A novel framework for the composition of schema matchers

BALAZS VILLANYI, PETER MARTINEK, BELA SZIKORA
Department of Electronics Technology
Budapest University of Technology and Economy
1111 Budapest, Goldmann György tér. 3.,
HUNGARY
martinek@ett.bme.hu

Abstract: - The variety of enterprise schemas has entailed a reasonable interest in those methods, which should resolve this divergence of business entities in integration scenarios. In the case of schema matching, the goal is to find means to identify semantically related concepts. This procedure contributes to the success of the aforementioned task. Several solutions have been presented with reasonable performance qualities and genius behind lying concepts. Our investigations showed that the performance is strongly dependent on the idiosyncrasies of the actual test scenario. The not optimal parameter settings of the algorithm could result in a complete failure on the algorithm’s part. On the other hand if the parameters are set correctly, the performance should significantly improve. Hence the key to success is the optimization of working parameters in the available algorithms. So far, we have introduced a new approach called calibration, which is a pre-run optimization technique providing an optimal parameter set to the process. In this paper we focus on the establishment of a new framework, which also facilitates the optimal composition of the input methods. Just like in the case of the calibration, we endeavor to provide optimization for given scenarios. Other prerequisites, such as the extensibility, scalability and maintainability were also considered.

Key-Words: - schema matching, optimization, algorithm analysis, performance improvement, framework creation

1 Introduction

System integration is of key importance in nowadays’ enterprise life. Some of the behind lying reasons are the diversity of applied systems or the defective, sometimes not even feasible interchangeability of these systems. The age of these systems also varies on a large scale, so in order to invoke functions in these systems different technical requirements should be met. The complexity of the problem is obvious. It constitutes a distinctive scope of research and is treated under Enterprise Application Integration (EAI).

One of the key elements in EAI is the database integration. This a low level manifestation of the integration task, where the communication between systems is carried out on back-end level. In order that this communication functions correctly, the database entities should be matched firstly. This is indispensable, as the essential information represented in both database schemas should be identified to initiate and maintain communication. The problem is treated not on data instance level, but on a more abstract one, where the structure of database is described with meta information. This information about the structure is stored in schema description files. Consequently, the main interest remains in these files. The process of extraction of identical entities, their related information and the subsequent matching of them is referred to as schema matching. The input of this process consists of schema definitions. For the task, only the information gained from these files can be used.

Main matcher types are linguistic, structural and constraint based ones. Methods which make use of semantic and syntactic similarity are classified into the first group, while methods investigating the structure and inner representation of schemas are classified into the second. Algorithms in the third group contribute to the end result with the inspection of constraints in schemas.

The task is fairly complex due to the absolute freedom of designers in schema creation. Although there are some standards and even business scope specific recommendations, the conformity to them remains only options, no one enforces them. As a result, there is serious divergence between enterprise application schemas. Consider for example naming conventions, which would not be such a big issue, if hard to follow abbreviations were not used. Even if abbreviation resolver is at hand, the task is still considerably hard. Besides naming conventions, one should face some challenge with the different entity structure and granularity. Among others, this lack of structure conformity makes the schema matching eligible as a standalone scope of research and distinguishes it from other plan semantic based similarity evaluation method.

There are several solutions, which should cope with these challenges with good performance. Some approaches excel from the other, but the need for further improvement on accuracy is reasonable. The issue is that
their performance is satisfying as it may be, though not so outstanding that human supervision could be set aside in the whole process. In other words an additional human evaluation is always required, which has considerable impact on the costs. Depending on the schema size the intervention required from human evaluator can result a significant superfluous time expense, which have to be reduced. The only way to prevent this is the accuracy enhancement of available techniques. Our focus fell over the enhancement of algorithms.

So far, we have developed a technique which should optimally set the parameters of an arbitrary chosen algorithm for a given scenario[12]. Our approach is two sided. Ones, the reference solution should be approximated by the output, while on the other hand the accuracy measures are maximized by choosing the adequate parameter set. Both approaches have entailed a substantial improvement of the accuracy. In the case of some algorithm though, this improvement did not go beyond a certain border. At that point, we have realized that no matter how optimal the parameter set is, the algorithms bear their own limitations. Nevertheless, these limitations do not appear in every scenario. It also means that the right algorithm for a given scenario is not definitely the right one for the other. Although, their divergence from the average performance is low, it is enough to produce situations, where one algorithm inferior to the other in previous runs suppress this latter one. Consequently the methods should be chosen for a given scenario. Strolling beyond that, we have analyzed the performance of the algorithms in depth to find those elements, or as we refer to them: components, which really distinguish themselves from the other.

