AUTOMATION OF SCENARIO-BASED SCHEMA MATCHER OPTIMIZATION

Balázs VILLÁNYI, Péter MARTINEK
Budapest University of Technology and Economics, Department of Electronics Technology, Egry József u. 18., V1. ép. fsz. 009, H-1111 Budapest, Hungary, Tel.: +36 1 463-2740, E-mail: {villanyi,martinek}@ett.bme.hu

ABSTRACT

Schema matchers are used to find related entities in schemas. Automated schema matchers are not infallible, consequently they need to improve on accuracy. Since the accuracy of schema matchers is scenario-dependent, our objective was to define universal methods with which the pre-run optimization of schema matchers for a given scenario is feasible. In this paper, we present our enhanced schema matcher optimization framework which allows the automated, scenario-based optimization of schema matchers. The output of this framework is the recombined schema matcher, which attained 33% average f-measure improvement over the input schema matchers. As part of the framework, we also devised a systematic comparison method for schema matcher components, the Comparative Component Analysis. We propose several performance evaluation bases for the ranking of schema matcher components.

Keywords: *schema matching, scenario-based optimization, accuracy improvement, schema matcher composition*

1. INTRODUCTION

Application integration, data integration, data warehousing and several other related application fields require that we find semantic correspondences in schemas. Schema matchers can be used to find these correspondences, i.e. to find schema entities which are semantically related [1, 2]. After the literature review, it has become evident that many of the schema matcher performance evaluation experiments seem to have been conducted under diverse special conditions. The consequence is that these schema matchers faced heterogeneous scenarios, impeding the adequate, unbiased, objective comparison of them. We can only draw a righteous and evenhanded conclusion, if we warrant that every candidate faces the exact same test conditions. That is to say they are given the same input schema, and they are all optimized for the specific scenario so that their maximal performance potential can be exploited. The result of the objective comparison is the ascending performance order of the input schema matchers, enabling us to rank these schema matching approaches.

There are several automated schema matchers. These matchers can be categorized. The linguistic matcher evaluates label similarities using syntactical methods. The auxiliary information or vocabular matcher uses some external dictionary, thesaurus, ontology, etc. to decide on the semantic relatedness of two terms. Schema graphs and entity relations are considered by the structural matcher, while instance-based matchers decide on the relatedness having schema instances as input. There are other schema matchers as described in [2], but hybrid matcher is the schema matching method which utilizes one or more schema matching approaches at the same time. In this paper, components of several schema matchers will be evaluated. The context-based matchers described by Boukottaya et al. [3] makes use of WordNet [4] dictionary. Another approach [5] uses related term sets and recursive structural matching. Lastly, Similarity Flooding [6] exploits Prefix/Suffix based linguistic matching and flooding in a combined schema graph.

Nearly in all of the related researches, methods have been analyzed as a "black box". Hence we were eager to gain insight into the working efficiency of the schema matchers with the intention of assessing the performance of their constituting elements. Treating the algorithms as black box did not fit our objectives. Hence we decomposed them into smaller parts, called schema matching components. Obtaining these components was only the first step. Several unanswered questions have emerged additionally. Among those, the alternative implementation possibility of the components was privileged, since this has not been investigated by others, to the best of our knowledge. We understand by this whether the usage of external thesauri is really necessary, for example. Namely, these latter methods have huge runtimes, so the possibility of their substitution with simpler syntactic based evaluators is desirable, while we could retain the original working mechanism. For details, see section 4.1.

Having performed some decision support based performance tests on the identified components, some new findings emerged. It turned out that certain types of components clearly outperform others. Among these, some components did not get the attention that they would deserve. Furthermore, we found that these performance rankings may only be valid for given scenarios. The performance ranking also defines the optimal schema matcher composition scheme for the given scenarios. In order to systematically exploit the potentials of this optimal composition and to formally define the steps required for this optimal composition, we have defined a schema matcher optimization framework.

The composition scheme can be used to define a scenario-based optimal schema matcher. Hence we decided to go further and define the concept of the recombined matcher, which should encompass so many advantages of the input methods as possible, while shaking off most of the drawbacks. This effort led to a comprehensive analysis of accessible schema matchers. In this paper, we present the results of this analysis, which also outlines a new algorithm comprising many benefits of existing solutions while omitting many drawbacks. In other words, it is our understanding that a substantial optimization possibility is provided by the unbiased, distortion-free, performance-based ranking among the input schema matchers by comparatively and

individually evaluating the performance of schema matcher components on a scenario basis. (See the Comparative Component Analysis in section 3.) Based on the comparative evaluation, a new, recombined schema matcher can be defined which incorporates the top ranking schema matching components of the input. At the same time, the recombined schema matcher is also the output of the proposed schema matching optimization framework, which contains every essential steps for the construction of the recombined schema matcher.

