A framework for schema matcher composition

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: - Enterprise schemas tend to be different, which is the key issue when the seamless communication between systems is of utmost importance. One solution could be the development of standards which then could be enforced, however, vendors seem to be reluctant to comply with them and communication between existing and legacy systems still remains unsolved. Other solution could be schema matching, which resolves the matter on data level and the process do not require vendors to adhere to any kind of predefined schemas. The task is very complex on the other hand, even for human evaluators. Some of the solutions aired so far are fairly promising, however, their accuracy varies. Our goal was to find means by which the results could be enhanced. We have been focusing on the development of solutions which do not change the concept of the algorithms, but fine-tune them so that they achieve higher accuracy. Our experiments showed that the results of the matchers may vary on a large scale depending on the actual parameter settings. It has also turned out that the parameters should set for each scenario individually, as the best results are warranted only this way. In this article, we present a general approach for optimally dissembling existing solutions, and combining some of the resulting components in a way that the new matcher supersedes the donor ones. The composition and the optimal parameter setting combined provide a framework, which is capable of an enhanced performance. Improved accuracy lessens the need for the follow-up human supervision.

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

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 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 with meta information described. This information about the structure is stored in schema description files. Consequently, the main interest remains in these files, among which the most wide-spread format and standard is the XML Schema Definition (XSD). It has also the benefit that schema definition itself is a valid XML (Extensible Markup Language) and this is the reason why it is easily handled with simple XML processors, which makes it from one side comfortable and may cut down on the runtime costs at the same time.

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 the 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. This third group is vital as it may be, but is more than less omitted from algorithms, their role is sometimes neglected.

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 these 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. Beyond 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 to average performance. Some approaches excel from the other, but the majority needs to be further improved on accuracy. The issue is that their performance is satisfying as it may be, though not so outstanding that human supervision could be set aside. In other words an additional human evaluation is always required, which has considerable impact on the costs. It results in a huge extra runtime cost (generating occasionally even longer runtime than that of the algorithm), on the other hand the general cost of the human resource should also be taken into account. Furthermore, this factor is proportional to the schema size and the targeted accuracy. In the end, significant superfluous time expense could be generated, which have to be reduced. The only way to achieve this objective is the accuracy enhancement of available techniques, in order that human supervisor not be necessary. Our focus fell over the enhancement of algorithms and we also made efforts to remain on a general level.

So far, we have developed a technique which should optimally set the parameters of an arbitrary chosen algorithm for a given scenario. 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 problem is not definitely the right one for the other. Although, their divergence from the average performance is low most of the time, 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.

Nearly in all related researches, methods have been analyzed as a “black box”. They performance were measured under highly optimistic conditions. In the past, our goal was to compare them fairly and unbiased. Now we were eager to gain insight in those methods with intension to obtain the ability to explain their performance results. Treating the algorithms as black box did not fit our objectives. We have decomposed them in to smaller parts, called components. Obtaining these components was only the first step. Several unanswered questions have emerged additionally. Among those were the exhaustiveness of the gained set and the adequate implementation of the components. By this latter we understand for example whether usage of external vocabularies and other sources are really required. Namely, these latter methods consume huge runtimes that is why the possibility of their substation with simpler syntactic based evaluators is much desirable (of course by keeping the original method basically).

Having performed some decision support based performance tests on the identified components, some new consequences emerged. It turned out, that certain types of components clearly outperform the other. Among these, some components did not get the attention that they deserve. Furthermore, these tests could be executed only on given scenarios, defining the optimal composition for the given task. In order to manifest this optimal composition we have defined a framework.

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 elicited 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 [2,3,5,6,8,10,11,13,14,15,16,19]. Among them are really trustworthy ones with quite convincing performance, but unfortunately some of them are really inferior to the others.

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 [11]. Most of the times, this inner representation is the Document Object Model (DOM) of the XSD, but other graph based inner representations are not unheard of. It has the undeniable advantage 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 [8] propose a framework for schema matching based on learning methods. Using learning techniques for schema matching is not a new idea in itself, but this framework is still a clever one. 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 should be based on scenario analysis, which urgency further accentuates the necessity of our researches. If this directive is provided for the choice, the framework should significantly improve on efficiency.

We have introduced in [9] a technique 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. We have concluded that neither of them is better than the other, the ideal is the one that better serves the actual goal of matching accuracy. 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 should 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 [10], 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 [11]. 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 idea is simple, but genius. In order to harness this presumption, 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 net. The iterative flooding is delimited by a halting condition, defined as either an iteration number or as a difference threshold between iterations. 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. That is why the omission of vocabulary based would have been a mistake. This approach also has a complex structural matcher. The evaluation is based on contexts, of which there are three. The ancestor context encompasses all the of ancestors up to the root, the child context is defined as direct descendants of the node, while the leaf context encompasses all nodes of the sub-graph the root of which is exactly the node in question. The node comparison is performed in 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. We have decided to analyze the method both with and without vocabulary invocation.

