Calibration and comparison of schema matchers

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

Abstract: - Schemas used in various environments become more and more numerous, though they do not comply to a universal standard. That is why the task of schema matching has emerged and its main objective is to find means to map a schema into another. Several initiations have occurred and algorithms have been proposed to solve the problem. They muster highly enticing solutions, though they have several flaws. We have reviewed the available algorithms and implemented some of them. We found that the accuracy of these solutions is strongly dependent on some well definable matcher characteristics, so if we calibrate the matchers they perform a lot better. Taking into account this fact we cannot compare matchers until after the necessary calibration. We propose matcher independent procedures and mathematical formulas to perform the highly desirable pre-run configuration of candidate solutions. The necessity of calibration implies that the unbiased comparison of solutions is not possible until the configuration is performed. We also introduce the technique of multiple thresholding which promises a better view of the result list returned by the individual matchers.

Key-Words: - Optimization of schema matching algorithms**,** Determine possible accuracy maxima, Similarity measure methods

1 Introduction

Due to the lack of comprehensive overall standardization between data schemas, solutions dealing with the integration of diverse data structure have become an important issue in business informatics. For this task, the identification of semantically related concepts in schemas is fundamental. This process however encompasses a lot of consideration, e.g. assessing schema similarity through diverse schema properties. Furthermore, current schema matching approaches simply do not offer a universal solution. The best possible result could only be acquired through the proper understanding of the scenario, the schema, the context, the task and maybe several miscellaneous user needs. The problem of schema mapping is referred to as schema matching, and several solution candidates have been aired, enumerating some cardinal features.

Schema matching is identified as the task, which takes two schemas as input and returns the similarity between them, i.e. the correspondence between their concepts. As the number of schema providers grows, the task becomes reasonably even more entangled.

Regarding the current stance, different vendors do not conform to world wide open standards. The task must remain a key issue even after the not yet prognosticated standards, as different vendors may remain reluctant to adhere to them. Not only the schemas increase in number, but also the application scopes. The experts of the fields of data ware housing and information retrieval have realized the possible benefits of an efficient solution long ago, but also so have done their colleagues occupied with development of web shops, price watching sites, and other web related application operating with multiple data sources.

Not always the integration task stays in the focus of business interest; sometimes the mediation between multiple sources is the desirable goal. Easy to see that two tasks are settled on the same basics, thus they are related. It also entails the figuring of corresponding concepts, though it may not involve the schema integration. Some promising data mining application fields may have similar requirements and leave out the joint schema construction after a successful extraction of the information needed. This multiplicity in both the schemas and the application scopes implicate that the solutions should return trustworthy results under highly varying conditions and do this possibly without much fluctuation in their accuracy. The procedure complicates when deceptive, hard to distinguish concept pairs appear in the input. Consider for example when two strongly related concepts are involved with totally different representation, e.g. labeled according to different naming convention and they also show structural heterogeneity. On the other hand, when two entirely different concepts are confronted in similar context, the solution should distinguish between them. After some brief ponder, we can conclude that the solutions should be able to outmaneuver some delusory falls often encountered

when comparing schemas. The prerequisite to perform under uneven conditions implies the need to find generic solutions, which are scalable, easily maintainable, widely deployable, universal and comply to given schema description standards.

The schema is a specification of data, a series of rules how the information should be stored. It is by itself hard grip, but its representational forms render it quite handy. In most of the times it is regarded as a tree structures enriched with additional meta-information. This is the form one most likely to associate the information contained in the schema with. However there are other forms that possess equal ability to express schema information, for example a series of SQL statements. The difference lies not only on the representation plane, but also on the conceptualization one. Note that when we mention nodes of tree structures when referring to concepts, we use tables, columns and column types when referring to SQL terminology. It is of the most evident nature that these terminologies should not be treated as being completely different when discussing schema matching, rather as two different conceptual approaching to describe the very same task input. This means of course these representations remain interchangeable and passage between them should be most of all without penalty. Another common representation form is the XSD schema, which is a de facto schema description standard. For the computational processing this representation may be the most lucrative, as by standard this encompasses the biggest number of meta-information types. Our choice also fell on this particular form though we found that some features of the XSD schemas are not advantageous, so we endeavored to rule them out one by one retaining only the utilized information. This had a remarkable effect on the runtime costs and turned the original schema description into an easier to comprehend representation. The task remains then to find a correct mapping between the concepts enlisted on both sites.

We distinguish two main archetypes in schema matching. Most common is the schema-level matcher which is eligible to work on every piece of information except those classified to instances. This means that these matcher use concept names, types and structural information primarily. They also typically capitalize on attributes, paraphrasing terms and idioms and roughly all the rest that can be derived from the schema description. In order to categorize them, three prevailing types are mentioned here. Linguistic matcher compares names, descriptions, labels and other annotations. The main goal here is to obtain similarity values among the schema nodes. Structural matcher bears the same importance, as it enables to gain insight in the structure of the schema. Constraint matcher incorporates nearly every other type of schema data which are not belonging to the managed data of the latter two. Specifically it may make use of cardinality restrictions, types, ranges and uniqueness. In the vast majority of cases an effective linguistic matcher (also exploiting type information) is combined with a structural matcher and maybe an optional constraint matcher is recommended. The actual combination depends on the field of deployment, the input information provided and the desired accuracy. Combined matchers are often referred to as hybrid matchers. The other main archetype is the instance-level matcher, which – adhering to its name – compares using instance data. The similarity value this time depends on the actual content and not the generic schema information. It plays a role not to underestimate when ambiguous information is presented. Schema-level matchers may not decide for the right in the case of a not straightforward, decisive pair mapping. Nevertheless schema-level matchers remain the prevailing in the integration task. Mentions-worth is the fact that combined schema-level and instance-level matchers may also be referred to as hybrid matcher.