There are other works targeting the optimization of existing schema matchers. eTuner [7] also aims at combining schema matchers. Although it uses a different approach to ours: it takes the maximum, minimum, average of the similarity values produced by the schema matchers. We dissemble schema matchers into elementary units called schema matching components and define goodness measures based on which the component ranking is carried out, as described in section 3. The Schema Matcher Booster [8] approach defines two layers: the first is the layer of existing matchers, while the second layer enhances the matches produced by the first layer. Learning based methods are used in [9], whereby training of the learning method takes place in the offline phase, which is followed by the actual trained matching in the online phase.

All in all, our proposed framework enables the scenario-based optimization of schema matchers. The outputs are an optimal (recombined) schema matcher and a schema match given by the recombined schema matcher.

This paper is divided into sections as follows. This first section contains the preliminaries. The second section is dedicated to the schema matcher optimization framework, which is further divided into subsections: the first subsection formally presents the framework elements, while the second subsection is about the execution sequence optimization possibilities and the life cycle management. The method of the Comparative Component Analysis is presented in section three. Section four contains the experimental evaluation. The last section presents the conclusions.

2. THE SCHEMA MATCHER OPTIMIZATION FRAMEWORK

To provide a systematic optimization approach for schema matching, we define our enhanced schema matcher optimization framework (the preliminary ideas of which are presented in [11]). The full concept contains an extended number of elements and lifecycle management. This framework comprises abstract elements which may represent any arbitrary chosen adequate concrete technique. In this paper, we will focus on the parameter optimization and the comparative evaluation of schema matcher techniques. The input for the process are a set of schema matchers (which will serve as a basis for the optimal schema matcher construction), and a set of schemas (on which the schema matching itself takes place). By systematically executing the techniques covered by the framework, we gain an optimized schema matcher for a given scenario. Furthermore, the schema matching itself is also provided by this framework.

The framework with input and output is detailed on Fig. 1. Fig. 1 details the elements contained as well as their proposed execution sequence.

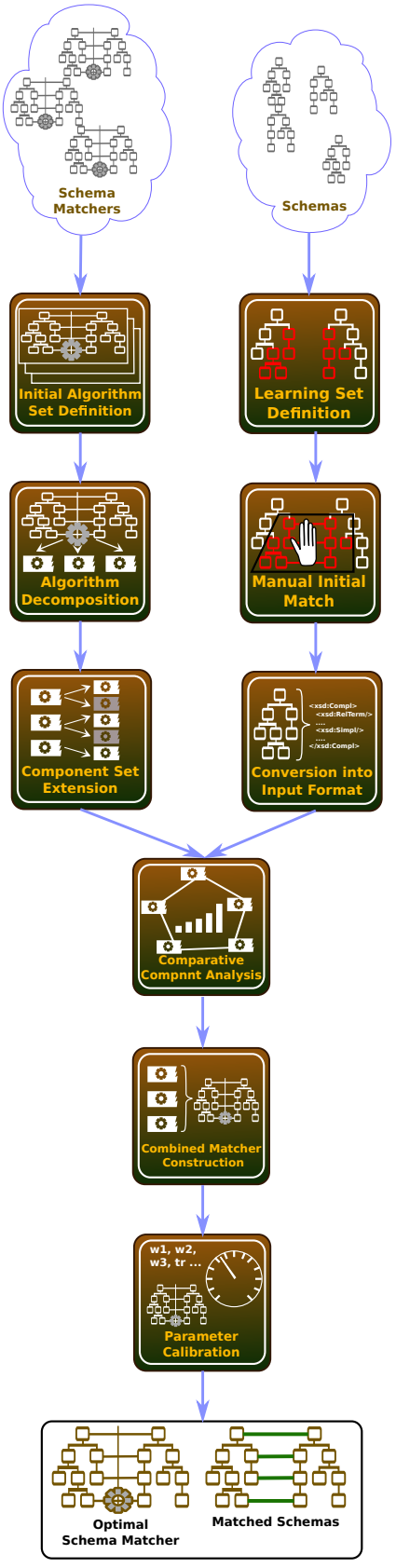

Fig. 1 The schema matching framework

2.1. The elements of the schema matching framework

This section describes the framework elements and the related tasks, while the next section details the proposed execution sequence focusing on the immanent, relevant parallelization possibilities. In this section, we will enlist and describe every element of the schema matching framework. We will also define the elements formally and introduce abstract functions for the associated tasks.