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 dissembled 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.

Components are 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 listed below with brief descriptions of their evaluation method:

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.
- *Prefix based matcher for names*: Return the ratio of the common prefixes in the names and the name word length. In case of full match this returns 1.
- *Prefix based matcher for types*: similar to the previous one, only this is evaluated for types.

Vocabular 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.
- *WordNet based word matcher for types*: In this case the similarity is assessed for types, but they are still handled as one word.
- *WordNet based sentence matcher for names*: Names are divided into single words and similarity is returned for the sentence.
- *WordNet based sentence matcher for types*: works similar to the previous one, only the dictionary query is performed on types.
- *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:

- *NTA attribute similarity*: the recursive method traverses the schema graph and uses the similarity values of the lower levels for a given node similarity calculation. If similarity value for a given node pair is not available (e.g. simple and complex type comparison), the algorithm uses simpler means to return the similarity
- *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.
- *WordNet based child context similarity*: Child context similarity, where label similarities are evaluated using WordNet dictionary.
- *WordNet based leaf context similarity*: Leaf context similarity, where label similarities are evaluated using WordNet dictionary.
- *String comparison based ancestor context similarity*: Ancestor context similarity, where label similarities are evaluated using string comparison.
- *String comparison based child context similarity*: Child context similarity, where label similarities are evaluated using string comparison.
- *String comparison based leaf context similarity*: Leaf context similarity, where label similarities are evaluated using string comparison.
- *Direct ancestor similarity using WordNet*: The similarity of father nodes was also inspected. In this case WordNet was applied once again.
- *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 is easy to acquire. There is 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, however not always. In this latter scenario further analysis should be conducted which targets the reason of this diversity. 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.

Neural networks also provide a distinguished alternative in component comparison techniques. Although they are wide spread and used in several scenarios, we preferred using the methods listed above. Experiments showed that they are appropriate for the component comparison task and their delivered result allows better evaluation. 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 the 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.

To exemplify the ideas presented so far, we will go

into the details of an experiment we conducted. As first we built a C4.5 decision tree [1] in order to identify the most relevant algorithm components. The actual component set and the node distance from root were considered as the indicator of relevance.

On the next diagram you will see one of the decision trees. The diagram shows that related term similarity are the most important component. This came as a surprise, but little wonder if one considers that the related terms were a bit too optimistic chosen. This result pointed out the necessity for good quality related terms which so far has been underestimated.

Looking more in depths, one can observe the prime position of the ancestor context similarity and what is more the string based one. This means that a good string based matcher can keep up with a dictionary based one. This has significant impact on the runtime. By using the methods presented in the WordNet based structural matcher [1] with simple string matchers one easily turns down the runtime outlay.

