Periodica Polytechnica
Electrical Engineering
and Computer Science

61(1), pp. 1-11, 2017
DOI: 10.3311/PPee.8993

RESEARCH ARTICLE

DIPROM: DIstance PROportional Matcher Exploiting Neighbor-levels and Related Terms

Balázs Villányi[1*], Péter Martinek[1]

Received 08 January 2016; accepted after revision 06 July 2016

Abstract

Schema matching has the task to find semantically related elements in the input schemas. Many automated schema matchers have been proposed, but none of the solutions performs consistently without mismatches. In this paper, we propose a new schema matcher which contains enhanced matching techniques in its components to improve accuracy. Especially, this approach exploits the so called related term sets to provide a more accurate matching, quickly. Since the related terms sets are rarely provided, a process is also proposed to extract related term sets from schema descriptions. Another specialty of our schema matcher is the definition of entity neighbor-levels. This technique is applied to evaluate the neighbor-level similarities and include these values in the assessment of entity relatedness. Our approach has been shown performing reliably and has been compared with other schema matchers both as a standalone hybrid schema matcher and component-wise.

Keywords

schema matching, hybrid schema matcher, linguistic matching, related term set matching, structural matching, related term set extraction

[1] Department of Electronics Technology,
Faculty of Electrical Engineering and Informatics,
Budapest University of Technology and Economics,
H-1521 Budapest, P.O.B. 91, Hungary

[*] Corresponding author, e-mail: villanyi@ett.bme.hu

1 Introduction

Schema matching has the task to identify semantic correspondence in two or more input schemas. This semantic correspondence is set up by identifying those schema entities in the input schemas which represent the same real world entity. The automation of this task is essential since the manual execution is time-consuming and error-prone. Thus there is a deep research interest in finding efficient and accurate automated schema matching algorithms [1].

Current schema matchers are semi-automated meaning that their resulting schema match needs manual adjustment, which has the aforementioned drawbacks. Our goal is to lessen the need for this follow-up manual adjustment by increasing the schema matcher accuracy. In this paper, we provide a novel schema matching approach with higher accuracy.

Schema matching plays a key role in several fields. Fundamentally, it can be applied in every field where there is a structured representation of data and there is a strong interest in finding semantic correspondence. The most straightforward example is data mapping and related fields, like data migration, data warehousing, data exchange and schema mediation [2]. It is also important that schema matching is not directly connected to a particular representation of structured data. Most commonly, it is performed on schema descriptions like XSD (XML Schema Description), but this is not a must.

Many of the schema matchers are composed of several schema matching components. These schema matchers are referred to as composite schema matchers. The approaches employed in the components include the linguistic, structural, vocabular, constraint- and instance-based matching [1, 3]. The linguistic matching assesses relatedness in the textual schema elements using syntactic methods. A more sophisticated – but potentially more resource-consuming – approach is the vocabular matching, where an external dictionary, ontology, thesaurus or other taxonomically ordered term collection is applied. This approach facilitates the semantic relatedness evaluation of the textual elements. The structural matchers utilize the hierarchical relationships in the data stored in schemas.

The techniques mentioned so far are the most commonly used approaches. Nevertheless, there are other techniques like the instance- and constraint-based schema matcher. The former technique analyzes instances of schemas and defines schema relatedness based on instance similarity. The latter technique takes into account the schema constraints and specifies schema element relatedness based on this information.

Schema matchers using more than one matching approach are called hybrid. Every component characterizes the relatedness of entity pairs with a normalized similarity value in the [0,1] range. Usually the incorporated schema matchers – referred to as (schema matching) components – are combined through weights. Other techniques include the union, intersection [4], minimum, maximum, average [5] or application of second-line matchers (2LM) [6].

The output of schema matchers is the matching entity set which consists of entity pairs having a semantic similarity value higher than a predefined threshold. These pairs intended to be the elements representing the same real world entities (e.g. customers, products or their attributes like name, address, age or size, color and price respectively) in the input schemas. There are several ways to establish the set of matching pairs based on the similarity value: thresholding [1, 3, 4], Maximum Weighted Bipartite Graph algorithm [6-8], along with the Hungarian Method [9], Stable Marriage algorithm [6] and Dominant pair algorithm [6]. Many of the classifiers known from the field of machine learning can also be used. The majority of these techniques, however, are limited to 1:1 matching. The simplest, most wide-spread technique – the thresholding – enables *1:1*, *1:n*, and *n:m* matching as well, hence we used this general approach to evaluate the performance of our proposed schema matcher.

Following our earlier research effort, we have developed our schema matching framework [10] capable of the fair comparison of the input schema matchers. We have thoroughly analyzed these results with special focus on the schema matching component ranking. In this paper, we propose a new hybrid schema matcher which is based on the conclusions drawn from the component ranking research. The proposed composite schema matcher consists of a linguistic, a vocabular and a structural component.

Our paper is structured as follows. This first section is the introductory part. The second section enumerates some of the related works. The third section presents the detailed description of our proposed schema matcher. It has four subsections: the linguistic matcher, the vocabular matcher, the structural matcher and the component composition. The forth section reports the evaluation of the proposed schema matcher. Lastly, the conclusion and future works are summarized in the fifth section.

2 Related works

Several automated schema matchers have been proposed by the schema matching community like [11-13]. In this section we will iterate over those which had influence on our work. Afterwards, some of the reviewing and optimization works in the field of schema matching will be discussed.

Similarity flooding [11] capitalizes on the underlying presumption that two schema nodes are similar if linked nodes are similar. In order to exploit this presumption on algorithmic level, node similarities are propagated in a specifically designed joint schema graph called similarity propagation graph. An important factor is the edge weight-setting in the similarity propagation graph as it defines the extent of similarity contribution of the adjacent nodes. Our structural matcher employs a similar approach, but the extent of similarity contribution can be adjusted by contribution functions. Also, we extended the concept of similarity contribution by taking into account not only adjacent nodes, but farther-lying nodes as well. The NTA algorithm [12] has three components: a linguistic, a vocabular and a structural matcher, just like our proposed approach. Its name comes from the abbreviations of names, terms and attributes since these are the main concepts, the similarities of which are evaluated by the components. The most efficient component of this approach is its vocabular matcher: the related term set matcher. The problem is that the related terms set expected by this approach is not as commonly available as entity descriptions. In this paper, we also provide a solution to generate the related terms sets and propose an enhanced method to evaluate their relatedness. Lastly, the attribute similarity is a recursive structural schema matching component which compares entity descendants. Our proposed approach refines this recursive approach by setting the extent of similarity contribution for each node distance. Also, our proposed approach evaluates similarity not only among the descendants, but also among the ancestors.