We will use the following notations. $\mathscr{L}_i$ is the learning set of schema $\mathbb{S}_i$, while $\mathbb{S}_i$ and $R_S()$ are the i^{th} schema and the schema subset relevancy function. We use $\mathscr{M}_\mathscr{I}$ for the initial matching (whereby the index "$\mathscr{I}$" refers to the initial manual match). The abstract function $\mathscr{M}(e_i,e_j)$ matches schema entities e_i and e_j. The abstract function $\mathscr{C}_I(\mathbb{S}_i)$ converts schema $\mathbb{S}_i$ of all input schemas in U^S (the schema universe) into the Input Format. S is the aggregated converted schema set, which will be used as input for the optimization. By the same token, A_i is the i^{th} element of all input schema matchers in U^A (schema matcher universe) and $\mathscr{F}(A_i)$ gives the general performance characteristic for schema matcher A_i. The abstract function $\mathscr{D}(A_i)$ decomposes input schema matcher A_i into components $C_i \ldots C_j$. C' is the component extension set, while C^p is the candidate extension component. The abstract function $\mathscr{C}_{\mathscr{S}\mathscr{M}}(C_i,C_j)$ comparatively evaluates components C_i and C_j, which produces performance value π_i. M is the recombined matcher assembled using the components with highest π_i ranking. Lastly, the abstract function $\mathscr{P}(M,S,\mathscr{M}_\mathscr{I},w,\tau)$ optimizes parameters (weight vector w and threshold τ) of schema matching components C on the aggregated converted schema set S.

The schema matching framework consists of the following elements:

1. **Learning Set Definition:** In this step, we select that part of the input schema set which should serve as the basis for the whole optimization process. Hence the set should be representative, yet minimal compared to the whole schema. The learning set will be supplied to the employed supervised learning techniques.

$$\mathscr{L}_i = \left\{ l \middle| \arg\max_S \mathscr{R}_\mathscr{S}(l), l \in \mathbb{S}_i \right\} \tag{1}$$

2. **Manual Initial Matching:** The human schema matching expert shall manually match the learning set to provide the ground truth match. The output will be supplied to the employed supervised learning techniques.

$$\mathscr{M}_\mathscr{I} = \sum_{e_i \in \mathscr{L}_m} \sum_{e_j \in \mathscr{L}_n} \mathscr{M}(e_i,e_j) \tag{2}$$

3. **Conversion into the Input Format:** The input schemas shall be converted into format which can be easily processed by the algorithms, contains all relevant information, yet it is devoid of redundant information. The aggregated output is the schema set supplied to the optimization processes.

$$S = \sum_{\mathbb{S}_i \in U^S} \mathscr{C}_I(\mathbb{S}_i) \tag{3}$$

4. **Initial Input Algorithm Set Definition:** This is the entry point for schema matchers. We should supply a set of schema matchers which cover all reliable schema matching approaches of interest and may serve as the component set in itself or after decomposition. (See next element.)

$$A = \left\{ A_i \middle| \max \mathscr{F}(A_i), A_i \in U^A \right\} \tag{4}$$

5. **Algorithm Decomposition:** The supplied schema matchers shall be dissembled into components to enable their analysis in details. The components will be the input for the Comparative Component Analysis.

$$C = \left\{ C_i | A_i \in A \Rightarrow C_i \ldots C_j = \mathscr{D}(A_i) \right\} \tag{5}$$

6. **Component Set Extension:** It is strongly recommended that the component set be extended with ones that are based on mechanisms of the existing components. This step is useful to have a wider spectrum of components and to investigate the cooperation of yet unpaired mechanisms (matching mechanism remix).(Optional element.)

$$C' = \{ C^p | \exists i : C^p \sim C_i \} \tag{6}$$

$$C \leftarrow C \cup C' \tag{7}$$

7. **Comparative Component Analysis:** An unbiased, distortion-free component performance evaluation shall be executed. This element is detailed in section 3, where we also provide some effective approaches to perform this task. The output of this step is a (accuracy based) ranking among components.

$$\pi_i = \sum_{j=1, j \neq i}^{n} \mathscr{C}_{\mathscr{S}\mathscr{M}}(C_i,C_j) \tag{8}$$

8. **Recombined Matcher Construction:** At this stage the recombined matcher is created using the most reliable components as given by the previous element. Further intuitive optimization possibilities could also be considered during the algorithm construction.

$$M = \left\{ C_i \middle| \max_{C_i}(\pi_i), i \leq 3 \right\} \tag{9}$$

9. **Parameter Calibration:** In this step we obtain the optimal parameter set for the new, recombined matcher with one of the techniques presented in [10]. This element is one of the – if not the – most important steps of the whole optimization process since the omission of calibration may lead to serious deterioration of the performance as detailed in [10].

$$w = \arg\max_{w,\tau} \mathscr{P}(M,S,\mathscr{M}_\mathscr{I},w,\tau) \tag{10}$$