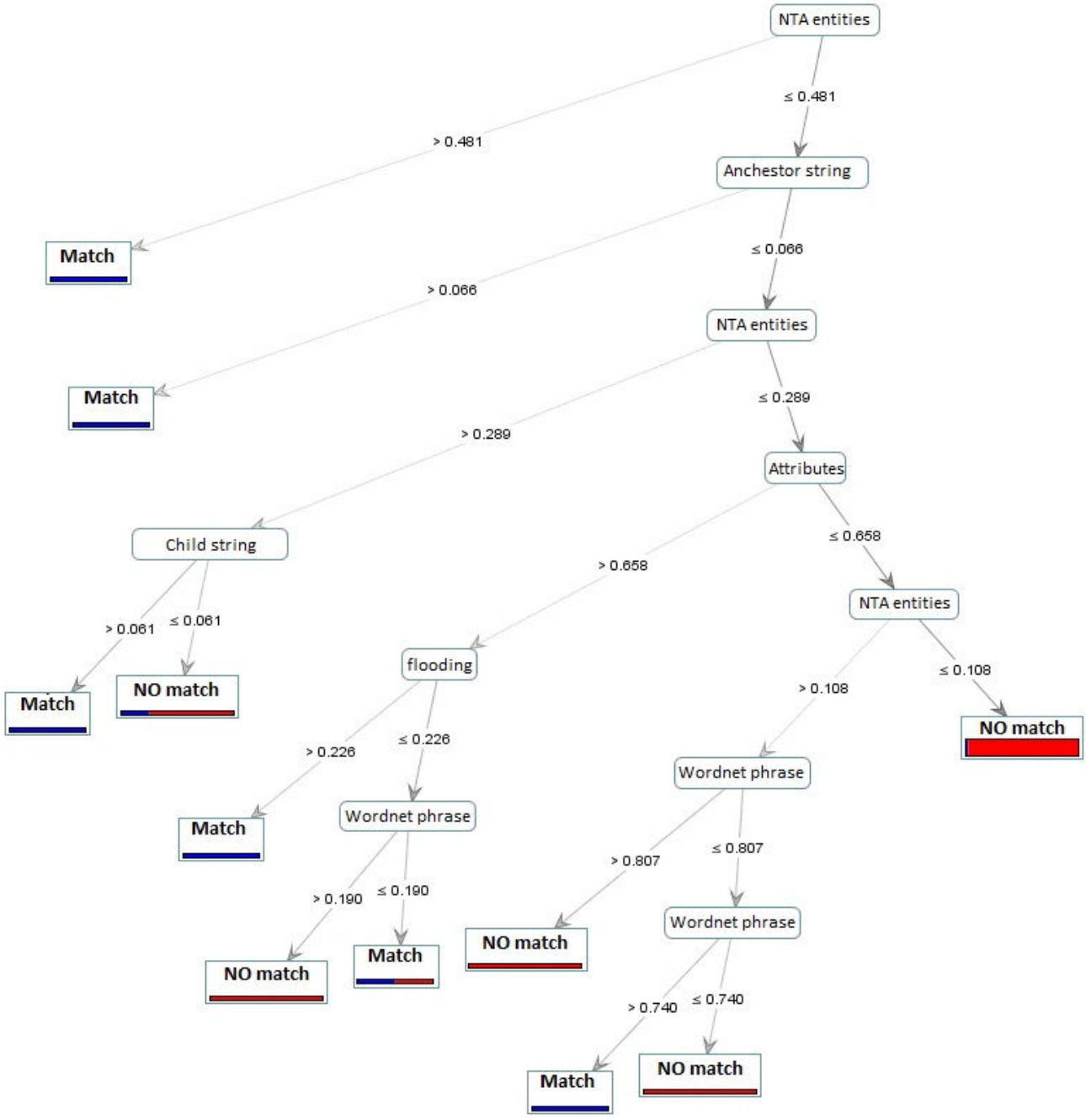

Fig. 1.
Decision tree of the similarity components

You will also find the NTA attribute and similarity flooding nodes, nonetheless not the closest vicinity of the root. This fact can imply they are suppressed by some of the more trustworthy components.

It has also become fairly obvious that the WordNet dictionary based similarity component is still a good choice, but one should opt for the sentence matcher for names. This result eliminates other WordNet based techniques as they are not even shown on the graph.

The next technique was to use attribute weighting and decide on the trustworthiness of each components based on several indicators.

The first weighting was performed using the information gain ratio as indicator. This delivered quite similar results to that of the decision tree. The most promising component turned out to be the related term similarity. The prefix/suffix based linguistic matching came up as second while the WordNet based matching also prospered. According to our results the ancestor context similarity using string comparison was the best among the structural matchers, this gave the most accurate results. Furthermore, it was also to be observed that attribute similarity and similarity flooding summed are weighted nearly the same as the leaf and child context similarity summed. This latter two are somewhat underdog to ancestor context similarity and they carry approximately the same amount of information.

The conclusions are in concordance with the Gini index [1] based weighing. The highest weights were given to the related terms similarity, the WordNet based sentence matcher for names and the prefix/suffix based similarity (both for names and types). The Gini index accentuated the importance of the child context and it also preferred it to the leaf and the ancestor context. Regarding the other algorithms the NTA attribute similarity overtook the similarity flooding by far. String based techniques were preferred to WordNet based ones, which is an unexpected outcome, not to mention runtime benefits.

In every case the linguistic and structural matchers on the whole were weighted nearly the same. On the other hand, some components were found to be completely useless. Among these can be listed the NTA linguistic matcher or the exclusively prefix based matching. The WordNet based techniques do not thrive either if they are not used as sentence matcher. The reason behind can be the multiword denomination of the entities, in which case these labels cannot be interpreted.

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. Consequently, we shall not forget about inspecting when analyzing its accuracy whether the result comes from the technique itself or the other involved in recursive repetition. 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.
- Among non vocabulary based linguistic matchers the prefix/suffix based comparison prevails. It is small wonder considering that this particular linguistic matcher is most complex one.

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 foreseeably has more potential than its donor ones.