Schema matchers do not always work on their own. They are sometimes extended with auxiliary units or combined with each other. Most notably schemas may use ontology, thesauri, dictionary in order to bolster the linguistic matching and draw better conclusions based on the semantic content of the schema. In this case the particular ontology to be used and the way the relatedness of two concepts interpreted into values are subjected to further considerations. The potential in composite matchers – that is an optimal combination of certain matchers – is already revealed and promisingly exploited. On the other hand we should not forget about the significant overhead. We also found that the introduction of linguistic decision supporting units (dictionaries, ontology) also results in an overhead, and if not scaled properly, the usage of them may not be remunerative.

There is a very promising number of algorithms concerning the schema matching. However most of the presented solutions are not reliable in the meaning that their output should be supervised before they could be deployed, so the human intervention cannot be set aside. This is not only inconvenient but manifests in a substantial overhead. They evaluation process should be improved, so as to crack down the superfluous time expense. The available algorithms incorporate a lot of parameters and occasionally fine interpretative distinctions. We were aimed at the identification of these and harnessing them so that they produce better results without the prior human schema analysis and solver fine-tuning.

Every known experiment that should evaluate the performance of the schema matcher seems to be conducted under artificial conditions, distorting the

conclusion. We can only draw a righteous and evenhanded conclusion, if we warrant that every candidate face the very 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 we acquire their best possible result. This probe defines the correct order of algorithm accuracy performance, enabling us ranking the existing solutions.

The paper presents the computational methods that enable the scenario based optimization of schema matching algorithms. The paper is structured as follows: the second chapter lists some related works. The detailed description of the accuracy evaluation of schema matcher methods is presented in chapter three, while chapter four presents the algorithms. Chapter five encompasses the accuracy enhancing methods. Our experimental results are detailed in chapter six, chapter seven contains some thoughts of consideration. The last chapter is about our conclusion.

2 Related Work

Numerous researches concerning the schema matching are available [3, 4, 5, 7, 8, 9, 18, 19, 20, 21, 22]. During our experiment we analyzed these approaches and implemented the approaches presented in [1, 11, 12]. These algorithms are detailed in chapter four.

The solution in [7] is a generic schema matching tool, called COMA+. The peculiarity of this proposal is that it is not a schema matcher by itself, but constitutes a sophisticated platform in which several others can be integrated. Authors believe that with this platform the combined advantage of schema matcher can be exploited. The individual matchers are arranged in to a library. The solution provides scalability by fragment schemas into subsets, trying to capitalize on the divide and conquer principle. This approach also contributes to the flexibility of the platform. On the other hand, it is not fair when comparing algorithms, because it does not include parameter optimization.

An automated schema matching solution working on XML schemas is presented in [20]. The described method combines linguistic and structural similarity extended with the evaluation of data type compatibilities. Within the linguistic part abbreviations and acronyms are also identified while prepositions and articles are disregarded due to domain-specific dictionaries constructed for the examined schemas.

Similarly, the authors in [1] also present a combined approach. The evaluation starts with a linguistic analysis based on the open dictionary called WordNet[6], but the main added value is the comprehensive structural method performed on the schema trees. The structural similarities are examined in three contexts: the children, leaves and root environment are compared to all possible pairs of input schemas. An implementation of this approach is also judged in details in our work from the point of view of accuracy. See later in experiments.

Another promising approach to the schema matching problem is presented in [10]. In this paper the algorithm called Cupid is described. It has a complex evaluator incorporating a composite structural matching and a linguistic matcher. The latter one provides initial value based on string-based node comparison. According to the comparative study presented in the paper, the Cupid outperforms the DIKE[14] and the MOMIS-ARTEMIS[2]. This comparison however lists only capabilities and does not tell us about their accuracy performance. An actual result of a test measuring accuracy on test schemas is obviously missing. The capability comparison does not provide a clear view of the goodness of the individual solutions.

In [13] the authors present a schema matching method working on XML schemas. Similarly to the approach Cupid the evaluation starts with the clustering of schemas into various groups. The syntactic similarity measurement is performed in 3 steps namely preprocessing, data mining and postprocessing while a specific graph representation called dendogram facilitates the generalization and specialization processes of the clusters to develop an appropriate schema class hierarchy. Unfortunately the analysis of the results is restricted to the parameterization of the presented approach and is only presented in the unique metrics of the paper. This hinders the comparison with other approaches. However taking into account the size of evaluated schemas and the values of applied efficiency indicators the performance should be at the same level as the methods in [1] and [10].

We have observed flaws also by the comparison method presented in [11]. Although the evaluation in the paper introduces the Precision, Recall and F-measure, the accuracy result is given based on a single test schema. The question whether other candidate solutions were optimized for test scenario remains open. They also fail to mention the particular output of the matcher may not be compatible with that of other approaches. They distinguish two type of nodes based on their path, but some other approaches only distinguish types. How this controversy should be resolved remains the subject of personal judgment. The problem how linguistic similarity values are elicited from the WordNet [6] is not detailed enough, implementations may provide various similarity values which have serious impact on the result.

3 Evaluating Accuracy

Having results at disposal is in itself not expressive when goodness and quality comes in to question. In order to decide on accuracy, the results have to be in a

compatible, comparable form, what will be covered in this section.