The schema matcher proposed in [13] has a context-based structural schema matcher. It defines three contexts: ancestor, child and leaf contexts. The linguistic component exploits the WordNet [14] lexical database to evaluate the relatedness of the node labels. This schema matcher utilizes more than one weight-setting, which may render this solution more vulnerable to schema matching scenario changes. Our proposed schema matcher also takes into account the various contexts treated by [13], but through the neighbor-levels, not separately.

An excellent overview of the current schema matching solution is found in [3]. Besides formally defining the match operator and architecture, this work also enlists and describes some of the application fields of schema matching, like e-commerce, schema integration and data warehouses. The distinction between instance- and schema-level matchers is also discussed. This survey comparatively evaluates schema matchers SemInt, LSD, SKAT, TranScm, DIKE, Artemis, Cupid. This work helped us to adequately categorize the components of our solution and to find an appropriate schema matcher component combination.

The book edited by Bellahsene, Bonifati and Rahm [15] is a compilation of state-of-the-art schema matching techniques. It is divided into three sections. The first part is about large-scale

schema matching. The challenges are described in [4], where several strategies are proposed to cope with large schemas, e.g. early pruning of the search space, parallel and holistic schema matching. A general workflow for automatic, pairwise schema matching is also proposed, in conjunction with which several large-scale optimization techniques are offered. The techniques like early pruning of dissimilar element pairs and partition-based matching can promote the large-scale applicability of our proposed schema matcher. The second part of the book is dedicated to the subjects of schema matching as a logic problem, the schema evolution and merging. Schema matching is formally described in [16]. The schema matching task is formed as first-order logic formulas and is solved using the chase procedure. This paper describes concrete techniques how queries can be answered in schema matching scenarios, hence it provides another possible application field for our proposed schema matcher. Nevertheless, authors conclude by mentioning the high level of redundancy and the low level of expressibility of schema mapping GUI tools in the current schema matching solutions. The last part of the book is dedicated to the evaluation and tuning of schema matchers. The very last chapter [17] serves as an overview on the field of schema matcher tuning and proposes a novel optimization approach. It has three levels: parameter setting, the selection of similarity value combination strategy and the selection of the schema matcher. Along with [5, 6, 10], the techniques described in [17] provide means to find the appropriate parameter-setting of our DIPROM. The schema matcher tuning also helped us to find efficient components for our proposed schema matcher.

The schema matching optimization approach called eTuner [5] helps to find the optimal combination of schema matchers. The main concept is the synthetic workload: a predefined schema matching scenario with reference matching and the derived schemas using schema transformation rules like label abbreviations, substitutions, etc. The other unique aspect is the staged tuning, which is a bottom-up approach: firstly the matchers, then their combination, finally the whole matching is tuned. Another tuning approach is found in [10], where a methodology is proposed which offers different optimization targets stemming from the alternative interpretations of schema matching accuracy: one is defined as the mean squared error from the reference match and the other utilizes the accuracy measures. The proposition includes several optimization techniques targeting different objective functions. A related research is found in [6], where the concept of Schema Matcher Boosting (SMB) is introduced. This proposal aims the automated selection of schema matcher to a composition (ensemble) based on a proposed error measure. To further refine the optimization process, schema matchers are divided into two groups: first- and second-line schema matchers. While the former group encompasses any existing schema matchers, the latter contains techniques which refine the results given by these schema matchers. The paper discusses a comprehensive set of related techniques, which can be used in conjunction with SMB.

Another summarizing work by Gal [7] focuses on the inherent uncertainty of schema matching. This work is based on the conclusion drawn in many related previous works: schema matches provided by the current solutions always incorporate uncertainty, though they level varies. The reduction – or possibly the complete elimination – of uncertainty would be highly desirable, but until then, we should find techniques dealing with this uncertainty. At this point, the propositions in [7] constitute an excellent basis for this purpose. Also, different aspects of uncertainty are presented in this work. Two notable solutions for the problem of schema matchings are described: the top-K matching and ensembles. Top-K matching is also presented in [8], while the ensemble approach in [6]. It focuses on the selection of best matches instead of the production of a single best match. Nonetheless, the authors aimed at keeping the complexity as low as possible. This aim is of special importance when managing several matchings simultaneously. The used schema matching parameters are geared towards higher precision values (at the cost of lower recall values). Top-K matching is a general approach leveraging a weighted bipartite graph which involves multiple schema matches. Thus this method can be used to refine the matches produced by several schema matchers like the one proposed hereby.

3 The matching approach of the DIPROM

In the followings, we will describe our proposed hybrid schema matcher. The solution incorporates three schema matcher components: a linguistic, a vocabular and a structural. By every component the algorithm was developed based on the best approaches found in current researches in the field of schema matchers [10]. The most important influencing techniques are mentioned in the previous section 2.

In order to identify the best approaches, we used the Comparative Component Analysis [10] for unbiased and comprehensive comparison of existing schema matching approaches. This helped us to define the working mechanism of the components, especially that of the neighbor-level-based similarity propagation.

3.1 The logistic homosequence linguistic similarity

Linguistic matching is the primary element of schema matching, which is used to evaluate textual elements in schemas. This technique does not take into account the relationships among entities, only string literals like names and types. This stipulation may render the linguistic matchers unduly limiting, but in fact, they are capable of justified matching decisions besides being efficient. We have found that those linguistic matchers that identify common substring sequences (homosequences) are the most accurate. Also, we found that long substring matches should be rewarded more than it would proportional

to the substring match length. Conversely, short substring matches should be rewarded less than it would proportional to the substring match length. Consequently, our schema matcher utilizes the proposed logistic homosequence linguistic similarity, which is a substring matcher with logistic activation function. In (1), the *affix()* function denotes the common substring, while $|x|$ denotes the length of variable x.

$$\mathrm{LHS}(s_1, s_2) = \frac{1}{1 + e^{-10\left(\frac{\max(|affix(s_1,s_2)|)}{\max(|s_1|,|s_2|)} - 0.5\right)}} \quad (1)$$

For numerical examples, see Table 1. The first row (Addr-AddrData) is calculated as follows. Firstly we determine the length of the longer term, which is the eight given by the string literal "AddrData". Then the lengths of the longest shared substring should be examined. In this case the longest substring is "Addr", thus the substring length is four and null. Following Eq. (1), we should end up having $1 / (1 + \exp{-10}(0.5 - 0.5)))$ as LHS similarity.