However, the task does not only consist of the selection of components, but of the proper parameter setting as well. For this task can be used the method called calibration [9]. Experiments proved the key role of this step. Omitting this final step, the new matcher is presumably completely useless. 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.

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:

	CM	NTA	SF	WN
Average	1	0,81	0,44	0,72
Divergence	0	0,4	0,26	0,33

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[10]. SF marks the similarity flooding [11] 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. In this article we have defined a matcher which serves our goal best. 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. I is the input algorithm set in the formula below.

$$A = \{a \in I | \min_{I-A}(a \cap A), a \in I\} \quad (1)$$

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.

$$C = \bigcup_{A} D(A) \quad (2)$$

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.

$$Z = \{z \in Z | \exists i: z \sim C_i\}, C = C \cup Z \quad (3)$$

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

$$\sum_{i=0}^{n} \sum_{j=0, j \neq i}^{m} Compare(C_i, C_j) \quad (4)$$

$$S = \{s \in C | \max_{C}(U(s)), \min_{C}(C(s))\} \quad (5)$$

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. If parameters are not optimal, the new matcher may render completely useless.

In other words, the first task should be the thorough investigation of existing solutions. This step is of key

importance for obvious reasons. While aiming at finding the best composition of existing methods, the task is only achievable, if the most potent matchers serve as input. It is also a good test for new ideas: if they are also enlisted among the inspected solutions, their true performance compared to the others may be unveiled. On the other hand, it is also strongly recommended that only the best schema matchers should be involved. Although weaker competitors may not matter, they unnecessarily complicate the whole task. While assessing the quality of solutions, a good performance indicator could be the results attached by authors, the prior knowledge and experience with similar solutions or even the perfunctory implementation and test run of the algorithm, called "shallow test".

The collection of donor solutions is followed by the prior analysis and decomposition. A prior analysis is needed in order that the details about the algorithm become clear. One reason for this is the need for the detection of similar and different components. In our praxis, nearly the same components were identified by different solutions sometimes. This fact leads to the conclusion that one of them will not be necessary. Of course, the components do not have to be the exact same, the same behind lying concept may justify for the omission of one of them. No exact rules apply for this decision, so it can be hard sometimes. This prior inspection of components also entails the better understanding of the algorithms; even the set of the most capable components can be envisioned. Nevertheless, the main task in the second step is to decompose existing solutions. Most of the time, this task is pretty straightforward, although it may become unobvious. The goal is to find elements or sub-routines, which are part of the algorithm and can be defined as stand-alone components. A good example for component could be the Terms similarity evaluator in the NTA algorithm [11]. Basically, every time when there are some sorts of weights involved in the solution, which is the case most of the time, the components can be defined as the weighted similarity evaluator. Note that this is the easiest way the extract components, but more complex scenarios can also emerge. The objective is to find to smallest units possible which can be handled separately.

Having said some words about the pre-filtering of components, it is also crucial to compensate for the loss. The idea is to define new components based on the existing ones. Being inspired by concepts, several unseen approaches may be invented, or a new component can be defined based on existing ideas. Sometimes the ideas combined into a new component have a better performance than each separate. An example could be the context based matching, where the vocabulary based elements are substituted for syntactic based ones. Our experiments clearly showed that the accuracy do not deteriorate, while the run-time efficiency improves hugely.

Component evaluation is probably the most significant part of the recomposition process. As already detailed in chapter five, we propose several techniques to make the comparison. The set of technique is comprehensive, but not exhaustive though. It may be augmented by arbitrary chosen other methods which qualify for a complex and relevant evaluation. Nevertheless, the mentioned comparison techniques gave us the best results. We concluded that methods originating from the decision support are the most trustworthy. The aim at this step is to find means to compare the components based on their accuracy produced on the input schemas and create a rank among them subsequently. In our view, the most unbiased way to do this is to utilize several techniques and then sum up the results gained. Undeniable, the most potent schemas are the ones which were elected as trustworthy unanimously by the majority of comparing techniques.

Based on the results gained in the previous step, everything is ready to create an enhanced matcher. Simply take the components and combine them into an enhanced matcher. One should only pay attention to the balance of the types of matcher. This is one the main reason why a previous category definition of matchers was needed. If a component clearly excels, then other component may be weighted less or even omitted, although this latter case is far from typical. On the other hand, tend to avoid to usage of more than one technique from a single category, as they may interfere. For every test scenario, different results can emerge, so even if similar results have been produced by components in the same category, it is highly recommended to refrain from the collective usage of them. Furthermore, there is no point in it if a component has high accuracy and is not superseded by the others. As presented in chapter five, the components can be combined conditionally. We combined the WordNet based matcher with the terms based matcher in this way. Namely, if there is no related terms set defined, then and only then execute the WordNet based matching. The behind-lying consideration is as follows: this latter is always feasible, although clearly runtime consuming. The former not only turned out to be the best among syntactic based techniques, but even runtime efficient. The problem is that the related term set is not available most of time. The rest is obvious. As this example shows, the composition should be made using considerations about the various scenarios. A clever approach can cut down immensely on the runtime needed.