The semantic distances – which are similarity characteristic values ranging from 0 to 1 – does not tell us whether they are meant to be matches or not. So the problem expands, encompassing the need of determining a limit called threshold that cuts the result set into two halves. Obvious that the adequate calculation of this value is at least as important as the proper working parameter set of the solutions.

The results are stored in matrices, so that the values are more expressive to the human supervisor. Only the compatible format should be warranted, which entails some consideration, among others the format difference [1] and [11] must be untangled. This format enables to discuss the result accuracy relatively to each other effectively. After injecting the threshold into the similarity matrix, it includes only 0 and 1 which makes it easy to compare with the reference solution.

Other important requirement we should fulfill is the definition of the reference solution. It is not so straightforward because of the aforementioned reasons. To be unbiased we decided to make a survey involving some twenty human evaluators, whose task was to solve problem under fairly similar conditions as it would be in the real life. The willing volunteers submitted their solutions which then were summarized after the necessary filtering. As the evaluators varied on a large scale, it has turned out that some of them clearly lacked the professional skills to give a perfect match. This is the reason why the necessity of the filtering ensued. This inaccuracy is not uncommon among human users, the survey clearly shores up our assumption on the complexity of the task.

Having the algorithm and reference results available, every condition needed to evaluate the efficiency is met. To calculate performance we need accuracy measures extracted from the result matrices. We used the most prevailing measures: the Precision, the Recall and the F-measure. They are best known for their usage in information retrieval, but they are not unheard of in other scopes of computational accuracy measurement. After making a brief survey, we found that these measures are utilized to describe the goodness of the individual solutions by the overwhelming majority of schema matching performance analysis. The measures are calculated with the formulas as follows: (1) Precision Formula, (2) Recall Formula and (3) F-measure formula.

$$\mathrm{Precision} = \frac{\left|\{\mathrm{proposed\_matches}\} \cap \{\mathrm{relevant\_matches}\}\right|}{|\{\mathrm{proposed\_matches}\}|} \quad (1)$$

$$\mathrm{Recall} = \frac{\left|\{\mathrm{proposed\_matches}\} \cap \{\mathrm{relevant\_matches}\}\right|}{|\{\mathrm{relevant\_matches}\}|} \quad (2)$$

$$\mathrm{F\_measure} = 2 * \frac{\mathrm{Precision} * \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} \quad (3)$$

Accuracy is then defined as the value of either the Precision or the Recall or the F-measure. Our main objective was to define methods which maximize these measures for a given scenario and algorithm. Note that this is not always possible as [12] does not define intermediate similarities which then could be summarized by a weighted sum. In this case other methods are necessary which include the possible partial revision of the algorithm in question. In this article, we focus on the weight-based calibration; on the other hand we should note that we came up with amending suggestions to [12].

The main goal is to maximize a measure for a given algorithm and scenario. To achieve this, we should seek the optimal weights and threshold. We call this task calibration or parametrization, referring to possible analogy to other scopes of science. For the problem to be manageable we should find correspondence among the algorithm properties. The most important of them is given by the definition of the measures. We are able to calculate them for every parametrization, thus when observing the behavior of these measures during the fine-tuning of the factors we witness the correspondence between them. More specific is the characteristic that weights complement each other to one. It implies that we should care about one less weight, as the last one can be expressed as the function of the others. In our case it means that we should tackle the problem for two instead of three weights. Other relieving fact is that the threshold can be expressed as the lowest matching or the highest non-matching value (more accurately a little higher than the exact value in the latter case). This proves to be useful when the involvement of the threshold calculation substantially complicates the problem.

4 Algorithms

We selected the most promising solutions available. This entailed a comprehensive analysis of published schema matchers. The main issue was to embrace a large scale of matching aspect. One of our main contributions is the detailed examination and ranking of these schema matchers.

Our choice fell on three algorithms. The first one is a complex matcher [11] encompassing the Name, related Terms and Attributes (called NTA in the rest of the paper) similarity assessment, eventually the semantic distance delivered by the weighed sum. Peculiarity of this solution is that it is relatively simple, though effective and also involves recursive elements. The evaluation is based on scores given to string comparison based similarity (0, 0.5, 1 respectively), which is then

combined in to set similarity values with methods described by equations. The second algorithm is the Similarity Flooding (called SF in the rest of the paper) [12]. This solution presumes that the more similar neighbor nodes of two concepts are the more similar the concepts themselves are. This presumption is implemented by defining an extended similarity propagation net, in which iterative flooding of similarity values along weighted edges is executed. The iterative flooding is delimited by a stop condition. Another candidate proposing a trustworthy solution is the WordNet-based Matcher (called WN in the rest of the paper)[1]. The hallmark of this solution is the capitalization of the complex conceptual synonym vocabulary called WordNet [6]. The usage of which forebodes the best possible linguistic similarity, for it not only exploits the string-based similarity but also the meaning based one. It also defines a complex structural matching, trying to take into account the relative and absolute position of the concepts in the structure. It designates three contexts in which then mingled linguistic and complex path similarity is performed. Where applicable also set similarity values are expressed, counting the best forthcoming values. The three context similarities is then combined in to the semantic distance by building the weighted some of them.

5 Accuracy measurement methods

In this section we will describe methods, which we propose to the problem. In the followings we assume that all necessary prerequisites are met (result matrices elicited from the semantic distances and a reference table). When describing the solutions we will do with notations used for the NTA. This does not make a difference as the solutions are applicable unaltered to the WN. The N, T, A matrices contain the similarity values of the name, related term and attribute similarities. The R, P, I matrices contain the reference values, the matches retrieved by the algorithm and the retrieved relevant matches respectively. The w_1, w_2, w_3 and τ values denote the weights and the threshold. The S matrix, s(m) and s(n)T vectors denote the matrix and the vectors consisting of value one. The sum of the elements of matrix X is marked with ||X||.

