only that particular correspondence as it does not add to any information about other proposals. On the other hand, approval of a proposal provides a valuable insight. From the hiding perspective, not only the approved correspondence is hidden but any sub-correspondence if it exists is also removed from the list. S is a sub-correspondence of correspondence T if transactions of S is a subset of transactions of T. In order to see the motivation behind hiding the sub-correspondences, consider the example: let A.InvoiceTo, B.BillTo and C.Invoice_To be a correspondence, where each of A, B and C are different schemas. Then any of the subsets of a set {A.InvoiceTo B.BillTo C.Invoice_To} is a called a sub-correspondence. Thus, if user agrees that {A.InvoiceTo B.BillTo C.Invoice_To} is a match, then {A.InvoiceTo B.BillTo}, {B.BillTo C.Invoice_To} and {A.InvoiceTo C.Invoice_To} must be a match too. This implies that the system needs to hide not only {A.InvoiceTo B.BillTo C.Invoice_To} but any of its subsets if present in the list of proposals. On the other hand, if user approves { A.InvoiceTo B.BillTo} and {B.BillTo C.Invoice_To} instead of {A.InvoiceTo B.BillTo C.Invoice_To}, by transitive property, we can then conclude that {A.InvoiceTo C.Invoice_To} and hence {A.InvoiceTo B.BillTo C.Invoice_To} is a match too. In this case also, the system needs to hide not only user approved proposals but any other proposals that match as a result of that approval. However, only approval of {A.InvoiceTo B.BillTo} will not lead to any conclusion about {A.InvoiceTo B.BillTo C.Invoice_To} as relationship between {A.InvoiceTo C.Invoice_To} and {B.BillTo C.Invoice_To} is yet unknown. Thus {A.InvoiceTo B.BillTo C.Invoice_To} in this case will remain the part of list of proposals.

Adapting correspondences

Adaption of correspondence is possible only in the case of approval of a correspondence. In order to understand the concept behind adaptation, assume mining algorithm proposes following redundancy groups:

g1: A.InvoiceTo, B.BillTo, C.Invoice_To
g2: A.InvoiceTo, B.Address, C.Address
g3: A.InvoiceTo, C.Organization
g4: A.InvoiceTo, B.BillTo, D.Address

Let's assume user approves the group g1. That would mean A.InvoiceTo can only form a correspondence with BillTo element from B schema and Invoice_To element from C schema. If the transaction (e,g. A.InvoiceTo) from an approved group (e.g. g1) forms a correspondence with any other schema element (e.g. B.Address, C.Address) besides already approved ones (e.g. B.BillTo, C.Invoice_To), the transaction from an approved group is said to be in conflict. In order to determine a conflict, the algorithm for adaptation determines if any of the transactions (e.g. A.InvoiceTo, B.BillTo or C.Invoice_To) from an approved group exists in some other proposed group (e.g g2, g3 and g4). In our example, A.InvoiceTo exists in all three other groups g2, g3 and g4, while B.BillTo exists in g4. Thus, we can see that A.InvoiceTo is in conflict with B.Address and C.Address in g2 and with C.Organizaion in g3. The transaction(s) in conflict must be removed from the proposal in order to make it valid. The group g2 will then be adapted to:

g2': B.Address, C.Address

Similarly A.InvoiceTo will be removed from g3 which reduces the group to a single transaction. Such groups are removed (hidden) from the list of proposed ones. Assuming there exists fourth schema D, which takes no part in an approved correspondence g1. If any of the transactions from an approved group (A.InvoiceTo, B.BillTo, C.Invoice_To) form a correspondence with a schema with which currently no correspondence exists (D), such a group will not be affected. In our example, g4 will remain unaffected as it could still result in a valid correspondence.

Backtracking
Besides hiding and adaptation, the system allows the users to backtrack. Backtracking means user can retrace the earlier judgments made if the user feels a mistake has been made. One possible way of determining that the mistake may have been made is to check the list of proposals. Approval of undesired correspondences may lead to unwanted list of proposals.

In order to facilitate backtracking, two lists are displayed. One is the list of previously approved correspondences where a user can click on each approved correspondence to see which other correspondences are hidden, when this particular correspondence was approved. The other one is the list of hidden correspondences, clicking on each hidden correspondence will show the correspondence approved due to which this particular correspondence was hidden. Backtracking can be performed on two levels: individual correspondence level and iteration level.