Table 1 Numerical examples for linguistic homosequence similarity

First String Literal	Second String Literal	LHS similarity value
Address	AddrData	0.5
Address	AddrData	0.796
Company	Comp	0.67
Company	CMP	0.1

The formula in (1) warrants that the linguistic matcher gives higher similarity values to those entities which have substring matches longer than the half of the longer label.

3.2 The related term set

Related term set helps us to specify the intentional meaning and domain of an entity. This property of the related terms is of particular importance in cases where the denomination of the entities does not suggest the precise meaning and domain (e.g. because of inadequate denominations). The related term set may be useful when trying to identify semantically related entities if they use significantly different entity denominations, abbreviations, synonyms, etc. The description of the related term sets can be formulated as follows: The related term set of a schema entity is the collection of terms, keywords, phrases and expressions given in the same language which circumscribe the given schema entity. The related term set may include synonyms, supercategories, subcategories, and close associations.

We should note that the related term set also contains the given entity denomination itself by definition, since this term also circumscribes the entity. For example, the related term set of the term "invoice" may include: receipt, bill, acknowledgement, customer documentation, payment, sum to pay, address and invoice.

As stated earlier related terms set based entity relatedness evaluation is an effective schema matching approach [10, 12]. It is less complex and requires less computational resource than other vocabular schema matchers. In order to better exploit the hidden capabilities of this approach, we have created an enhanced version of the related term set evaluator found in the NTA which embraces term frequency as well.

Unfortunately the related term set – effective as it is – is rarely available in schema definitions. Naturally, this does not mean that there would be no means to generate it. Thus we also set the objective of designing a process to extract related term sets from available schema descriptions.

3.2.1 Related term set extraction

As stated, the effective related term set based schema matching can only be exploited if the appropriate related term set is present. Hence we developed a process [18] which automatically generates related terms sets with term frequencies using text mining techniques [22, 23]. Although this proposed process is not the focal point of this paper, we describe it for the sake of completeness.

In the related term set extraction process, we used the following techniques:

- **Information Extraction (*InfExt*):** identifies potential sources of entity description. Entity descriptions may have diverse sources. Besides finding entity description elements in (structured) schema definitions; entity denomination, entity categorization, super- and subcategory labels, etc. may be used as related term. Also, scattered entity descriptions should be merged before the related term set extraction could take place.
- **Tokenization (*Tok*):** splits texts into individual words. The splitting may take place on the basis of simple or complex conditions. Simple conditions may include whitespace or other single splitting characters. Complex conditions may include regular expressions or other text-context-dependent conditions. Each word is represented by a token.
- **Letter Case Reconciliation (*LCR*):** transforms letters to lower cases. Words may occur with different letter cases for several reasons. This may impede the identification of identical related terms. For this reason, all of the letters are transformed into lower case.
- **Stop Word Filtering (*SWF*):** eliminates meaningless words from the extracted word lists. There are words which do not have any special importance in the semantic specification of a given entity. These words are used only to connect meaningful terms in descriptions or have other grammatical role. Nevertheless, they have to be removed as they have no place in a related term set. This filtering is carried out using a stop word list.
- **Length-based Token Filtering (*LTF*):** eliminates short words. The matching of short – mostly

meaningless – words does not imply the relatedness of the given entities. Nonetheless, this filter should be applied carefully as short terms may include important abbreviations too. (When using the Length-based Token Filtering, it is better to expand abbreviations so that the filtering only eliminates meaningless words.)

- **Stemming (*Stem*):** shrinks words to base form and nominalize them if not noun. Linguistic matchers – often applied in related term set matching – lag behind in performance significantly if same words appear with different prefixes/suffixes or as different part of speech. Hence base form is required so that the term matching can be carried out.

The process is detailed in Alg. 1. We used the following notations. The input schema is denoted with $\mathbb{S}$. The string array ι contains texts extracted from the input schema $\mathbb{S}$. The entity descriptions in the string array ι are denoted with δ. The entities in the schemas will be referred to with ε. Subsequently ω^0, ω^{SWF}, ω^{LTF}, and ω^{ε} are the string arrays obtained as output of the methods Letter Case Reconciliation (*LCR*), Stop Word Filtering (*SWF*), Length-based Token Filtering (*LTF*), and Stemming (*Stem*) respectively. Lastly, we define the term frequency vector f^{ε} with the same dimension as ω^{ε}, and the related term list L^{ε} of entity ε.

Algorithm 1 The related term set extraction process

Description: *This algorithm generates related term sets for schema entities.*
Input: $\mathbb{S}$ – the input schema
Output: $\mathcal{L}^{\epsilon}$– list of entities with related term sets

```
1:  procedure ExtractRelTerm(𝕊)
2:      ι ← ExtractInf(𝕊)
3:      ℒ^ϵ ← entities in 𝕊 with related term sets r^ϵ ← ∅
4:      for all δ ∈ ι do
5:          ω^0 ← Tok(δ)
6:          for all c ∈ ω^0 do
7:              c′ ← LowerCase(c)
8:              Substitute c with c′ in ω^0
9:          end for
10:         ω^SWF ← FilterStopWord(ω^0)
11:         ω^LTF ← FilterShortTones(ω^SWF)
12:         ω^ϵ ← Stem(ω^LTF)
13:         Initialize array f^t with identical dimension as ω^ϵ
14:         for i ← 1, ||ω^ϵ|| do
15:             f^ϵ(i) ← the occurrences of term ω^ϵ(i) in δ
16:         end for
17:         Add terms ω^ϵ to the related term list r^ϵ of entity
    ϵ in ℒ^ϵ with term frequency vector f^ϵ
18:     end for
19:     return ℒ^ϵ
20: end procedure
```

3.2.2 Related term set comparison

In the previous section, we proposed a process to automatically define related terms with auxiliary term frequency information to schema entities. This serves the purpose of being able to perform schema matching based on the related term set.