As the last and maybe most vital step execute the calibration proposed in [9]. This may be regarded as the cutting edge between the useless results and the best results of the composed matcher. Without this one final

step – the proper parameter setting – the results may deceive the contemplators, as the matcher may perform suboptimal. We have also shown that only this way is the unbiased comparison of matcher possible. Consequently, all the results presented in chapter seven are that of calibrated matchers.

9 Conclusion and future works

In this article a new 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. This framework provides an approach which involves many automated steps. Nevertheless, designers' consideration shall not be set aside. 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 training schemas 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.

References:

[1] J. Abonyi, *Adatbányászat a hatékonyság eszköze*, Computer Books, 2006

[2] A. Algergawy, E. Schallehn, G. Saake - Improving XML schema matching performance using Prüfer squences, *Data & Knowledge Engineering*, Vol. 68, 2009, pp. 728-747.

[3] A. Boukottaya, C. Vanoirbeek, Schema Matching for Transforming Structured Documents, *Proceedings of the 2005 ACM symposium on Document engineering*, 2005, pp. 101-110.

[4] Cognitive Science Laboratory, *WordNet - a lexical database for the English language*, at http://wordnet.princeton.edu/

[5] Y. P. Chen, S. Promparmote, F. Maire - MDSM: Microarray database schema matching using the Hungarian method, *Information Sciences*, Vol. 176, 2006, pp. 2771-2790.

[6] T. Erl, *Service-oriented architecture: concepts, technology, and design*, R. R. Doneilly, 2005

[7] Hong-Hai Do, Erhard Rahm, Matching large schemas: Approaches and evaluation, *Information Systems*, Vol. 32, Issue 6, 2007, pp. 857-885.

[8] B. Jeong, D. Lee, H. Cho, J. Lee - A novel method for measuring semantic similarity for XML schema matching, *Expert Systems with Applications*, Vol. 34, Issue 3, 2008, pp. 1651-1658.

[9] P. Martinek, B. Villányi, B. Szikora - Calibration and Comparison of Schema Matchers, *WSEAS Transactions on Mathematics*, Vol. 8, Issue 9, 2009, pp.489-499.

[10] Martinek P., Szikora B., Computational Requirement of Schema Matching Algorithms, WSEAS Transactions on Information Science and Applications, Vol. 6, Issue 8. 2009, pp. 1412-1422.

[11] P. Martinek,B. Szikora, Detecting semantically related concepts in a SOA integration scenario, *Periodica Polytechnica*, 2008

[12] S. Melnik, H. Garcia-Molina, E. Rahm, Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching, *Proceedings of the 18th International Conference on Data Engineering*, 2002, pp. 117-128.

[13] J. Nathan Foster, Michael B. Greenwald, Christian Kirkegaard, Benjamin C. Pierce, Alan Schmitt, Exploiting schemas in data synchronization, *Journal of Computer and System Sciences*, Volume 73, Issue 4, 2007, pp. 669-689.

[14] H. Nottelmann, U. Straccia - Information retrieval and machine learning for probabilistic schema matching, *Science Direct Information Processing and Management*, Vol. 43, 2007, pp. 552-576.

[15] K. Saleem, Z. Bellahsene, E. Hunt - PORSCHE: *Performance ORiented SCHEma mediation, Information Systems*, Vol. 33, 2008, pp. 637-657.

[16] A. Salguero, et. al, Ontology based framework for data integration, *WSEAS Transactions on Information Science and Applications*, Volume 5, Issue 6, 2008, Pp. 953-962.

[17] Yu J., Zhou G., SG: A structure based Web Services matching framework, *WSEAS Transactions on Information Science and Applications*, Vol. 4, Issue 4, 2007, Pp. 669-673.

[18] B. Villányi, P. Martinek - Analysing Schema Matching Solutions, *microCAD Conference*, 2009, pp. 127-132.

[19] B. Villányi, P. Martinek – Schema Matchers' Performance Improvement, *10th International Symposium of Hungarian Researchers*, 2009, pp. 613-624.

[20] Wu X., Feng J., A framework and implementation of information content reasoning in a database, *WSEAS Transactions on Information Science and Applications*, Vol. 6, Issue 4, 2009, pp. 579-588.