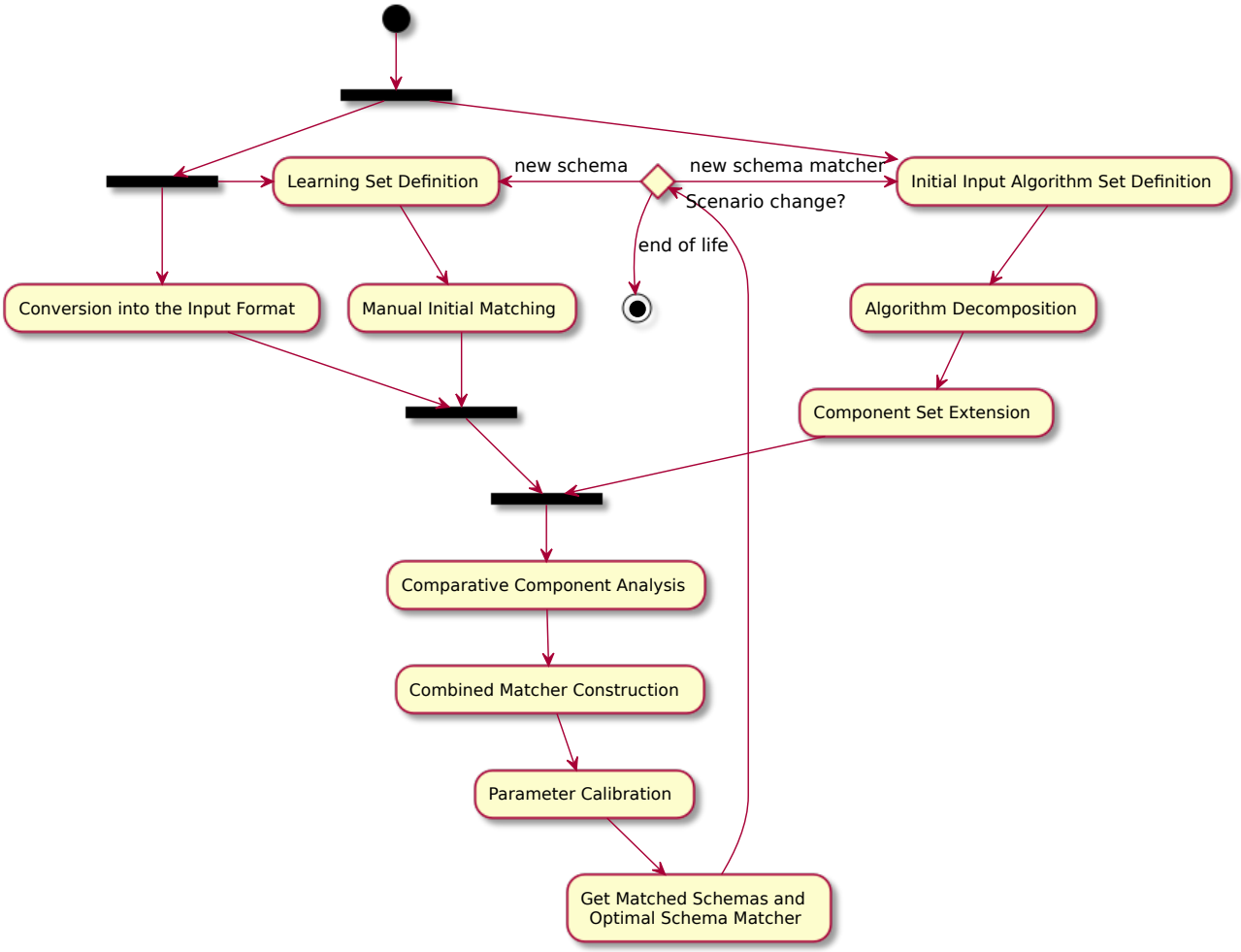

Fig. 2 Activity diagram of the schema matching framework

2.2. The execution sequence of the framework elements

In this paper, we also propose lifecycle management and execution sequence optimization for our framework. Although the tasks can be executed in the sequence that they are introduced in the preceding section, there are several optimization possibilities to cut down the execution time needed for the task. The execution sequence also enables the lifecycle management of schemas and schema matchers.

The startup sequence of the optimization process can be divided into two main sequences. The first sequence handles the tasks executed solely on the schemas, while the second includes those involved in the input algorithm decomposition. These sequences can be executed concurrently and have the schema set and the schema matcher set respectively as inputs. (See Fig. 2.)

The proposed execution sequence also enables the dynamic reconfiguration of the schema matching optimization environment. Should the set of used schema matching methods or that of the schemas change, not all elements have be re-executed. If a new schema matcher is added, only the algorithm decomposition steps – and those followed – are needed to be reiterated over. On the other hand, when the schema set changes – in so far as the schema matchers remain the same – only the learning set definition, the manual initial match, and the input schema conversion should be re-executed (along with the performance evaluation steps followed). For further information please refer to the diagram shown above, Fig. 2.