2.1 Ranking of Proposals

The list of proposals generated by mining algorithm is ranked (step 7, Figure 2). To understand why ranking is important, consider the following example:

Table 1. Data Structures

Type	Elements
Customer	ID, Name, DateOfBirth, Phone, City, Street, State, Zip, Country
Partner	ID, DateOfBirth, Email, Fax, City, Country
Party	ID, Name, Email, Address
BuyerParty	ID, Phone, Fax, Address

Table shows the types (transactions) and their corresponding elements (one per row), each belonging to a different schema. Obviously, there are some overlapping elements among the types. These overlapping or common elements lead to redundant types, that is, the types which share same set of elements (not necessarily all). Mining results in set of redundancy groups as shown in the table below:

Table 2. Redundancy Groups (Proposals)

Group	Redundant Types	Common Elements
g_1	Customer, Partner, Party, Buyer Party	ID
g_2	Customer, Partner	ID, DateOfBirth, City, Country
g_3	Customer, Party	ID, Name
g_4	Customer, BuyerParty	ID, Phone
g_5	Partner, Party	ID, Email
g_6	Partner, BuyerParty	ID, Fax
g_7	Party, BuyerParty	ID, Address

This table shows the common types which each group share and the elements those types share among them (one per row) Even from this small example, it can be seen that some redundancy groups are more interesting than others. For example, g_2 although shared among fewer types, has more common elements and is thus more interesting than g_1 which only shares ID of types. In real-world scenario, there can be several such cases. Hence, it is important to rank mining results which orders potentially interesting results higher than others.

A redundancy group g_2 could be more interesting than $g_1$based on three factors. Two of them are evident from the motivating example above: the number of types or transactions in a group and the number of common elements which forms the core of a group. However, it is also important to take uncommon items in a redundancy groups into account as well. If ratio of common to uncommon elements for one of the groups (g_2) is higher than the other (g_1), then g_2 could potentially be more interesting than g_1. The rank of a redundancy group is thus a product of three components: the number of transactions, the number of common elements and ratio of common elements to average number of elements in a group.

In an iterative approach where a user is not likely to go through all the proposals in a single iteration, it is important to rank proposals in a way which makes the task of finding a right correspondence easier for the user. Therefore to facilitate the user we rank elements lower in the hierarchy higher. The reason is that it prevents the user from overwhelming with lot of elements to compare. It can be argued that lower the

element in hierarchy (fewer children), fewer elements are needed to be matched. This not only decreases the user effort but as more and more information becomes available in subsequent iterations (by matching lower level elements) user becomes more confident in decisions regarding the higher level element matching.

3 Evaluation

In this section, we will demonstrate the results and analyze them to verify our claim that user-guided iterative approach for n-way schema matching results in an increasingly improved accuracy. First we will describe the experimental design and settings and then present the results in the light of those settings.

3.1 Experimental Design and Settings

Any schema integration approach is deemed successful by evaluating the quality of correspondences it produces. It is also necessary to examine whether the approach is able to find every possible correspondence to unify the schemas. Since our approach is iterative, we not only evaluate the quality of final sets of correspondences, but also assess the quality of proposals during iterations and at the end of all the iterations. But what is quality? Quality encompasses *correctness* and *completeness* of an integration proposal, which we have defined in the following ways:

Correctness of proposal
Here correctness is defined in terms of an individual proposal. Thus, it determines the quality of a single proposal. Assuming the proposal is composed of 50 transactions and 45 of them corresponds, precision or correctness of a proposal would be 0.9. Thus, if there are 10 proposals in total and assuming 5 of them have precision of 1.0, 3 have 0.9 and 2 have 0.8. Then overall precision of 10 proposals would be 0.93 (5*1 + 3*0.9 + 2*0.8)/10. This measure is used to assess the user effort required to correct the proposal if it is not entirely correct. If 15 or 20 out of 50 transactions need to be corrected, it would make sense for a user to ignore or discard such a proposal.

Correct correspondences-to-incorrect-correspondences ratio
Here the quality criteria refer to how many of the total proposals correspond. Proposal is either precise or it is not, and it measures the overall quality of proposals in a list. For example, there are 10 proposals in a list and 5 of them have precision of 1.0 according to quality criteria defined in point 1. Then the precision would be 0.5(5/10) according to this criterion. This measure is used to take into account the fact that user may not have complete knowledge of domain. If entire list of proposals are correct, user is less likely to make a mistake.

Correctness and completeness of proposal
Besides correctness, this criterion also includes completeness. Completeness means, transactions from each corresponding schema should be a part of correspondence.