In this section, we propose a method to evaluate the similarity of two related term sets. The following denotations are used: r^{ε}_1 and r^{ε}_2 are the compared related term sets (vectors). Terms t_1 and t_2 denotes two elements of r^{ε}_1 and r^{ε}_2 respectively. f^{ε}_1 and f^{ε}_2 denote the term frequency vectors belonging to related term vectors r^{ε}_1 and r^{ε}_2. $f^{\varepsilon}_1(t_1)$ denotes the frequency of t_1 in f^{ε}_1. LHS(t_1, t_2) is the logistic homosequence similarity of terms t_1 and t_2. Using these denotations related term set similarity can be calculated with (2). Note that each term t_1 in the related term set of r^{ε}_1 is compared to its best matching counterpart t_2 in the related term set of r^{ε}_2, and is weighted by the relative term frequency.

$$\begin{aligned}\text{Terms}\left(r^{\varepsilon}_1, r^{\varepsilon}_2\right) = &\frac{1}{2}\cdot\sum_{t_1\in r^{\varepsilon}_1}\frac{f^{\varepsilon}_1(t_1)}{f^{\varepsilon}_1\cdot 1}\cdot\max_{t_2\in r^{\varepsilon}_2}\text{LHS}(t_1,t_2)\\ &+\frac{1}{2}\cdot\sum_{t_2\in r^{\varepsilon}_2}\frac{f^{\varepsilon}_2(t_2)}{f^{\varepsilon}_2\cdot 1}\cdot\max_{t_1\in r^{\varepsilon}_1}\text{LHS}(t_2,t_1)\end{aligned} \tag{2}$$

Expressed by (2), our proposed related term set comparison method works as follows. Given the related term sets of two compared entities, we should find the best matching counterpart for every related term in the other set. The best matching counterpart is determined by the similarity values given by the linguistic matcher. One of the key elements of our proposed matcher is the logistic homosequence, which proved itself trustworthy (cf. Section 4). Hence we use this linguistic evaluator even for the related term set comparison. Once again, this similarity value is weighted by the relative term frequency, i.e. the term frequency divided by the summed term frequency in the related term set. There is another key element introduced in this related term set similarity assessment: the best matching related terms are searched independently in the two input related term sets, i.e. one best matching related term in the first related term set may not be the best matching counterpart for the other related term in the second related term set. Consider the following short, motivational example. The first related term set consists of: “computational record”, ”company data”, etc. The second consists of: “company”, “corporation”. For the related term “computational record”, “company” is the best matching counterpart, but this is not true vice versa: “company data” is the best matching counterpart for “company”. This duality is also expressed by (2), where the two aspects are taken into account evenly. This approach and the LHS similarity are clear improvement of the approaches described in [12] and [18].

Table 2 contains two matched entities with their related term sets. As the human evaluator should realize, they both represent the same real world entities. However, a pure linguistic

evaluator faces a considerable challenge by this example: the entity denominations are completely different, plus the second entity has a second term in its denomination.

Table 2 Numerical examples for linguistic homosequence similarity

Related term	Term frequency
Invoice:	10
bill	2
account	1
receipt	8
customer document	2
purchase document	2

Related term	Term frequency
BillingDocument:	1
invoice	12
purchase	2
receipt	8

In the example shown in Table 2, the related term set similarity (semantic value) is 0.845 given by (2): $((10 \cdot 0.993 + 2 \cdot 0.08 + 1 \cdot 0 + 8 \cdot 0.993 + 2 \cdot 0.427 + 2 \cdot 0.427) + (1 \cdot 0.08 + 12 \cdot 0.993 + 2 \cdot 0.427 + 8 \cdot 0.993)) / (25 + 23)$. It means that even by significantly discrepant naming conventions, we were able to conclude that there is a strong semantic linkage between these two entities. For this conclusion solely the evaluation of the related term set similarity was needed.

3.2.3 Neighbor-level based structural matching

In this section, we present the structural matcher component of our proposed hybrid matcher. The approach was inspired by the assumption that two entities are semantically similar if schema graph nodes in the vicinities are similar [11]. Also, we intended to exploit the matching of farther-lying schema elements. Hence we developed a method which evaluates the similarities of nodes at a given (node) distance, then contributes this similarity to the compared entity similarity (weighted inversely proportional to the distance). As opposed to what was described in [19], we have now incorporated the logistic homosequence similarity even in the neighbor-level based structural matching. The other main improvement is the dual search strategy for the best matching counterpart in a given neighbor-level as it is described in the previous section.

Firstly, we introduce the concept of neighbor-levels: The first, second, etc. neighbor-level of a given entity ε is the set of distinct nodes in the schema graph which are the last nodes of an exactly one, two, etc. long paths starting from the node of entity ε.

Our structural matcher works as follows. For every node in a given neighbor-level the best matching node is chosen from the same neighbor-level of the other entity. This best matching neighbor is found by the logistic homosequence linguistic matcher described in Section 3.1. After having obtained the maximal similarity value in a given neighbor-level for every node, we should evaluate the relatedness of the whole neighbor-level for the two schema entities being compared. Accordingly we take the average of the computed similarity values in a given neighbor-level. The neighbor-level relatedness – characterized by the thus obtained similarity value – is used to contribute to the relatedness evaluation of the entities being compared. The extent of contribution (the contribution weight) is given by the contribution function and is inversely proportional to the node distance of the neighbor-level. For further details, see Alg. 2 and Fig. 1 below. On Fig. 1, two entities from schema A and B are compared. The neighbor-levels are outlined with the red dotted lines. The extent of the neighbor-level contribution to the entity relatedness evaluation is symbolized with the red arrows. (The thickness of the arrows represents the extent of the contribution.) Lastly, the blue arrows show that only same neighbor-levels are compared.