Considering the mentioned concurrency and the savings induced by the reduced number of elements to be re-executed if a given type of change takes place, it is obvious that substantial runtime can be spared. Our framework is designed so that it can be applied to a wide scale of schema matchers.

3. COMPARATIVE COMPONENT ANALYSIS

As stated earlier, the Comparative Component Analysis is a group of our proposed schema matcher optimization approaches. It provides an accuracy-based ranking among the input schema matchers on a scenario basis.

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. For

example, 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 them applicable even by very large component sets. The number and the place of occurrences of component nodes form the basis of ranking.

Another technique we often used at evaluating the component performance is the attribute weighting. 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, Information Gain and Principal Component Analysis (PCA) based ranking [12]. These methods can be used to calculate the relevance of attributes – i.e. the schema matching components in this case – with respect to the class labels [13, 14]. In other words, higher attribute weighting of a component means higher component relevance. This behavior makes these methods adequate to be a good ranking basis. They are calculated as follows. Following the notations given in [12] and [15], $X = \{x_0, \ldots, x_{n_c}\}$ is a discrete variable (the cumulated result matrix in this case), c is the number of classes, $P(x_i)$ denotes the probability of the event $P(X) = x_i$, $|X|$ is the size of X. $H(X)$ denotes the entropy and $Gini(X)$ (Eq. 12)denotes the Gini Index of variable X, while $\Delta_{info}(X,a)$ (Eq. 13) denotes the Information Gain if candidate attribute a of X is chosen. (The expression$\{x \in X | x_a = i\}$ refers to the entries, where a^{th} attribute of X equals to i.)

$$H(X) = -\sum_{i=0}^{c-1} P(x_i) \log_2 P(x_i) \tag{11}$$

$$Gini(X) = 1 - \sum_{i=0}^{c-1} [P(x_i)]^2 \tag{12}$$

$$\Delta_{info}(X,a) = H(X) - \sum_{i=0}^{c-1} \frac{|\{x \in X | x_a = i\}|}{|X|} H(\{x \in X | x_a = i\}) \tag{13}$$

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, but most of the time this was not true. In this latter scenario, further analysis can be conducted which targets the reason of this diversity. Based on the result, the decision which evaluator to choose can be made. Nevertheless, the results can also be aggregated to compare components using several evaluators at the same time.

Basically, all of the individual components are analyzed on the same schemas and a rank is compiled based on their achieved accuracy. Throughout our experiments, we preferred to use the decision tree building and the weight attributing based evaluations. However if other techniques are not applicable, then the MSE might also serve as a good basis for the component ranking.

The following Fig. 3 presents the Comparative Component Analysis.

Algorithm 1 The Comparative Component Analysis Algorithm

Description: *This algorithm ranks the schema matcher components using various predefined methods.*
Input: C – the cumulated result matrix
r – the reference match
μ – the evaluator method
Output: $\hat{o}$ – the component rank vector

```
1: procedure MaxPrec(C, r)
2:     for i ← 1, n do
3:         switch μ do        ▷ Rank every component C_i based on the chosen μ
4:             case 1
5:                 δ_i ←C45TreeRank(C_1, C_2, ..., C_n, r)
6:             case 2
7:                 δ_i ←GIRank(C_1, C_2, ..., C_n, r)
8:             case 3
9:                 δ_i ←IGRank(C_1, C_2, ..., C_n, r)
10:            case 4
11:                δ_i ←PCARank(C_1, C_2, ..., C_n, r)
12:            default
13:                δ_i ← Σ_{i=1}^{n_ep} ( r_i − Σ_{j=1}^{n_c} C_{i,j} w_j )^2
14:    end for
15:    ô ←Order components C_i by δ_i
16:    return ô
17: end procedure
```

Fig. 3 The Comparative Component Analysis Algorithm

As it is listed on Fig. 3, C4.5 tree ranking, Gini Index based, Information Gain based and Principal Component Analysis based ranking were used in our experiments. If no comparison basis is specified, then the mean squared error of the schema matcher from the reference may be used for evaluation purposes. Furthermore, other relevant comparison bases can also be used, as already stated.

4. EXPERIMENT RESULTS

In this section, we present the results attained with the proposed schema matcher optimization framework. The first subsection presents the results of the component performance assessment and the ranking obtained through Comparative Component Analysis, while second section presents the results attained with the recombined matcher. The third subsection is a brief overview on the performance improvement potential of optimized schema matchers after the application of the steps defined in the schema matching framework.