5.1 Accuracy maximization with reference-approximation

This first method is mention-worth for its simplicity beyond its ability to work properly. The result is given by formulas, easily calculable even with simple computational tools, e.g. calculator. Consequently, it is effective and can be determined instantly even for larger schemas. It returns a single value, other possible results are not listed. This however does not diminish the solution, as in most cases a single good result is satisfactory.

It uses an indirect approach. The key idea is that the F-measure value can be maximized by seeking the weight distribution where the ensuing result matrix nearest approximates the reference table. This approximation is understood as the aggregation of the element differences between the two matrices. In other words, we should build the average of the quadratic deviation of every element, and minimize by means of mathematical analysis. This approach has the benefit of resulting in exact formulas, which have coefficients ready to be substituted for values. *It is not hard to see, that the method guarantees the extrema is not achieved through a few low deviation values, but all involved*. The threshold is not included in the calculation, it can be obtained as the minimum of the matching values.

As described we suggest approximating the reference solution by modifying the weights of the partial similarity matrices. In general this method is not restricted exclusively to the optimization scenario where exactly three matrices are involved. The principle remains the same, only the number of variants varies in the single cases. Note that there could be contradictions and discrepancies that may not be resolved by the weight manipulation. An indication pointing towards the possibility of conflicts is that the weights assume highly unlikely values. It also means that maximal accuracy – e.g. maximum f-measure value – cannot be achieved given this particular set of data. The problem is easily solved after some manual fine-tuning. In real life deployment it means, that instead of a single instant calculation step, a posterior supervision is required. The filtering of discrepancies is done by observing values that contradict the trend represented by other concept pairs. Consider for example that by a particular set of weights, every semantic distance approximates the value given in the reference table, except for one or two that approximate quite the other end of the semantic scale – zero or one. If we correct these pairs, by changing either the result values (requires more consideration) or the reference values, we conclude with a much better accuracy than the originally possible. If changing of some values not allowed for some reason, we can stay by the not so accurate approximation. Note that in this case the correct approximations – that is result values approximating the reference values – may not be as accurate as after the filtering.

Performing the filtering the supervisor should pay attention to the original semantic set. She should seek to rule out only the discrepancy peeks, and avoid upsetting the whole value set. By proceeding heeding this remark,

a much better accuracy can be achieved as when simply performing exclusively the accuracy maximization with measure maximization.

The base task can be formulated as follows. Expression (4) shows the base task of the reference-approximation and expressions (5-6) show the final formulas of the reference-approximation and accuracy maximization with measure maximization respectively.

$$\min(w_1 N + w_2 T + w_3 A - R) \tag{4}$$

$$w_1 = \sum_{i=1}^{n}\sum_{j=1}^{m} \frac{r_{ij} - a_{ij}}{t_{ij} - 2a_{ij} + n_{ij}} \tag{5}$$

$$w_2 = \sum_{i=1}^{n}\sum_{j=1}^{m} \frac{(r_{ij} - a_{ij})}{(t_{ij} - a_{ij})}\left(1 - \frac{1}{t_{ij} - 2a_{ij} + n_{ij}}\right) \tag{6}$$

5.2 Accuracy maximization with a given measure maximization

This method is an approach to manipulate and eventually maximize the measure directly with the adjustment of the weights. First of all let us discuss the F-measure maximization, but you will also find the precision and recall maximization in this chapter. This task however is not so straightforward as the former one. The task entangles because of the conversion of semantic distances to result values. Clear that this task involves a not continuous function as decision about a value being match or not is involved. This prevents the usage of the means used in the case of the reference-approximation, yet with computational means approximation of the output list possible. This results in a much more complex approach than the reference-approximation, though it returns all possible solutions. The goal is then to derive the formula of the F-measure so that it has only the weights and threshold as unknowns.

The choice on which particular measure perform the optimization task should be done after taking into account the needs. One could argue that the f-measure is the most complex measure and in addition it also involves the other two. Unfortunately the decision is far more complex than just savoring the justice of this statement. One thing is for sure. F-measure delivers an overall accuracy. Nonetheless not always all aspects of the accuracy are equally important. This is the main reason behind introducing a whole variety of accuracy measures.

Sometimes the precision is preliminary measure. In this case not the comprehensive integration is the priority, but the quality of it. This is so because in a fully automated environment a single wrong match could cause much more harm than left out cohesive pairs. The main issue is to find related pairs, to find most of them. No injuries sustained of not all of the unimportant cohesive pairs are matched. This could prevail when working with global and service schemas. Not the exact mapping of the global schema is the key issue, but to integrate the schemas involved.

On the other hand do not forget about correction costs. Should the full matching be the priority, one cannot leave out the recall maximization. What is more he/she is nearly required to do so. Here is the why: in the vast majority of cases the matching pair set is considerably smaller than the non-matching. So when filtering out wrong matches – because precision was not the key issue – he/she has to overview a smaller set than if he/she had to look for more not found coherent pairs. So if we presume that the correction in both cases – adding a not found and removing a wrongly found pair is the same then we can conclude that the smaller the set to oversee is the better. In this case the measure to maximize is the recall.

Finally f-measure maximization remains the desirable goal if global accuracy is sought and the aforementioned considerations do not apply. This time the precision and the recall play an evenly important role, no one is preferred. Note that the best f-measure value may be compounded by an outstanding precision and a merely good average recall or vice versa. This could result in a not so accurate result set or coherent pairs left out.

The final formula is generated with the help of the following auxiliary formulas (7-9).

$$P = \mathrm{sgn}\left(\mathrm{sgn}\left(w_1 N + w_2 T + w_3 A - \tau S\right) + S\right) \tag{7}$$

$$I = \mathrm{sgn}(\mathrm{sgn}(P + R - 2S) + S) \tag{8}$$

$$P = \frac{\| I \|}{\| R \|} \qquad R = \frac{\| I \|}{\| P \|} \tag{9}$$

The base task and the ultimate formula are given below in formula 10 and 11. Formula 10 shows the base task of the F-measure maximization and formula 11 shows the base task of the Precision and Recall maximization.

$$\max(F) = \max\left(2\frac{PR}{P + R}\right) \tag{10}$$

$$\max(P) \qquad \max(R) \tag{11}$$

With the above listed formulas the accuracy maximization problem unfolds in to a far less complex

problem. Each maximization target measure may deliver different solution for the same scenario. For every single calibration task, the measure on which the optimization is performed should be decided based on the schema matching objective.

5.3 Multiple thresholding

Scrutinizing the problem in its depths, it may come into the sight of the heedful observer that the choice of the threshold plays a crucial role. Maybe this role is a bit too overestimated, while it is far too obvious that we should decide on an exact value in the end. To alleviate troubles ensued we could choose to have many of them, not a single when. This is the point where multiple thresholding emerges.

Multiple thresholding can be described as the post-run analysis where the semantic distances are interpreted into matching pairs, however not with a double choice estimation but with a multiple choice. The semantic values are then categorized based on their values. In other words the result set are tested against value ranges delimited by given thresholds. Should they fell in a certain category then label the pair with that. If it makes sense by the categorization used we allow for a given pair to fall into multiple categories; that is we permit overlapping ranges.

The benefit is self explaining. With the deployment of this technique we can avoid those cases when a non-matching value suppresses the threshold value even by less, or coincidentally a coherent pair falls just below the threshold. This can have serious impact on the accuracy, not mentioning the effort necessary to seek those misjudged pairs. With multiple thresholding the cost of error is not so outrageous, as categorizing a “strongly coherent concept pair” to “coherent concept pair” or even “possible concept pair” is not as expensive as categorizing it to “non-matching” pairs. Staying by the example, other reason in favor of multiple thresholding is that “coherent pairs” enumerates most likely less elements as “non-matching” for reasons discussed in 5.2. For this the correction costs are reduced.

How then should be defined these thresholds? Using equal ranges is not advantageous because some categorize may count more elements than others, besides we may lose the benefit gained from multiple thresholding. Normally we focus on values near the threshold, and we are not interested by how much a non-matching is non-matching. We propose to use small ranges near to the threshold, medium ranges in above the threshold and one or maybe a few large range(s) below the threshold. Using this recommendation leads to a clear classification of concept pairs. We may decide to execute the integration exclusively the topmost categories involved or embrace also pairs in the vicinity of the threshold.

When rapid and accurate matching is the key issue we should decide to optimize on precision and then perform the multiple threshold matching. The result list generated from the elements which are to be found in the top categories. However this proceeding we may leave out several matching values, consequently further inspection is of the result list is normally highly recommended.

6 Experimental Results

The experiments were performed on test schemas taken from the literature and created specifically for the challenges of the calibration task. Hereby we would like to introduce three of them. Others have showed similar results and the schema phenomena can already be observed through introducing three schemas with different structures. We have also executed our tests on schemas extracted from open standards, but for manageability reasons we only worked on small slices of them.

The first schema models an enterprise architecture. The to be integrated schema A comprises a CompanyData object, which encompasses all the information related to the company, e.g. name, address,

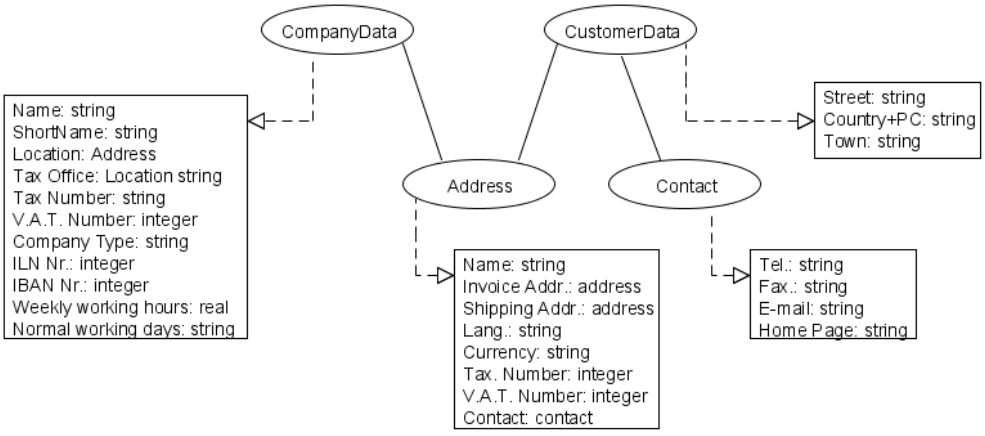

Fig. 1 Schema A of the company test scenario

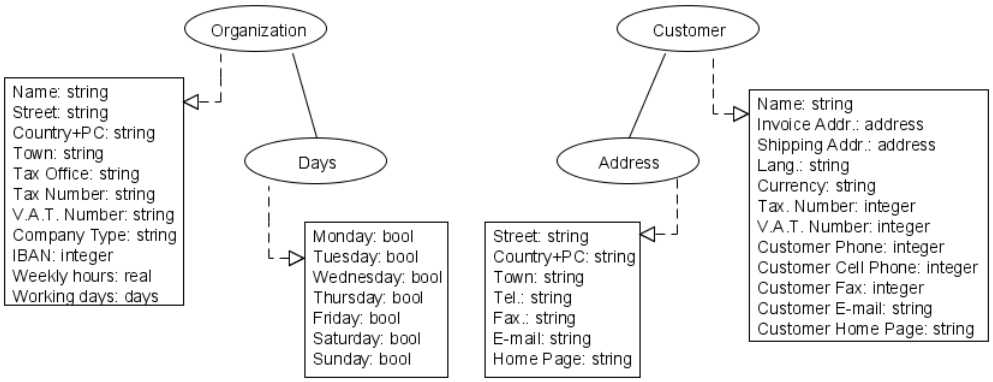

Fig. 2 Schema B of the company test scenario

tax number etc. Its child concept is the Address, which enumerates concepts to describe the address.

The customer is represented in the schema by the CustomerData which mainly handles all the necessities to contact the customer and for the billings, e.g. invoice and shipping address. Other built-in concept is the Contact which includes all the data to get in touch with the customer.

Schema B is very similar to schema A, but it was created using other naming conventions. For example when referring to company and customer it does not suffix the data annotation. Contact node is omitted its role is taken over by the Address, but as new a concept Days is introduced which is logical enumeration to register working days. Figure 1 and 2 shows the graph representation of company schemas A and B.

In contrast to the enterprise schema the university schema is deeper. This schema pair shows structural resemblances which maybe harder to detect for machines. The schema can be divided into units, so the professional-operational division let themselves better distinguish. Nonetheless University also comprises such complex attributes which appear on numerous branches so they are packaged into separated complex types.

Schema A and B show a lot of identical features, after a rash scrutiny they may seem to be quite the same. Nonetheless this schema holds many challenges for the matching algorithms. In verity they have some very meaningful, though not obvious difference. In schema B the address is a complex entity, while in A it is defined as a simple attribute. The University employs some researchers, who write publications. The publication can be either journal-article or proceedings-article or book. These concepts are further detailed with their attributes.

On the other hand, schema A maintains only simple attributes about the address of the university and what is more important the publications are managed not through the researchers but through the library giving a different idea about publications. The schema tells apart among book, monograph, article and journal. To them complex author attribute is defined whose address is maintained by the address entity which similar to that located in schema B.

The third schema is similar to the university regarding the structure of it but it lacks some of the complex matching challenges involved in the university. This time the main task for the automated matcher that same entities are rendered entirely differently, that is with other terminology and structure.

In schematic graph we can easily realize coherent pairs and they differ from one another. The main concept is called Autotrader in schema A, while Carseller in schema B. Both concepts have stocks (Stock - Inventory) and staff (Crew - Staff). In addition one maintains a showroom. The staff is divided into manager and members in case of schema B, while they are handled as one conceptual unit in the service schema. Schema B also defines available options for the cars, though schema A not. Figure 3 and 4 show the graphs of test scenario Cartrader.

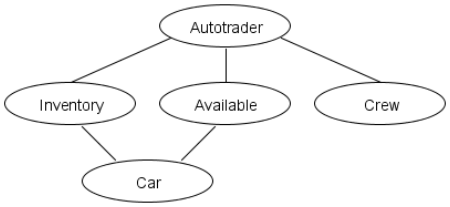

Fig. 3. Schema A of test scenario Cartrader

We have conducted several calibration experiments on test schemas. Hereby we would like to focus on the results returned for the NTA algorithm on three test schemas. In table below the reference-approximation weights and thresholds are presented. The runtime costs of computation of these values were negligible. We can observe how schema dependent the ideal parameters are.

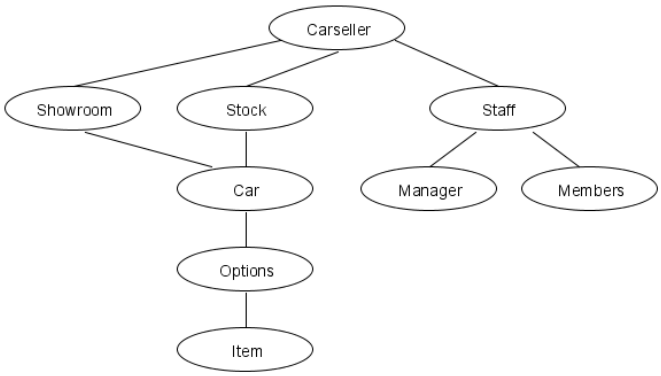

Fig. 4. Schema B of test scenario Cartrader

Table 1 Optimal parameters given by our approximation method

	w1	w2	w3	Threshold
Company	0,149	0,225	0,625	0,291
University	0,837	0,109	0,054	0,975
Trader	0,043	0,372	0,584	0,511

We have analyzed the impact of weight adjustment on the deviation from the reference table. This graph shows the phenomenon for the Company and the University test schemas. The graph shows how seriously the actual choice of the parameters influences the deviation on a schema.

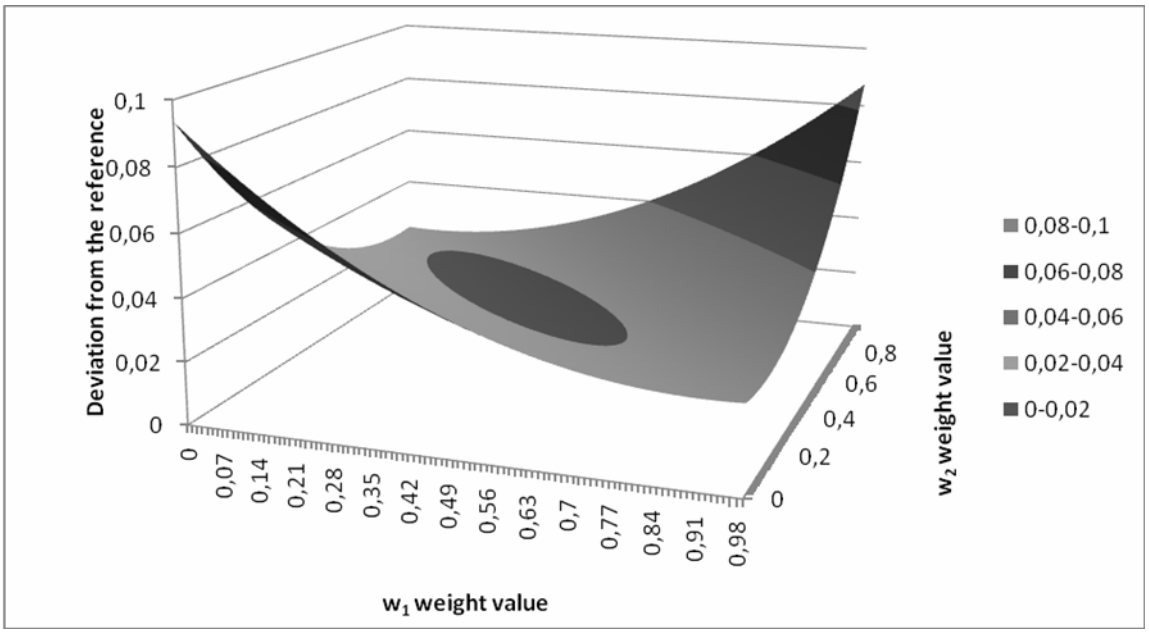

Fig. 5 Deviation from the reference table by the University test scenario

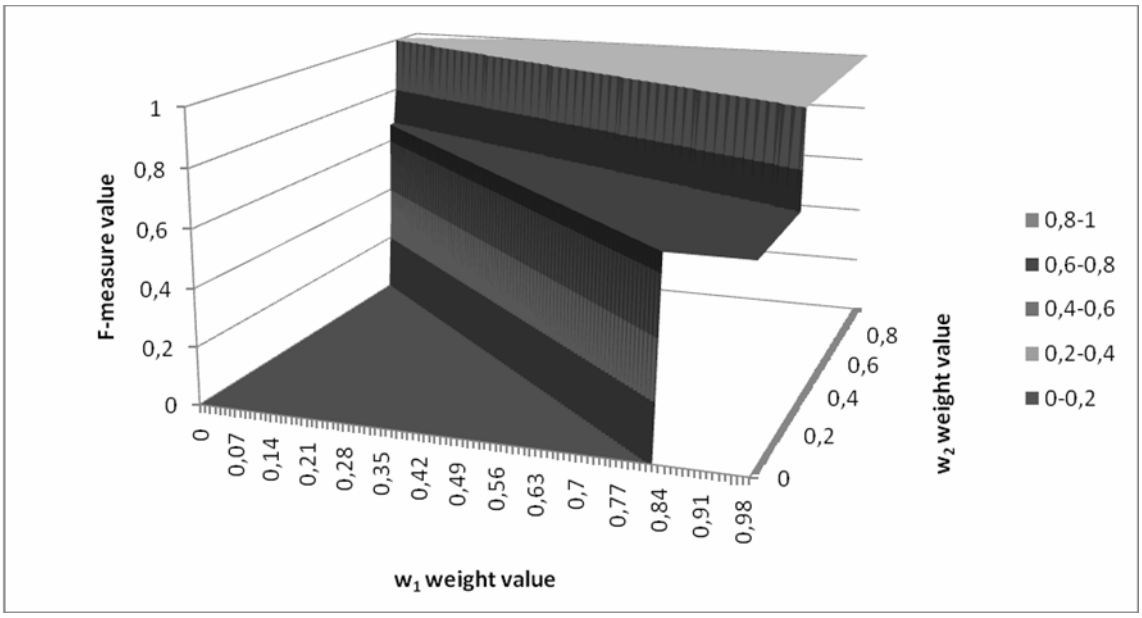

Fig. 6 F-measure values by a given threshold for the University test scenario

Based on what the table and graph suggest, it is obvious how unfair a not optimized schema matcher comparison can be. All scenarios need different parameter set in order to achieve their best possible result. If it is not taken into account it may a serious aftermath.

The actual weight setting not only influences the deviation from the reference table but also the measures. Given above is graph of f-measure values in the function of weights. The required threshold value is defined as the one returned by the reference approximation approach.

7 Discussion

Tests were performed on large real life schemas with promising results, though deployment of schema matching techniques in these scenarios entails further considerations. This statement not only pertains to the methods presented but also the matcher algorithms themselves. Others also noted the performance of the algorithms may substantially regress as the size and complexity of input schemas grow.

This phenomenon also underlines the necessity for evolving techniques that should solve the problem for arbitrary chosen scenarios. These techniques as remedies could ease trouble often confronted by real life integration tasks. Still in their infancy, though the algorithms performs pretty well by smaller schemas. In the near future we plan to take serious measures to handle this problem. As a first step we planned and executed tests on standards such the OAGIS and XCBL. As we experienced the problem of larger schemas, we also found that by fracturing schemas into sub-schemas the performance improves a lot. Although this is not always possible, we will conduct further tests to see how exactly these large schemas are affected by introducing alleviating factors, such as the pre-run calibration. Schema fracturing may prognosticate the solution for real life deployment, though it should be always used with great care and consideration.

We also strive to scale existing solution which may entail serious alterations and modification to the original method. By larger schemas the runtime is so overwhelming that online usage is nearly impossible. These matching tasks, on the other hand, should be performed several times not only ones as number of schemas grow and grow. Simple matchers – matchers which return they result in not too complex runtime – would also be desirable, though only the NTA is eligible to that. The WordNet-based matcher could also be among the candidates but it loses its good starting position because of the time consuming dictionary browsing. We try to evolve techniques that could cut down on this superfluous time expanse. This could be mainly done by predefined linguistic values. Other issue emerging when concerning larger schemas is number of comparisons as it may grow as context similarities involve more comparison. This step however cannot be obviated as all similarity values are needed.

Currently we have calibrated the algorithms the given scenario (if possible), and evaluated their performance under these fair conditions. Other factors, such as the effect of the weight adjustment to the diversion from the reference table and also to the measure values were examined and expressed. This showed us how the scenario influences the accuracy measurement outcome. We found that the factors may vary on a large of scale or just stay seemingly untouched by the modification of the weights based on the choice of the test schemas. We concluded that this has serious corollaries. In some cases the calibration task bears greater importance than by the others. It is also true that by some cases optimization to the local optima instead of the global optimum sometimes has no visible effect, sometimes results in a serious movement on the accuracy scale. Now the question arises when is it profitable to use these techniques. Of course when it results in a considerable improvement but it is hard to tell in advance.

Besides the gained evenhanded test configuration, we provide means to better compose algorithms thus providing more accurate results. According to our recommendation the pre-run calibration should be executed as a first step and only after that perform the composition of schemas. We emphasis though that the employment of composite schema matcher may result in a not negligible overhead. That is why our focus fell primarily on simple matchers. We also believe that the accuracy of composite matcher are also achievable with simple one, but not with same runtime costs. However the fact worth mentioning that when runtime is not aspect, for example in an offline integration scenario, composite matchers remain a valid option, for combining the best results available may further improve their accuracy.

8 Conclusion

We scrutinized the tested methods for three test scenarios. Such schemas were selected that as a whole they represent a wide range of real life schemas. We have paid special attention to embrace various types of schemas.

Finished with all tests we have summarized the accuracy values gained in order to assess their accuracy. The performed experiments showed the applicability of our method and formulas. Using our approach it is now already possible to evaluate and compare the accuracy of different schema matching algorithms in a correct way. This fair treatment is one of the side achievements introduced with these techniques. We have probed the

candidates under similar conditions and remained unbiased. With that in mind we gained a clear picture about the actual performance and potentials of each algorithm.

References:

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

[2] Bergamaschi, S., S. Castano, and M. Vincini: Semantic Integration of Semistructured and Structured Data Sources, *SIGMOD Record*, Vol. 28, Issue 1, 1999, Pp. 54-59.

[3] V. C. Bhavsar, H. Boley, L. Yang, A weighted-tree similarity algorithm for multi-agent systems in E-business, *Computational Intelligence*, Vol. 20, Issue 4, pp. 584-602.

[4] D. Buttler, A Short Survey of Document Structure Similarity Algorithms, *Proceedings of the 5th international conference on internet computing*, 2004.

[5] Artem Chebotko, Mustafa Atay, Shiyong Lu, Farshad Fotouhi, XML subtree reconstruction from relational storage of XML documents, Data & Knowledge Engineering, Vol. 62, Issue 2, 2007, pp. 199-218.

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

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

[8] 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.

[9] Buhwan Jeong, Daewon Lee, Hyunbo Cho, Jaewook Lee, A novel method for measuring semantic similarity for XML schema matching, *Expert Systems with Applications*, Vol. 34, Issue 3, 2008, pp. 1651-1658.

[10] J. Madhavan, P. A. Bernstein, E. Rahm, Generic Schema Matching with Cupid, *Proceedings of the 27th International Conference on Very Large Data Bases*, 2001, pp. 49-58.

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

[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] Richi Nayak, Wina Iryadi, XML schema clustering with semantic and hierarchical similarity measures, *Knowledge-Based Systems*, Vol. 20, Issue 4, 2007, pp. 336-349.

[14] Palopoli, L. G. Terracina, and D. Ursino: The System DIKE: Towards the Semi-Automatic Synthesis of Cooperative Information Systems and Data Warehouses, *Lecture Notes in Computer Science*, Vol. 2282, 2002, Pp. 228-276.

[15] H. Quyet Thang, V. Sy Nam, XML Schema Automatic Matching Solution, *International Journal of Computer Systems Science and Engineering*, Vol 4., Num. 1, pp. 68-74.

[16] E. Rahm, H.H. Do, S. Massmann, Matching large XML schemas, *SIGMOD Rec*. 33(4) (2004).

[17] Anton V. Riabov, Eric Bouillet, Mark D. Feblowitz, Zhen Liu, Anand Ranganathan, Wishful Search: Interactive Composition of Data Mashups, *Proceeding of the 17th international conference on World Wide Web*, 2008, pp. 775-784.

[18] Khalid Saleem, Zohra Bellahsene, Ela Hunt, Performance Oriented Schema Matching, *Lecture Notes in Computer Science*, Vol. 4653, 2007, pp.844-853.

[19] 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.

[20] D. Theodoratos, T. Dalamagas, I-T.Liu, Semantic Integration of Schema Conforming XML Data Sources, *Lecture Notes in Computer Science*, Vol. 3806, 2005, pp. 588-589.

[21] 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**.**

[22] 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.