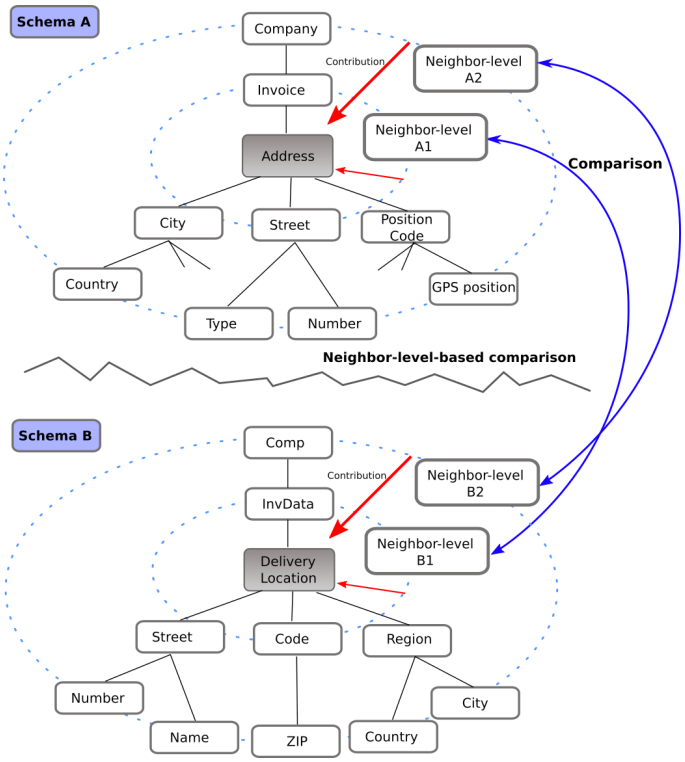

Fig. 1 The concept of neighbor-levels

Fig. 2 shows two contribution functions. An important characteristic of the contribution function is that the contribution decreases as the node distance increases. What is more, the contribution of the neighbor-levels can be parameterized. The red function dots are parameterized with the Euler's constant, so there is an accelerated drop in the similarity contribution as the node distance grows. This contribution function is used in our experiments – see Section 4. The similarity of the zeroth neighbor-level is given by the logistic homosequence similarity

of the compared entity labels. (The inclusion of the zeroth neighbor-level similarity is optional, since this value is already considered in the linguistic component.)

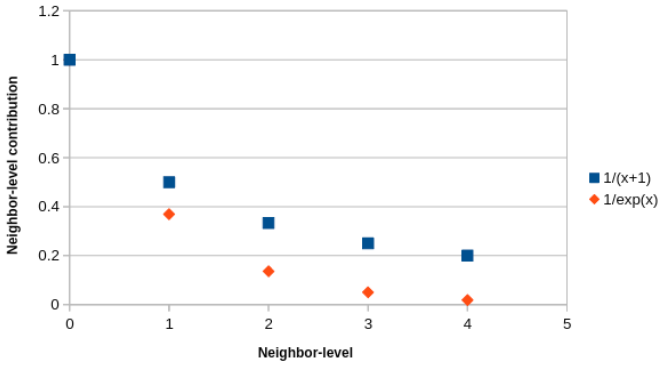

Fig. 2 The similarity contribution function

Algorithm 2 The neighbor-level based structural matcher

Description: *This algorithm evaluates the structural similarity of two input entity based on the similarity of neighbor-levels.*

Input: $\mathbb{S}_1, \mathbb{S}_2$ – the input schema graphs
e_1, e_2 – the evaluated entities

Output: r – the normalized similarity value

procedure NEIGHBORLEVELSIMILARITY($\epsilon_1, \epsilon_2, \mathbb{S}_1, \mathbb{S}_2$)
  $r \leftarrow$LHS(ϵ_1, ϵ_2) ▷ Initial similarity
  $c \leftarrow \frac{1}{e^i}$ ▷ Contribution function
  for $i \leftarrow 1, 4$ **do** ▷ Neighbor-levels 1 to 4 considered
    for $j \leftarrow 1, 2$ **do** ▷ For both entities
      $N_j \leftarrow$GETNEIGHBORLVL($\mathbb{S}_j, \epsilon_j, i$) ▷ get given neighbor-level
    end for
    for all $n_1 \in N_1$ **do** ▷ For every node in the neighbor-level
      $r_i \leftarrow r_i + \max_{n_2 \in N_2}$LHS($n_1, n_2$)
    end for ▷ get maximal LHS similarity value
    $r \leftarrow r + \frac{r_i}{e^i}$ ▷ Contribution function
    $c \leftarrow c + \frac{1}{e^i}$ ▷ Neighbor-level contribution
  end for
  $r \leftarrow \frac{r}{c}$ ▷ Normalization
  return r
end procedure

We should note that there is strong accent on the choice of contribution function. Considering too many neighbor-levels – possibly with high contribution weights – might be misleading. On the other hand, if we consider too few neighbor-levels, the application of the technique will not have its full beneficial effect. Also the depth of the schema is an upper limit for the number of inspected levels by shallow schemas.

The neighbor-level based structural similarity in the example shown by Fig. 1 is 0.385 (whereby the neighbor-level similarities from 0 to 2 are 0.0, 0.364 and 0.428 respectively, while the zeroth level is not considered hereby). It is noteworthy that the linguistic matcher could not identify the related entity pair due to the completely different denominations, but thanks to the involvement of the similarities of neighbor-levels, the similarity value is significantly boosted. With adequate component weighting, the 0.0 linguistic similarity can be entirely overturned to correctly decide on the relatedness of the evaluated entity pair.

3.2.4 The component composition of the hybrid matcher

In the previous subsections we have described the building components of our proposed hybrid schema matcher. The linguistic matching is carried out by logistic homosequence similarity. The vocabular matcher utilizes the related term set similarity described in Section 3.2. Lastly the neighbor-level based structural matching is deployed to reflect the entity relationships hidden in schema structures.

The three components individually evaluates the entity relatedness. The results are aggregated through weights given by (3) after that. Note that in (3), $l(\varepsilon)$ denotes the label, while $r(\varepsilon)$ denotes the related term set of entity ε. The functions LHS, Terms, and NLSC are the linguistic, vocabular and the structural similarity evaluators detailed in the previous sub-sections.

$$\mathrm{Sim}(\varepsilon_1,\varepsilon_2) = w_1 \cdot \mathrm{LHS}\big(l(\varepsilon_1), l(\varepsilon_2)\big) + w_2 \cdot \mathrm{Terms}\big(r(\varepsilon_1), r(\varepsilon_2)\big) + w_3 \cdot \mathrm{NLSC}(\varepsilon_1,\varepsilon_2) \tag{3}$$

The weighting is crucial and there is no globally optimal weight-setting for hybrid matcher as detailed in [10]. Nevertheless, we recommend the w = [0.2, 0.4, 0.4] weight vector for the general purpose initial weight-setting.

4 Results

We executed several experiments to evaluate the performance of our proposed schema matching approach. We have tested our approach both on real world schema standards like OAGIS [22] and XCBL [23], as well as on test schemas. Our test schemas stem from literature like [12, 13] and contain typical schema matching anomalies like discrepant naming conventions, structural differences, different level of schema granularity and multi-match scenarios. Further test schemas include company, university, vehicle dealership and purchase order test scenarios [12, 13].

The Company test scenario [12] is a relatively straightforward matching only with few entities. Its main feature to bear in mind is that it includes a *2:1* entity matching. The University scenario [13] poses a stronger challenge: it has more entities, *1:n* matching and some hard decisions (e.g. "Author" vs. "Researcher" of the "Article"). The Dealership scenario is the most challenging, since it was geared towards the misleading synonyms and the completely different naming conventions. Purchase-order is a simple e-commerce scenario borrowed from OAGIS [22] and XCBL [23].