Thus correct and complete proposals are the actual correspondence to be identified. Correct but not complete proposals are sub-correspondences.

Based on these definitions of quality we evaluate whether or not we were able to achieve the following goals:

- *G1: have quality proposals.* The iterative process can be seen as a journey towards an ideal UDM consisting of only semantically correct correspondences. To achieve that target, the objective is to have high quality of proposals.
- *G2: have better quality proposals at the top.* Achieving 100% correct proposals cannot be expected because matching is a subjective task. It can be expected that there will be better and worse proposals. At the same time, there will be many proposals when integrating multiple large schemas and we expect the user to only go through certain top proposals before moving on to next iteration. Therefore, ordering of the proposals is of the utmost importance. The proposals which are of better quality must be ranked higher than others.
- *G3: improve overtime.* This is the whole purpose of learning that we gain from user actions. The more information we obtain, the higher the chances that results would improve. Improvement can occur in many ways: generating more semantically correct proposals, the way these proposals are ordered, hiding or removing incorrect proposals, automatic adaptation of proposals, etc.
- *G4: minimize the number of iterations.* To help reducing the user effort as in case of very large schemas, computation can even take days.
- *G5: find a Unified Data Model (UDM).* This is the output that the system is expected to produce. It consists of a set of schemas and a set of correspondences.

For space reasons, we will show the evaluation results for three of the above five goals. We have used 5 XML hierarchal schemas. These schemas, CIDX, Noris, Excel, Paragon and Apertum belong to the purchase order domain. We have used them because on one hand they are fairly complex real-world schemas, and on the other, their mappings are available which helped us simulate the user and verify our results.

In order to avoid manual specifications of mappings, or the acknowledgement or rejection of proposed matches by the user, we have simulated a user which performs these tasks automatically. *The simulated user* approves the first correct and complete proposal from the list of proposals, while ignoring any proposal above it. Thus, the user interaction is minimal, because he only approves the proposed matches. Any incorrect proposals will be ignored and no corrections are made.

One proposal is approved every iteration if there remains an actual correspondence to identify, otherwise algorithm stops. This was necessary to demonstrate improvement during iterations, although algorithm is able to find approximately 80% of actual correspondences to be identified in very first iteration. There were 14 correspondences to identify for 5 schemas and it took 14 iterations, as one correspondence is identified in each one.

3.2 Experimental Results

Have Better Proposals at the Top (G2): Analyze top N proposals presented to the user with respect to quality.

For the five schemas, the number of proposals generated in the first iteration was 75. We have set a limit of 10 proposals for which accuracy is demonstrated. If top 10 proposals' accuracy is high, that would mean the user is most likely be able to find all correspondences within these proposals over certain number of iterations. In order to demonstrate the degree of accuracy for top 10 proposals, we will show where the first correct and complete proposal lies in the list of proposals and what the average quality of top 10 proposals is in comparison to average quality of all the proposals.

Table 3. Data for Average Index of first actual correspondence to identify

Index of first actual correspondence to identify	Best	Worst	Avg.
	1	8	2

Table 3 represents the data for average index of the first correct and complete proposal. Every 2nd proposal was the actual correspondence which user needed to identify. This implies that our simulated user only needed to process 2 proposals in every iteration, ignoring the first and approving the 2nd one. The worst index of 8 means that user should be able to find every correspondence within top 10 proposals.

Figure 4 shows average quality comparison for top 10 vs. all proposals. It can be seen that there is a significant increase in quality of top 10 proposals in comparison to all proposals. On average 4 out of 5 transactions correspond, while 8 out of 10 proposals were correct. This signifies that our ranking algorithm is able to order proposals such that quality proposals are processed first by the user. This also shows that even 56% correct proposals appear on top and hence represent significant quality.

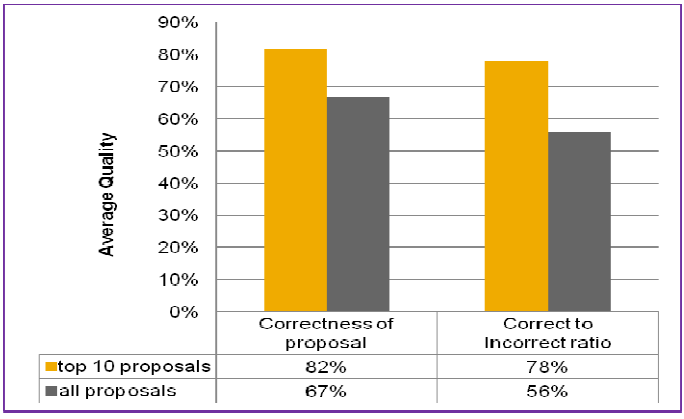

Fig. 4. Average Quality comparison – top 10 proposals vs all proposals

Improve Overtime (G3): Analyze top N proposals in order to demonstrate the improvement in quality based on user feedback in previous iteration.