The paper is divided into sections as follows. In the next chapter some related works are briefly described. The subsequent chapters describe the matcher composition. Chapter three presents the algorithms that we have used in our researches, while chapter four enumerates the components elicted from them. Chapter five contains some considerations regarding the comparison. In chapter six, we present the consequences gained from the comparison of components presented in chapter four. The composed matcher and the results obtained with it are presented in chapter a seven. A step-by-step definition of the composition framework is defined in chapter eight, while chapter nine contains our future plans.

2 Related works

Several schema matching methods have been introduced, like [7,8,9,10,11]. Among them are really trustworthy ones with quite convincing performance.

At first, let us present [2], which incorporates a unique inner representation method. The vast majority of schema matchers use graph based inner representation. Some of them strongly exploit this graph representation and defines the schema matching process as sequence of operations and transformations of graphs, just like [8]. The undeniable advantage of this is that it is easy to follow and humans treat them with comfort. This is not the most optimal representation, however. The graphs may not use the memory economically. As a remedy, this approach uses Pruefer codes as inner representation. The Label Pruefer Sequence stores linguistic information, while the Number Pruefer Sequence incorporates structural information. It has also a unique optimization consideration, namely compatible elements are collected as a first step and only after that, the comparison process is evaluated. This has significant impact on the runtime needed as it could dramatically reduce the number of comparisons required, provided only the not eligible entity pairs are filtered out in the first step. The construction of the Pruefer sequence does not manifest a problem, either. It is performed as the post-order traversing of the schema graph. The structural matching part is very similar to the one presented in [3]. In our researches this meticulous approach proved to be very accurate. On the contrary to what is recommended in [3], the WordNet vocabulary is not used this time. This fact should also dramatically reduce the runtime of the process. This method should manifest in a more runtime economic alternative to [3].

Authors of [5] propose a framework for schema matching based on learning methods. The process is divided into two parts. The first is called offline preparation, while the second is the online matching phase. In the first phase the most adequate supervised learner is looked for, which is followed by the matching of the similar pairs in the second phase. The system is a somewhat trial and error like procedure as one does not get any clue, which learner to choose first. Some recommendations should be provided, which to choose. This decision could be based on scenario analysis, which urgency further accentuates the necessity of our researches.

We have introduced a technique in [6] to optimize the parameter set of the algorithms. The technique is called calibration and two different approaches are presented. The first one is to approximate the reference result with the output with parameter manipulation, which is called reference approximation. The second one is a direct approach, where the accuracy measures are maximized through the parameter setting. However several experiments with calibration have been performed and it has turned out that algorithms do have their limitation. That is why calibration is highly required in itself as it is, but not sufficient. A new technique which should recompose existing schema
matchers is sought after. This approach could result in wider optimization possibilities.

3 Algorithms used
Just like in the case of the definition of the calibration, we were aimed preserving the general applicability of the approach. The goal was to develop techniques which work under various conditions. Algorithms may have different complexity, component number, parameter number and even input prerequisites and this framework should cope with all of them. That does not mean however that we did not use some specific algorithm in specification, design, implementation and testing phase, quite on the contrary. When selecting from the available methods our right candidates, we especially considered the diversity of them, in order to have the widest range of solutions. We have experimented with the optimal recomposition of three solutions.

One of them is called the NTA [7], which compares the names, the related terms and the attributes of the entities in the schema and assesses their relatedness through scoring based evaluator. The approach traverses recursively the schema graph, which is defined by the relations of the entities. Its peculiarity is given by the attribute comparison, which incorporates recursion. The technique is surprisingly fast, substantially faster than the other candidates. It has also achieved good results, which makes it a powerful candidate.