We have measured the average accuracy (expressed in precision, recall, f-measure) of the components of our proposed schema matching approach as well as the standard deviation in the test scenarios. By every schema matching approach, we have compared our solutions with other relevant methods from the literature using the same metrics. In the followings we will see the accuracy evaluation of linguistic, vocabular and structural schema matchers separately and also as part of the assembled, composite schema matcher.

We also analyzed how the schema matchers performed in different test scenarios. Our aim was to also find correlation between the schema characteristics and the schema matcher performance, i.e. how the individual schema matchers respond to the schema matching anomalies and challenges of given test scenarios.

First our logistic homosequence linguistic similarity based matcher (LHS) was evaluated. It clearly showed a beneficial performance characteristic, see Fig. 3. Compared to the NTA name similarity, the homosequence similarity attained 0.135 higher f-measure on average, and the standard deviation was also reduced by 0.11. The same accuracy improvement can be observed in the cases of prefix/suffix matching and prefix only matching. The precision was also higher in case of nearly all of the linguistic matchers, except for the WordNet-based linguistic evaluator, but this was compensated by the higher recall value (0.83). The standard deviation varied between 0.24 and 0.33 for the precision and recall values.

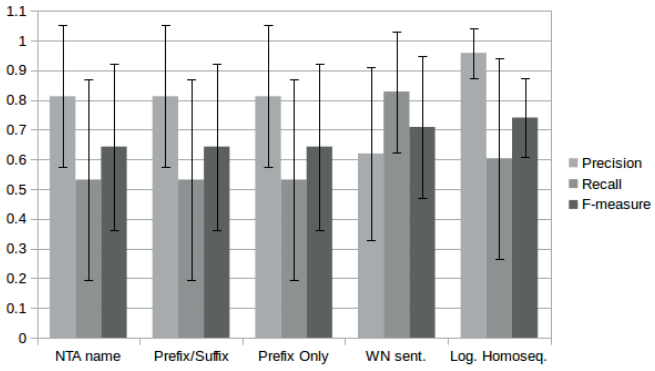

Fig. 3 Average accuracy comparison of linguistic matchers

Seeing the details in Fig. 4, we can find the reason why the precision was higher in average. It turns out that the maximal (1.0) precision was attained by every analyzed linguistic matcher in the Company and Purchase-order scenarios. This is because of to the common naming convention used in the input schemas of these scenarios. By the entity pairs where the naming conventions differ, we may witness reduced precision and recall: consult the University and Dealership scenarios. Note that the Dealership scenario involves many synonyms which are prone to fail if there are no character sequence matches in the input string literals. The result is clearly observable in Fig. 4: there is a substantial fallback in every accuracy measure. The logistic homosequence method had high precision, which means it did not return false matches, even if it could not find the true matches. In these cases, the other components should compensate for this decreased accuracy – see later in this chapter.

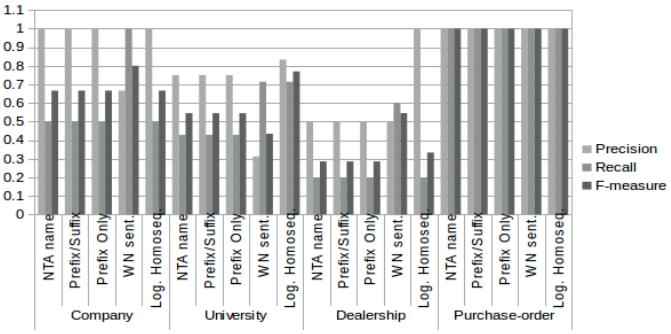

Fig. 4 Detailed accuracy comparison of linguistic matchers

Secondly we have assessed the performance of our enhanced related term set matching component. An obvious comparison reference was the related term set matcher of [12], hence we evaluated the performance of our enhanced related term set matcher compared to it. On Fig. 5, you can see that the term frequency information has a significant impact on the accuracy of the related term set matching. The observed f-measure improvement was 0.187. This improvement was attributed to the increase in recall, since the precision was 1.0 in both cases. The deviation of the recall was also reduced from 0.16 to 0.12.

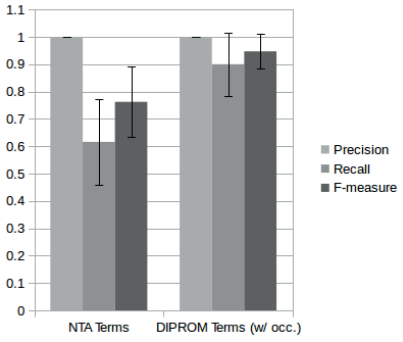

Fig. 5 Average accuracy comparison of vocabular matchers

The details of the vocabular matching can be found in Fig. 6. We can see that the precision of both matchers was indeed maximal in every tested scenario. This means that every matching entity pair proposed by the vocabular matchers was proven to be real matches. This characteristic makes these evaluators especially recommended in scenarios where precision is the target accuracy measure. The other consequence is that the varying f-measure can be importuned to the varying recall. Accordingly we can state that even the vocabular matchers struggle to retrieve matching entity pairs. It is also true that this phenomenon can be mitigated by the use of term frequency information: we can see improved recall in every scenario of Fig. 6.

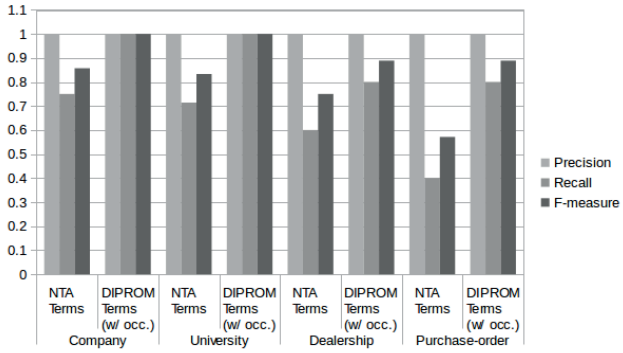

Fig. 6 Detailed accuracy comparison of vocabular matchers

Also we have compared our neighbor-level based structural matcher component with other structural matcher approaches. We compared it with the recursive attribute evaluator of the NTA [12], the Similarity Flooding [11] and context-based structural matchers of [13]. The results are shown in Fig. 7, which reveals that neighbor-level based structural matcher is capable of higher accuracy than many of its rivals. The neighbor-level similarity reached 0.188 higher accuracy than the context-based structural evaluator. The highest precision was produced by the leaf context matching. The mediocre precision of the neighbor-level based structural matcher (0.76) was compensated by its outstanding performance in the recall (0.88). The second best recall value (0.84) was produced by the NTA, but its precision value was significantly lower (0.61).