To demonstrate the improvement, we will show how proposal quality varies over iterations and does the additional information lead to finding of missing correspondences. Figure 5 shows the average quality of top 10 proposals over 14 iterations. The graph shows all quality variants as defined in the previous section. All the three curves show similar trend of increase in accuracy and reaching peak point by the 10th iteration. Ratio of correct to incorrect correspondences increases, as number of iterations increase (see correct to incorrect ration curve). While only 50% of top 10 proposals were 'correct' when only leaf information was available, from 4th iteration to 9th iteration the percentage increases to 90%. In the 10th iteration, there was no 'incorrect' proposal in top 10.

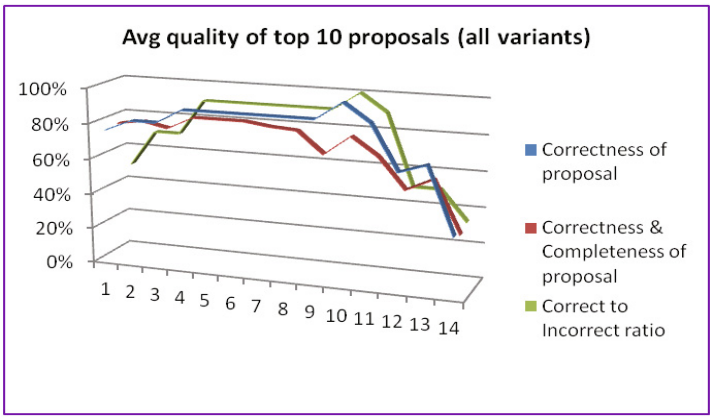

Fig. 5. Average quality of top 10 proposals (all variants)

The trend demonstrates the improvement in accuracy of proposals as additional information is gained through user feedback. However, it can also be noted that there is a falling trajectory after 10th iteration. One possible reason for this behavior is that there are too few correspondences left to identify. Since our simulated user is not rejecting any 'incorrect' proposals, but ignoring them, the number of incorrect proposals may possibly be much higher as compared to correct proposals at this point in time.

The demonstrated improvement in accuracy of proposals also leads to finding missing correspondences during iterations. Figure 6 shows there were 6 missing correspondences in first iteration, which were reduced to zero by the 13th iteration. Thus our improvement measures of hiding and adapting proposals were successful in finding all the correspondences to unify schemas.

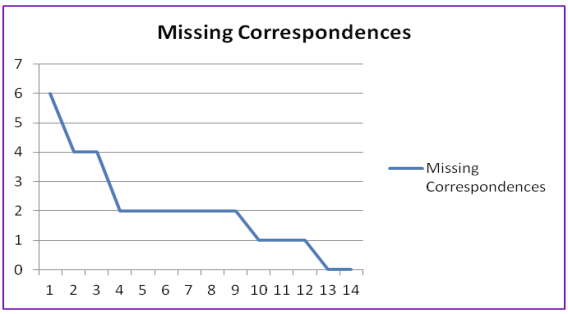

Fig. 6. Missing Correspondences

Find UDM (G5): Analyze unified data model (UDM) with respect to quality in order to assess the user guidance process.

It can be argued that the improvement during iterations, having best proposals at the top and having quality proposals, would mean nothing if due to some missing correspondences unified data model could not be accomplished. The correct and complete UDM is obtained if correct and complete identified correspondences equals number of actual correspondences to be found and false and missing correspondences are zero. Our results show that we are able to identify every correspondence needed to unify schemas and there were no missing correspondences. In addition, our simulated user was 'informed' so no 'incorrect proposals' are expected to be approved. Therefore, we can say that ideal (correct and complete) UDM is obtained.