4.1. Results of the Comparative Component Analysis

We used four methods for ranking as proposed in Section 3: Gini Index, Information Gain, Principal Component Analysis (PCA) and Decision Tree based attribute weighting. This ranking is also the output of 7^{th} step of the schema matcher optimization framework. Many schema matching components have been inspected (for a complete list, see [11]), their aggregated number is 20. Numerous components have been evaluated for both entity names and types. In the experimental evaluation phase, it became clear

that this differentiation is not necessary, since both evaluation types lead to similar results, as it is demonstrated by Prefix/Suffix matching in Fig. 4. Secondly, some of the schema matching components consistently underperformed like Ancestor context. Hence in order to keep the presentation concise, yet demonstrative, we decided to offer a selection of the most characteristic results. After filtering out the components with comparatively poor performance, the following components shall be presented hereby out of those described in [11]: Ancestor context, Child context, Flooding, Leaf context, NTA attributes, NTA name, NTA related terms, WordNet-based evaluator, Prefix/Suffix matching. These components comprise both linguistic and structural schema matching techniques. We have conducted experiments on test schemas stemming from both own and literature sources. While using the scenarios defined in [3] and [5], we had test schemas defined using the OAGIS and xCBL standards. The obtained attribute weightings are listed on Fig. 4:

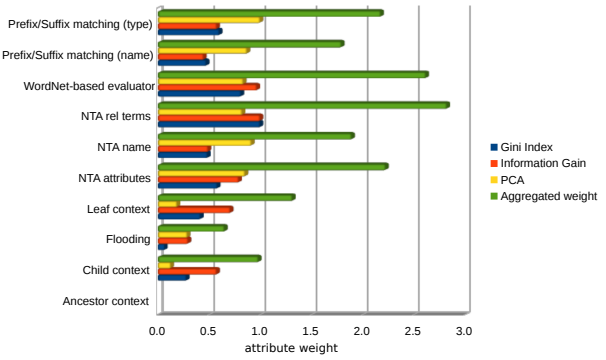

Fig. 4 The component weightings results for three test scenarios

On Fig. 4, you can see attribute weightings produced by Gini Index, Information Gain, Principal Component Analysis (PCA), as well as their aggregated weights, viz. the green bar sums the weights produced by the Gini index, the Information Gain and the PCA based attribute weightings. As already stated, we made the distinction between the same schema matching being applied to the entity names (denomination) and type only in the case of Prefix/Suffix matching to demonstrate what typical difference was observed.

As it can be seen on Fig. 4, the group of top-echelon components incorporates the NTA attributes, Prefix/Suffix matching (type), WordNet-based evaluator (specifically the WordNet based sentence matcher for types) and the NTA related terms. The group of second rank components includes the NTA name, the Leaf context and the Prefix/Suffix matching (name).

Examining the ranking by attribute weighting methods, we can observe that the Gini index and the Information Gain based attribute weightings delivered very similar results. By both techniques, the NTA related terms and the WordNet-based evaluator were ranked the highest. These components were also ranked high by the PCA-based attribute weighting. Also, the other end of the rankings are the same: Ancestor context was ranked lowest. Prefix/Suffix matching (type) and NTA attributes were ranked below the WordNet-based evaluator (obtaining similar weights) in the case of the Gini index attribute weighting, while the Information Gain based attribute weighting ranked the same the NTA attributes and the Leaf context components. (There is only a minor difference in the weighting of these components, so they are considered to be ranked the same.) The rank of these components is somewhat different in the case of the PCA-based attribute weighting, but basically the same components were ranked high except for one: the NTA name. This latter result is not in concordance with the ranking of the other two and we would also have expected lower ranks also in the case of the PCA-based attribute weighting.

We also built a RapidMiner C4.5 like decision tree [13] 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, see Fig. 5.

WordNet-based evaluator > *0.945*
WordNet-based evaluator *0.945*
| **NTA rel terms** > *0.178*
| | **NTA attributes** > *0.297*
| | **NTA attributes** *0.297*
| | | **Flooding** > *0.189*
| | | **Flooding** *0.189*
| | | | **Child context** > *0.303*
| | | | | **Ancestor context** > *0.186*
| | | | | **Ancestor context** *0.186*
| | | | **Child context** *0.303*
| **NTA rel terms** *0.178*

Fig. 5 Decision tree of the component result vectors

This experiment also showed us that related term similarity is a top-level approach, which fact further accentuated the necessity for good quality related term sets. It has also become clear that the WordNet dictionary based similarity components are excellent evaluators, but we should choose the sentence matcher for names from its variations, since the other components had no place in the pruned tree. You will also find the NTA attribute and similarity flooding nodes in the tree, however, not in the closest vicinity of the root. The structural matcher which was ranked highest in the previous experiment – NTA attributes – can also be found in the tree. In fact, if we compare Fig. 4 and Fig. 5, we can agree that the group of top-echelon components is comprised of the NTA attributes, WordNet-based evaluator and the NTA related terms.