The second inspected approach is the similarity flooding [8]. It has earned its candidacy with its revolutionary idea. The approach is defined on the presumption that the more similar the entities in the direct vicinity of the compared entities are, the more similar the compared entities themselves are. The input schemas are transformed into extended similarity propagation nets. The iterative propagation of the similarity values is performed along the weighted edges of this and the flooding itself is delimited by specific halting conditions. The idea is fascinating, albeit the results sometimes fall behind that of the other two. It is also runtime efficient. Unfortunately there are only a small number of parameters, by which the calibration could be customized.

Lastly, we have also analyzed the WordNet based matcher presented in [3]. The specialty of this approach is definitely the usage of the WordNet dictionary [4], developed at Princeton University. The dictionary is itself an extended synonym dictionary which has its own classification of words. As the dictionary handles only English words, abbreviation and concatenation must be resolved before the usage, otherwise failure is guaranteed. The usage of the dictionary constitutes at least two drawbacks. The first is the need for preprocessing (if abbreviation, concatenation etc is used), while the second is the considerable runtime surplus. As shown in [2], the vocabulary usage is not even necessarily needed. On the other hand though, it seems to be too obvious that semantic comparison has its advantage over simpler linguistic matchers. This approach also has a complex structural matcher. The evaluation is based on contexts, of which there are three: ancestor, descendants and leaf. The node comparison is performed in all of these contexts with linguistic and complex path similarity evaluation methods. The similarity of the contexts is computed based on these results, which is used to define the node similarity in turn. The approach is indeed complex and requires relatively large number of node comparisons. This manifests a serious growth in the runtime needed as the node comparisons usually require that the external vocabulary be invoked. On the other hand, the accuracy is one of the bests, thus making it eligible candidate. Referring to what is presented in [2], the vocabulary based linguistic comparison is not obligatory.

4 Algorithm components
As earlier mentioned, finding the optimal weighting of these components is in itself not always sufficient. The first step towards defining our new framework for schema matchers’ recomposition is to dissemble existing ones. The need for this step is a stressing one as algorithms are analyzed elsewhere as a whole. Our approach was to identify smallest whole part of the algorithm which is able to perform a comparison. We refer to them as components.

Having disassembled all the algorithms presented in the previous chapter, we have decided that the set should be augmented. The reason for it is our suspicion that a little modification to the original component may result in higher accuracy. Spurred by this idea, we have also defined new components are not used in any algorithms though they resemble to existing ones. This processing of available solutions is an important step. Our experiment showed that the component set augmentation could indeed lead to better results.

The components were classified into categories. This categorization is required as it is obvious that a structural matcher cannot be compared to a linguistic one as their analyzing methods are different. We have further refined the set of linguistic matchers by distinguishing between simple string comparison methods and vocabulary based ones. We felt this distinction necessary as the usage of vocabulary could entail a dramatic enlargement of the runtime needed. One of our answered questions was whether the runtime surplus comes along with a higher accuracy. Based on what was elicited from the three algorithms and what was added and modified to them, we have ended with 20 components. For example a few of them are listed below with brief descriptions:
Linguistic matchers:
- **NTA linguistic matcher**: full point in case of full match and half in case of substring match. Null point otherwise.
- **Prefix/Suffix based matcher for names**: Return the ratio of the common prefixes and suffixes in the names and the name word length. In case of full match this returns 1.
- **Prefix/Suffix based matcher for types**: similar to the previous one, only this is evaluated for types.

Vocabulary matchers:
- **WordNet based word matcher for names**: The similarity of labels is assessed with a dictionary query, which returns the semantic distance between the labels. In this case the query is executed for names and names are handled as a single word.
- **NTA related terms similarity**: A scoring approach which is quite similar to that introduced in linguistic matchers section. In addition to that, the scoring is executed on related terms. For every term in the set the best matching in the other set is sought. After every term is paired, the similarity value is proportioned to the cardinality of the joined set.

Structural matchers:
- **Flooding similarity**: The values returned after the iterative flooding of the similarities in the extended similarity propagation graph.
- **WordNet based ancestor context similarity**: Ancestor context similarity, where label similarities are evaluated using WordNet dictionary.
- **String comparison based child context similarity**: Child context similarity, where label similarities are evaluated using string comparison.
- **Direct ancestor similarity using string comparison**: Similar to the previous one, the difference shows in the application of string comparison instead of WordNet dictionary.