Summing up, we can claim that the user-guided iterative approach for multi-schema integration leads to increase in precision of proposals as additional information becomes available. Although overall precision of proposals is low, we have shown that proper ordering of proposals with the flexibility to iterate without going through each and every proposal still makes that precision significant. Moreover, the reliance on user feedback means it can affect the results both positively and negatively. The user has the final word when it comes to a decision regarding the proposed matches. If the user approves a seemingly 'incorrect' proposal, perhaps in the user context it was the 'correct' match. However, through backtracking feature, the user can always make corrections, if any proposal was approved by mistake.

4 Related Work

A lot of research has been done in schema matching over the past decade. In [13] an overview of the approaches is given. There are also many surveys reviewing matching and mapping tools for schemas [14, 5, 15] and for ontologies [16]. Iterative approach has been studied so far only in the context of two way schema matching. N-way matching is not solved and remains a big area of research [11].

Two closest approaches to our work are [17, 18] which takes existing mappings and user action history into account in order to incrementally and iteratively match

schemas. In order to address problems associated with single-shot approach, [17] proposed to match schema incrementally, asking the user to select a single element for which top-k matches are generated. The user action history is exploited to rank the match candidates of a selected element. Our approach is similar in a sense that we also generate top-k match candidates and use existing mappings and user action history to rank these candidates. However, we generate top-k match candidates for each element of the schema. It is true that it will result in lot of false positives but we negate their impact by ranking the potentially correct matches higher as our results show and allow the user to iterate without going through each match candidate. Additionally, we not only exploit user action history for ranking of match candidates but also molding the result set. For large schemas, the approach of selecting each element and generating candidates does not seem feasible due to user effort and time involved. In this respect, our approach is heuristic and is much more applicable.

In [18] the user is involved by presenting the best guess results to the user which user can tailor according to their needs. More specifically, user can reject the alignments they found to be incorrect. The rejected alignments are then excluded from the result set in next iteration. In an evaluation, they demonstrated iterative improvement by executing five iterations. For large ontologies or schemas, it appears that their approach is going to take significant number of iterations before final result set can be achieved. Moreover, their approach does not allow for undoing of previous user actions. This is a significant drawback, because if user did accidently reject a correct alignment, the process has to be started all over again. Our objective of improving precision of the result set during iterations is similar, but it is more interactive in a sense that we allow the user to approve, disapprove, and correct the correspondences. Major learning occurs from approval action as it helps reducing the size of match candidates significantly, not only the approved correspondence is hidden but also any sub-correspondences and conflicting correspondences also get hidden. We also learn if the user revokes an incorrect decision.

Finally, the two approaches described above perform pairwise matching. Though some approaches are proposed for holistic schema matching, they are of single-shot type. From the user interaction perspective, they follow a different matching process. The learning occurs through statistical methods without user involvement. This usually limits the success to specific domains, and especially to the domains with limited number of concepts. In our case learning occurs through user involvement which makes it much more generic. It can be expected that our approach performs worse than other holistic approaches in domains where these approaches are expert.

5 Conclusions and Future Work

In this paper, we have described a user-guided iterative approach for large-scale multi schema integration. The approach learns from a user feedback and produces matching proposals with an increased accuracy during iterations. The role of a user is to judge the proposals presented and guide the iterative matching process until a unified data model is accomplished. The matching problem is abstracted to find the closed

frequent itemsets, using CFIM implementation. The output of CFIM is redundancy groups or proposals. Among these proposals, some are more interesting than others. There are three components which make a proposal interesting: the number of transactions, the number of common elements and ratio of common elements to average number of elements. Thus potentially more interesting proposals are ranked higher than others. In ranking of proposals, preference is given to elements lower in the hierarchy and they are ranked higher. This prevents a user from overwhelming with lot of correspondences to compare. Moreover, as additional information becomes available user becomes more confident in decisions regarding the higher level correspondences.

We conducted a comprehensive evaluation of our work using large e-business schemas. In the evaluation, we have proved our hypothesis that the iterative approach leverages the growing confidence from the user feedback loop to better rank the proposals with each iteration. At present we cannot guarantee overall higher level of precision, but one thing where we are confident is that the proposals get more precise than they were in the past, irrespective of the number and size of schemas.

In future, our work can be improved in different ways. Specifically, fragment-based matching as proposed by [8] could be considered in order to solve the problem of execution time that may arise in case of greater number of schemas. Furthermore, more and larger schemas demand an advanced graphical user interface. Interactive techniques proposed by [19] for matching tasks are needed to be taken into account. Finally, a community driven approach where a group of users together (governance board) match schemas is also a possibility [20, 21].

Acknowledgments. This research is conducted at SAP Research Karlsruhe, Germany, under the supervision of Jens Lemcke. It is the part of iGreen Project for providing users with "standardized, industry wide connectivity". To support this, the aim is to build a service network where small businesses can have access to secure e-services. Semi-automatic integration of schemas is used to generate transformations between different schemas. These transformations are then used in a service network to facilitate German agricultural sector.

References

1. SAP Netweaver. Adaptive Technology for the Networked Enterprise (August 15, 2011), http://www.sap.com/platform/netweaver/index.epx
2. Microsoft BizTalk Server, Microsoft BizTalk Server website (August 15, 2011), http://www.microsoft.com/biztalk/en/us/default.aspx
3. IBM InfoSphere, InfoSphere Platform (August 15, 2011),
4. http://www-01.ibm.com/software/data/infosphere/
5. Madhavan, J., Bernstein, P.A., Rahm, E.: Generic Schema Matching with Cupid. In: Proceedings of the 27th International Conference on Very Large Data Bases (VLDB 2001), pp. 49–58. Morgan Kaufmann Publishers Inc., San Francisco (2001)
6. Rahm, E., Bernstein, P.A.: A Survey of Approaches to Automatic Schema Matching. VLDB Journal 10(4), 334–350 (2001)

7. Berlin, J., Motro, A.: Autoplex: Automated Discovery of Content for Virtual Databases. In: Batini, C., Giunchiglia, F., Giorgini, P., Mecella, M. (eds.) CoopIS 2001. LNCS, vol. 2172, pp. 108–122. Springer, Heidelberg (2001)
8. Doan, A.H., Domingos, P., Halevy, A.Y.: Reconciling Schemas of Disparate Data Sources: A Machine-Learning Approach. In: Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data (SIGMOD 2001), vol. 30(2), pp. 509–520. ACM, New York (2001)
9. Rahm, E., Do, H.H., Maßmann, S.: Matching large XML schemas. ACM SIGMOD Record 33(4) (2004)
10. Bernstein, P.A., et al.: Industrial-strength Schema Matching. ACM SIGMOD Record 33(4) (2004)
11. Euzenat, J., Shvaiko, P.: A survey of schema-based matching approaches. Technical report, Informatica e Telecomunicazioni, University of Trento (2007)
12. Rahm, E.: Towards Large-Scale Schema and Ontology Matching. In: Schema Matching and Mapping. Data-Centric Systems and Applications, pp. 3–28. Springer (2011)
13. Uno, T., et al.: LCM ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets. In: IEEE International Conference on Data Mining, Workshop on Frequent Itemset Mining Implementations (FIMI), Brighton, UK (2004)
14. Bellahsene, Z., Bonifati, A., Rahm, E.: Schema Matching and Mapping. Data-Centric Systems and Applications. Springer (2011)
15. Do, H.-H., Melnik, S., Rahm, E.: Comparison of Schema Matching Evaluations. In: Chaudhri, A.B., Jeckle, M., Rahm, E., Unland, R. (eds.) Web Database System and Web-Services 2002. LNCS, vol. 2593, pp. 221–237. Springer, Heidelberg (2003)
16. Shvaiko, P., Euzenat, J.: A Survey of Schema-Based Matching Approaches. In: Spaccapietra, S. (ed.) Journal on Data Semantics IV. LNCS, vol. 3730, pp. 146–171. Springer, Heidelberg (2005)
17. Noy, N.F.: Semantic Integration: A Survey of Ontology-Based Approaches. SIGMOD Record 33(4), 65–70 (2004)
18. Bernstein, P.A., Melnik, S., Churchill, J.E.: Incremental Schema Matching. In: Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB 2006), pp. 1167–1170 (2006)
19. Chen, D., et al.: A User Guided Iterative Alignment Approach for Ontology Mapping. In: Semantic Web Enabled Web Service (SWWS), pp. 51–56 (2008)
20. Falconer, S.M., Noy, N.F.: Interactive Techniques to Support Ontology Matching. In: Bellahsene, Z., Bonifati, A., Rahm, E. (eds.) Schema Matching and Mapping, pp. 29–52. Springer (2011)
21. Rech, J., et al.: Intelligent assistance for collaborative schema governance in the German agricultural eBusiness sector. In: Proceedings of the 12th International Conference on Information Integration and Web-based Applications & Services. ACM, New York (2010)
22. Zhdanova, A.V., Shvaiko, P.: Community-Driven Ontology Matching. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 34–49. Springer, Heidelberg (2006)