In every case, some components were found to be consistently underperforming. Among these the exclusively Prefix based linguistic matching can be mentioned. The WordNet based techniques also underachieve if they are not used as sentence matcher. The underlying reason can be the dense multi-word denomination of the entities, in which case these labels are not to be found in the dictionary.

Based on what we presented earlier in this paper, the following conclusions can be formed about the components of the input schema matchers:

- The related terms matcher performed better than the WordNet-based matcher, hence the related term set can be used instead of external dictionaries.

- There was no significant difference between Prefix/Suffix matching (type) and Prefix/Suffix matching (name), but the former approach slightly superseded the latter.
- Among the context matchers, the Leaf context matcher is the most accurate, while Ancestor context was ranked lowest (by all of the evaluator methods).
- NTA attributes outperformed the context matchers, but it was ranked lower than both the NTA related terms and WordNet-based matcher.

4.2. Recombined matcher

Having performed all the necessary comparisons and the subsequent evaluation of the results, all the necessary prerequisites are met to define a new schema matcher which has foreseeably more potential than its donor matchers. The recombined matcher is also the output of the schema matcher optimization framework.

However, the task does not only consist of the selection of components, but also of the proper parameter setting. For this task the techniques presented in [10] can be utilized. The recombined matcher is constructed according to the conclusions presented in the previous section. Some implementation level optimization was also applied so that the algorithm consume only the runtime absolutely necessary. To sum up, the optimization steps defined in section 2.1 were systematically executed to gain the recombined schema matcher of the input schema matchers.

The linguistic matcher was Prefix/Suffix based matcher because of its outstanding performance among linguistic matchers. As next, there are two related aspects to be considered: the distinguished role of the related terms matcher and the fact that this set is not always provided. Consequently, we implemented an automatic choice between these two available methods. The recombined matcher examines whether the related terms set is available. If the answer is positive, so the procedure uses the related term comparison; if negative, then the procedure initiates vocabulary query. 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. This solution seemed to be reasonable according to the conclusions of the component comparison. All in all, the approach involves linguistic, vocabular and structural matchers, and the parameters were optimized using f-measure maximization method.

Several experiments have been conducted. Our goal was to obtain the highest f-measure values possible. The Table 1 below summarizes the averages and the standard deviations of the attained maximal f-measures in the test schemas:

Table 1 The average f-measure and standard deviation of the composed matcher

	RCM	NTA	SF	WN
Average	0.97	0.91	0.58	0.84
St. deviation	0.05	0.08	0.12	0.16

The table uses the following abbreviations. RCM denotes the recombined matcher, while SF marks the similarity flooding [6] and WN is the WordNet-based matcher [3] and NTA is the schema matcher in [5]. The table shows us that the new matcher, which consists of the selected components of the others, performs better than the originals. This result is provided as a remedy for the given scenario where originally outstanding accuracy had not been achieved by any of the input methods. On other schemas these values may differ somewhat, but in those scenarios the construction of the recombined matcher should be reiterated.

4.3. Performance improvement potential of optimized schema matchers

We have conducted several analyses on various types of schemas and it has turned out that substantial accuracy improvement can be obtained using the optimization methods described hereby: c. 33% f-measure improvement was obtained in average compared to the initial, unoptimized weight setting, and this improvement is not even measured to the worst case. By random weight setting, we can expect the f-measure to be 0.45-0.5. The maximal observed improvement was 0.46 as expressed with f-measure, which corresponds to a c. 100% accuracy improvement regarding the 0.45 f-measure at the first attempt. We have also obtained convincing results regarding the performance of the algorithms that had shown mediocre results earlier. The framework also provided means by which we could automatically acquire those calibrated recombined schema matchers, which we had had to construct with laborious manual effort earlier.

5. CONCLUSIONS AND FUTURE WORKS

In this paper, a schema matcher optimization framework has been described, which can be applied to a large scale of schema matchers and schema matching scenarios. Although we have offered our own optimization solutions to the framework, any substitute method can be employed as long as it efficiently carries out the parameter optimization and comparison tasks. The innovative aspect of the framework is its capability to provide optimal schema matcher in arbitrary chosen semantic integration scenario and thus it allows the obtainment of ideal matches. Automation, concurrence and lifecycle management have been especially considered during the design process. The design contains many easily extensible interfaces, which offer dynamic extension points in the future. Moreover, several experiments have been conducted on the implementation and we have witnessed some fairly convincing performance boosts.

Also, we have enumerated several techniques to obtain fair ranking among schema matching components and a list of the most reliable schema matching components has been compiled. We refer to this group of techniques as Comparative Component Analysis. We have analyzed the weak points and the merits of some solutions. In fact it did turn out that they include some remarkably useful components. These components then can be used to make ground for a

new solution which could supersede current solutions both in accuracy and efficiency. With the help of the Comparative Component Analysis, a new recombined schema matcher can be built with enhanced performance characteristics in any schema matching scenario.

Other schema matchers will also be tested with the methods introduced in this paper. Based on what we believe, a new testing benchmark could be set up, which can be used to analyze schema matchers and makes their fair comparison possible. The most important issue is that the tested algorithms should face exactly the same challenge and their accuracy should be measured with the same methods. This is exactly what we have done during our research and we are making efforts to define more testing procedures.

We will also investigate the usage of continuous learning, whereby the schema matching framework tasks should be reiterated over as the schema matching scenario changes. The main challenge is to appropriately and efficiently modify the framework state as the externally-triggered change event takes place. We should find the minimal framework parameter set change required for each and every identified schema matching scenario change type.

REFERENCES

[1] RAHM, E. – BERNSTEIN, P. A.: A survey of approaches to automatic schema matching, *The VLDB Journal*, vol. 10, no. 4, pp. 334-350, 2001.

[2] BERNSTEIN, P.A. – MADHAVAN, J. – RAHM, E.: Generic Schema Matching, Ten Years Later, *Proceedings of the VLDB Endowment*, vol. 4, no. 11, pp. 695-701, 2011.

[3] BOUKOTTAYA, A. – VANOIRBEEK, C.: Schema matching for transforming structured documents, *ACM Symposium on Document Engineering*, pp. 101-110, 2005.

[4] MILLER, G. A.: WordNet: a lexical database for English, *Communications of the ACM*, 38(11), pp. 39-41, 1995.

[5] MARTINEK, P. – SZIKORA, B.: Detecting semantically related concepts in a SOA integration scenario. *Periodica Polytechnica Electrical Engineering*, vol. 52, no. 1-2, pp. 117-125, 2009.

[6] MELNIK, S. – GARCIA-MOLINA, H. – RAHM, E.: Similarity flooding: A versatile graph matching algorithm and its application to schema matching, *Proceedings. 18th International Conference on Data Engineering*, pp. 117-128, 2002.

[7] LEE, Y. – SAYYADIAN, M. – DOAN, A. – ROSENTHAL, A. S.: eTuner: tuning schema matching software using synthetic scenarios, *The VLDB Journal*, vol. 16, no. 1, pp. 97-122, 2007.

[8] GAL, A. – SAGI, T.: Tuning the ensemble selection process of schema matchers, *Information Systems*, vol. 35, no. 8, pp. 845-859, 2010.

[9] JEONG, B. – LEE, D. – CHO, H. – LEE, J.: A novel method for measuring semantic similarity for XML schema matching, *Expert Systems with Applications*, vol. 34, no. 3, pp. 1651-1658, 2008.

[10] VILLANYI, B. – MARTINEK, P. – SZIKORA, B.: Calibration and comparison of schema matchers, *WSEAS Transactions on Mathematics*, vol. 8, no. 9, pp. 489-499, 2009.

[11] VILLANYI, B. – MARTINEK, P. – SZIKORA, B.: A framework for schema matcher composition, *WSEAS transactions on computers*, vol. 9, no. 10, pp. 1235-1244, 2010.

[12] TAN, P.-N. – STEINBACH, M. – KUMAR, V.: *Introduction to Data Mining*, Addison Wesley, 2006.

[13] AKTHAR, F. – HAHNE, C.: *Rapid Miner 5 Operator Reference*, Rapid-I GmbH, 2012.

[14] HOFMANN, M. – KLINKENBERG, R.: *RapidMiner: Data mining use cases and business analytics applications,* CRC Press, 2013.

[15] RUSSELL, S. – NORVIG, P.: *Artificial Intelligence: A Modern Approach*, second ed., Prentice-Hall, 2003.

[16] ROWELL, M.: OAGIS: a canonical business language, *XML Journal*, vol. 3, no. 09, 2002.

[17] ONE, COMMERCE. INC.: *xCBL Specification*, 2001. `http://www.xcbl.org/`

Received March 17, 2016, accepted May 23, 2016

BIOGRAPHIES

Balázs Villányi was born on 13. 02. 1986. In 2011 he graduated (MSc) with distinction at the Department of Electronics Technology of the Faculty of Electrical Engineering and Informatics at the Budapest University of Technology and Economics. He is currently a PhD candidate in the field of enterprise application integration. He investigates questions related to schema matching.

Péter Martinek was born on 17. 10. 1980. In 2010 he received his PhD degree in the field of enterprise application integration. He is currently an associate professor at the Department of Electronics Technology of the Faculty of Electrical Engineering and Informatics at the Budapest University of Technology and Economics. He investigates production scheduling, enterprise cloud systems and application integration.