5 Component evaluation
In order to gain a more appropriate algorithm than the original input set a thorough comparison is required. Principally no restrictions apply regarding the means by which this comparison should be executed. However, we recommend the usage of decision support based techniques. We principally used techniques like the decision tree building and the weigh attributing.

Decision trees are particularly appropriate for the comparison evaluation. They are easy to understand and to evaluate. A pleasing feature is the tree pruning, which makes it applicable even by very large component sets. The number and the place of occurrences of component nodes deliver the result of the comparison. It is pretty straightforward and it should perfectly fit this need as we are only interested in the relative performance of the components.

Another technique we often used at evaluating the performance is the attribute weighing. Although no tree pruning like feature is available, with the help of some adequate visualization, the result is fairly easy to acquire. There are also a lot of alternatives to choose from based on what requirement the analysis should fulfill. We have obtained promising results with Gini index and information gain ratio based weighing. It is worth to try several techniques and compare their output. In our experiment, there were occasions where all of them showed nearly the same result. Based on the result, the decision which evaluator to choose can be done. In a very few occasions, where the discrepancy between result was not substantial, we used a weighted average approach rather. That is to say, we took into account the results of both attribute weighing technique, but with different weights.

In the case of the decision trees, the pruning and relative position of the nodes enables better the evaluation. The accent falls on the relative vantage of the components, the comparison should be done based on thorough considerations. The accuracy relation of the nodes to each other in the tree is easier to obtain. Nevertheless, attribute weighing also provides this indispensable feature. If rendered on diagrams the evaluator is able to easily comprehend these relations. This does not mean, however, that the neural networks should be discarded. They incorporate wonderful evaluation ability, only the other methods are more apt for this kind of task.

6 Comparison result
No strict rules apply regarding the optimal number of comparisons. The result and consequences of the individual comparisons should be aggregated and if further comparisons do not deliver new ones, than the experiment shall be terminated. Should the experiment lead to some kind of ambiguity, than the reason must be uncovered and eventually be resolved. In our research grave discrepancies did not come forth. This kind of grave discrepancy would be that one technique particularly estimates a component particularly valuable, while the other renders worthless. Normally, this contradiction shall not happen.

Based on the components presented in chapter four and on the considerations presented in chapter five, we have obtained the following comparison results:

- The related terms play distinguished role. Based on the output of all component evaluator technique, they are the most valuable, provided the entities are supplied with related terms.
- Synonyms, antonyms, types and paraphrasing terms of entities are essential in concordance with the previous point. Unfortunately the sufficient quantity and quality of related terms is rarely the case. At best, only description is provided, which is still not a related terms set.
- The vocabulary based method can be substituted with an appropriate syntactic based one. This is a relieving factor as the potential for runtime saving is immense.
- Context based matching was the best among structural matching. They clearly surpass other techniques. Used with non vocabulary based matching, they are runtime efficient as well.
- Ancestor context based matching excels somewhat from the other two. Leaf context matching on the other hand is a slight underdog, while child context based matching is outshined by attribute matching.
- Attribute matching is in itself hard to define as it involves the recursive repetition of other techniques. Our experiment concluded that the technique is in itself a valuable one and it is seemingly a more elaborated alternative to child and leaf context based evaluation.

7 Composed matcher
Having performed all the necessary comparisons and subsequent evaluation of the results, all the necessary requirements are met to define a new technique which is foreseeable has more potential than its donor ones.

However, the task does not only consist of the selection of components, but of finding of the proper parameter setting as well. For this task can be used the method called calibration [6]. Experiments proved the key role of this step. Note that our final results presented in this section are attained with calibration involved.

We present the composed matcher constructed according to the conclusions presented in chapter six. We have also added some self optimization aspects, so that the algorithm consumes only the runtime absolutely necessary.

Prefix/suffix based matching is applied as linguistic matcher. Taking into account the distinguished role of the related terms comparison and the fact that this set is not always provided, we have decided to opt for an automatic choice between available methods. The composed matcher examines whether the related terms set is available. Only if this is indeed the case, does it use the related term comparison; if not, then they invoke the vocabulary. This choice does not involve any human intervention and saves runtime automatically. As structural matcher, we have implemented the recursive method defined in the attribute matching with context based evaluation. This solution seemed to be reasonable according to what is experienced during component comparison. In conclusion, the approach involves syntactic, semantic and structural matchers, where the parameters are set using f-measure maximization [6].