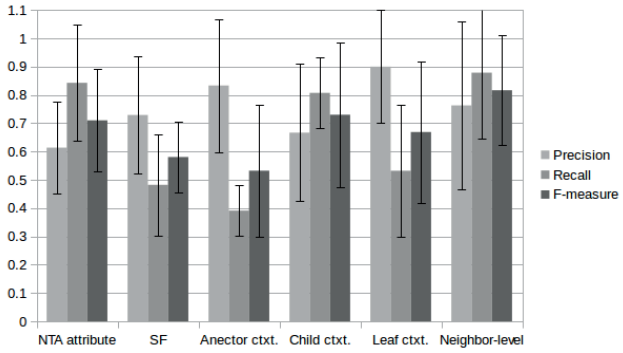

Fig. 7 Average accuracy comparison of structural matchers

Fig. 8 shows how the individual structural matchers performed in each scenario. The structural matchers performed best in the Purchase-order scenario, where the precision was 1.0 for every evaluator. The worst accuracy was produced in the University and Dealership scenarios. Even in these scenarios, the neighbor-level matching provided a slightly better performance than other approaches, with higher recall than precision. This was strengthened further by the Company and Purchase-order scenarios, where this method proved itself infallible. NTA attribute matching performed above average in the Company scenario, while the Context matching was outstanding in the Purchase-order. It is also interesting to see how different context matchers excelled in different scenarios: Ancestor context matching performed best in the University scenario, while Child and Leaf context matching in the Purchase-order scenario.

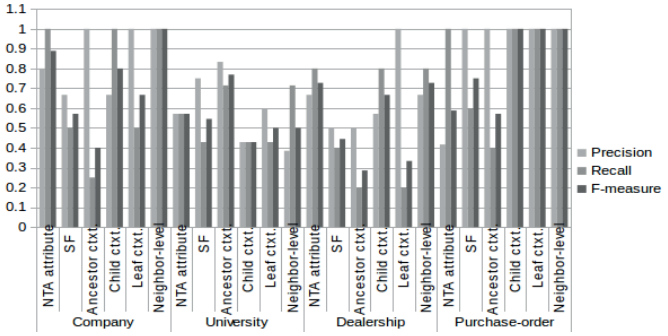

Fig. 8 Detailed accuracy comparison of structural matchers

Lastly, we built and tested our hybrid schema matcher. We have compared it with other approaches on the exact same test schemas. The outcome of this experiment is shown on Fig. 9. In order to warrant peak performance, the tested schema matchers were optimized to the test scenarios using techniques described in [10]. This calibration and the fixed test environment served as basis for the fair performance evaluation of the schema matchers. Also the threshold was set carefully so that the f-measure could be maximal under the given conditions. All in all, we attained an accuracy improvement of 0.048, 0.379 and 0.117 over the NTA, SF and WN Context matchers with our proposed composite matcher, which corresponds to a 0.182 accuracy improvement on average, with a 0.04 standard deviation among the analyzed test cases. Except of the DIPROM, the precision was higher than the recall for nearly every schema matcher (the DIPROM produced a significantly higher recall value of 0.96). The components of the DIPROM collaborated well to consistently result in both high precision and recall values.

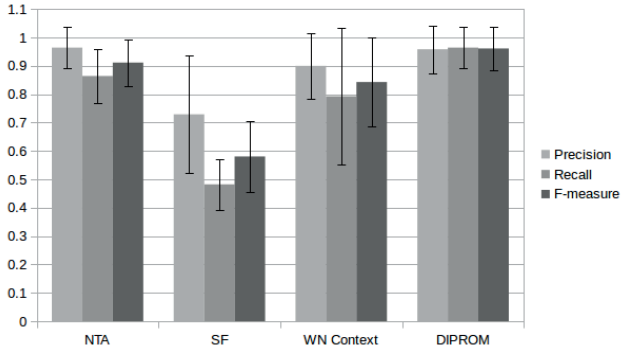

Fig. 9 Average accuracy comparison of the composite matchers

Based on Fig. 10, we may make the following observations. In the Company scenario, the NTA and DIPROM attained the highest performance possible. Besides the effective structural matchers, the main role can be attributed to the vocabular matchers. The WN Context method also performed acceptable, with its 1.0 recall value: the application of WordNet and the

aggregated power of context matchers may be praised for this. In the University scenario, a major set-back was witnessed. Some of the hard-to-find matches are beyond the capabilities of the featured schema matchers. Nonetheless, the DIPROM and NTA were top ranking. It is noteworthy that the precision was higher than the recall by every matcher. If the threshold had been set higher for the recall, the precision would have dropped significantly. A lower performance characteristic also prevailed in the Dealership scenario. As already described, this scenario is especially challenging because of its numerous synonyms. Nonetheless the WordNet-based matcher had the highest precision besides the related term set matchers. Not surprisingly the relatively high accuracy values returned for the simplest Purchase-order scenario. The WN Context matcher accomplished well: in fact, it outperformed also the NTA, which was supplied with its related terms component.

All in all, we may conclude that our DIPROM performed excellent in all scenarios. It attained its peak performance in the first and forth test scenario and suffered a modest decline in the second and third scenarios.

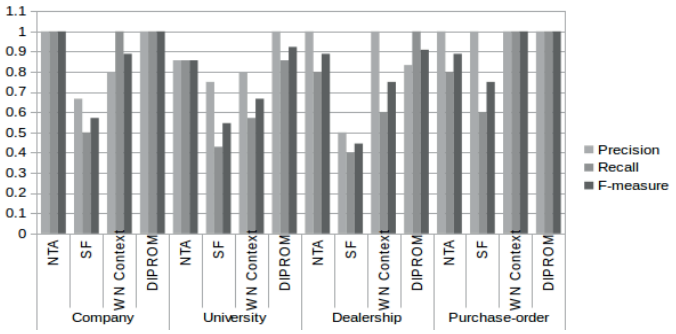

Fig. 10 Detailed accuracy comparison of composite matchers

5 Conclusion and future works

In this paper, we presented a novel schema matcher. Like many other schema matchers, this one is a hybrid schema matcher with three components: a linguistic, a vocabular and a structural matcher. The linguistic matcher is a refined syntactical label comparator evaluating common substring lengths through a logistic activation function. The vocabular matcher evaluates the similarity of the related terms sets also taking into account the term frequencies. This characteristic boosted the already promising capabilities of related term set similarity evaluation. The proposed structural matcher systematically exploits the vicinity relatedness. It is based on the new concepts called neighbor-levels and contribution function. The neighbor-level based structural matcher can also be tuned by defining the number of the considered neighbor-levels and the contribution function itself.

Related terms set may be a key schema matching input as it is, but unfortunately it is rarely provided to schema descriptions. Hence we presented a process that extracts related term set from available schema descriptions. As an additional output of this process, related term frequency information is also gained from the descriptions, which was used to further improve the related term set similarity evaluation in our approach.

We have thoroughly tested our solution and compared it with other known schema matchers from the literature. Test scenarios were selected so that they contain relevant schema anomalies to set real challenges for the schema matchers. The attained average f-measure of our proposed schema matcher was 0.96, which also means 0.182 average accuracy improvement compared to other tested schema matchers with an average 0.08 standard deviation reduction among test cases.

We are currently investigating other structural matching approaches which systematically evaluate information given by the schema graph. Also, the related terms set similarity deserves potential further enhancements in the future.

References

[1] Bernstein, P. A., Madhavan, J., Rahm, E. "Generic schema matching, ten years later." *Proceedings of the VLDB Endowment*. 4(11), pp. 695-701. 2011.

[2] Linthicum, D. S. "Enterprise application integration." Addison-Wesley Professional. 2000.

[3] Rahm, E, Bernstein, P. A. "A survey of approaches to automatic schema matching." *The VLDB Journal*. 10(4), pp. 334-350. 2011. https://doi.org/10.1007/s007780100057

[4] Rahm, E. "Towards large-scale schema and ontology matching." In: *Schema matching and mapping*. Springer Berlin Heidelberg. pp. 3-27. 2011. https://doi.org/10.1007/978-3-642-16518-4_1

[5] Lee, Y., Sayyadian, M., Doan, A., Rosenthal, A. S. "eTuner: tuning schema matching software using synthetic scenarios." *The VLDB Journal*. 16(1), pp. 97-122. 2007. https://doi.org/10.1007/s00778-006-0024-z

[6] Gal, A., Sagi, T. "Tuning the ensemble selection process of schema matchers." *Information Systems*. 35(8), pp. 845-859. 2010. https://doi.org/10.1016/j.is.2010.04.003

[7] Gal, A. "Uncertain schema matching." In: *Synthesis Lectures on Data Management*. 3(1) pp. 1-97. 2011. https://doi.org/10.2200/S00337ED1V01Y201102DTM013

[8] Gal, A. "Managing uncertainty in schema matching with top-k schema mappings." In: *Journal on Data Semantics VI*. Springer Berlin Heidelberg. pp. 90-114. 2006. https://doi.org/10.1007/11803034_5

[9] Chen, Y.-P. P., Promparmote, S., Maire, F. "MDSM: Microarray database schema matching using the Hungarian method." *Information Sciences*. 176(19), pp. 2771-2790. 2006. https://doi.org/10.1016/j.ins.2005.11.015

[10] Villanyi, B., Martinek, P., Szikora, B. "A framework for schema matcher composition." *WSEAS Transactions on Computers*. 9(10), pp. 1235-1244. 2010.

[11] Melnik, S., Garcia-Molina, H., Rahm, E. "Similarity flooding: A versatile graph matching algorithm and its application to schema matching." In: Proceedings of the 18th IEEE International Conference on Data Engineering, San Jose, CA, USA, 26 Feb.-1 March 2002, pp. 117-128. https://doi.org/10.1109/icde.2002.994702

[12] Martinek, P., Szikora, B. "Detecting semantically related concepts in a SOA integration scenario." *Periodica Polytechnica Electrical Engineering*. 52(1-2), pp. 117-125. 2009. https://doi.org/10.3311/pp.ee.2008-1-2.14

[13] Boukottaya, A., Vanoirbeek, C. "Schema matching for transforming structured documents." In: Proceedings of the 2005 ACM symposium on Document engineering. pp. 101-110, 2005. https://doi.org/10.1145/1096601.1096629

[14] Madhavan, J., Bernstein, P. A., Rahm, E. "Generic schema matching with cupid." *VLDB*. 1, pp. 49-58. 2001.

[15] Miller, G. A. "WordNet: A Lexical Database for English." *Communications of the ACM*. 38(11), pp. 39-41. 1995. https://doi.org/10.1145/219717.219748

[16] Bonifati, A., Mecca, G., Papotti, P., Velegrakis, Y. "Discovery and correctness of schema mapping transformations." In: *Schema matching and mapping*. Springer Berlin Heidelberg, pp. 111-147. 2011. https://doi.org/10.1007/978-3-642-16518-4_5

[17] Bellahsene, Z., Duchateau, F. "Tuning for schema matching." In: *Schema Matching and Mapping*. Springer Berlin Heidelberg. pp. 293-316. 2011. https://doi.org/10.1007/978-3-642-16518-4_10

[18] Villanyi, B., Martinek, P. "Weighted related terms set comparison in semantic integration scenarios." In: Proceedings of the XXVI. microCAD International Scientific Conference. pp. 256-262. 2012.

[19] Gupta, V., Gurpreet, S. L. "A survey of text mining techniques and applications." *Journal of Emerging Technologies in Web Intelligence*. 1(1), pp. 60-76. 2009. https://doi.org/10.4304/jetwi.1.1.60-76

[20] Hofmann, M, Klinkenberg, R. (eds.) "RapidMiner: Data mining use cases and business analytics applications." *CRC Press*. 2013.

[21] Villanyi, B., Martinek, P. "Towards a novel approach of structural schema matching." In: Proccedings of the 13th IEEE International Symposium on Computational Intelligence and Informatics (CINTI). Budapest, Nov. 20-22, 2012 pp. 103-107. https://doi.org/10.1109/cinti.2012.6496741

[22] Rowell, M. "OAGIS: a canonical business language." *XML Journal*. 3(09), 2002.

[23] ONE, Commerce. Inc.: xCBL Specification. 2002. [Online]. Available from: http://www.xcbl.org/xcbl35/documentation.shtml [Accessed: 30th December 2015]