Several experiments have been conducted both on test schemas and on real life ones. Our goal was to obtain the highest f-measure values possible. The table below summarizes the averages and the divergences of the attained maximal f-measures in test schemas:

<table>
<thead>
<tr>
<th></th>
<th>CM</th>
<th>NTA</th>
<th>SF</th>
<th>WN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>1</td>
<td>0,81</td>
<td>0,44</td>
<td>0,72</td>
</tr>
<tr>
<td>Divergence</td>
<td>0</td>
<td>0,4</td>
<td>0,26</td>
<td>0,33</td>
</tr>
</tbody>
</table>

Table 1. F-measure average and divergence

The table uses the following abbreviations. CM denotes the composed matcher, while NTA is the matcher with the same name[7]. SF marks the similarity flooding [8] and WN is the WordNet based matcher [3]. The table shows us that the new matcher which consists of the most accurate components of the others performs better the original. This result is provided as the solution for a scenario where originally the best accuracy had not been achieved by any input methods. The components were selected heeding the comparison results on schemas on which the end result is measured. On other schemas this values may differ somewhat, but in those scenarios the construction of the composed matcher might worse to be reinitiated.

8 The framework
Based on what has been presented so far, the definition of the framework can be formulated. Of course, the comparison results and the ideally composed matcher may differ significantly in other experiment scenarios. The whole process can be divided into following steps.

1. **Initial algorithm set definition:** This step involves a thorough survey among available methods and a subsequent selection of those, which covers the widest range of implemented principles. It is the input algorithm set in the formula below.

   \[
   A = \left\{ a \in I \mid \min_{I \setminus A} (a \cap A), a \in I \right\}
   \]  
   \hspace{1cm} (1)

2. **Decomposition:** Having chosen the input algorithm set, they shall be dissembled in order to acquire components, which can be used as building stones for the new one. D is the decomposition function in the formula below.

   \[
   C = \bigcup_{A} D(A)
   \]  
   \hspace{1cm} (2)
3. **Composition set augmentation**: Available components might not be the best. Modified versions of them may worse to analyze or even the definition of completely new ones should be considered.

\[
Z = \{ z \in Z | \exists i: z \sim C_i \}, C = C \cup Z
\]  

(3)

4. **Component evaluation**: Using diverse techniques, the components have to be compared. This step is best described as a competition between them, where the victor is the one, which is the more accurate while generating only the smallest runtime surplus. U is the accuracy function, while C is the cost function in the second formula below.

\[
\sum_{i=0}^{\infty} \sum_{j=0}^{\infty} \text{Compare}(C_i, C_j)
\]  

(4)

\[
S = \left\{ s \in C \mid \min_{c} \{ \max_{u} (U(s)), \text{min}_{c} (C(s)) \} \right\}
\]  

(5)

5. **Matcher composition**: Based on the consequences gained in the previous step, the components from which the new matcher is to be built are defined. The new matcher definition should not be a mechanical assembly of the optimal components. The optimization possibilities should be noted and then should be involved if possible.

6. **Matcher calibration**: This one last step may be of the same importance as the previous ones added together. Its method is described in [6] in details.

9 **Conclusion and future work**

In this article the creation of a new, optimized framework for schema matching composition is presented. The composition process takes into account several aspects and evaluation results that are present in schema matching scenarios simultaneously. The framework provides an approach which involves many automated steps. The potential of these ideas best harnessed if used as a base, and should be tweaked with further tricks.

We plan to use this novel approach under various conditions and see how it behaves if only a small set of training schemas are available. We are also curious how well the conclusions gained at a component evaluation scenario apply to others. This is of key importance. Should it turn out that beyond a certain training schema size we are able draw conclusion in general, our focus shall fall on the research of these general applicable composition rules. That would result a general rule set, which should give directives which component to compose in order to attain best results on a particular schema.

This work is connected to the scientific program of the "Development of quality-oriented and harmonized R+D+I strategy and functional model at BME" project. This project is supported by the New Hungary Development Plan (Project ID: TÁMOP-4.2.1/B-09/1/KMR-2010-0002).

References:


