Managing Uncertainty in Schema Matching with Top-K Schema Mappings

Avigdor Gal

Technion – Israel Institute of Technology
Technion City, Haifa 32000, Israel

Abstract. In this paper, we propose to extend current practice in schema matching with the simultaneous use of top-K schema mappings rather than a single best mapping. This is a natural extension of existing methods (which can be considered to fall into the top-1 category), taking into account the imprecision inherent in the schema matching process. The essence of this method is the simultaneous generation and examination of K best schema mappings to identify useful mappings. The paper discusses efficient methods for generating top-K methods and propose a generic methodology for the simultaneous utilization of top-K mappings. We also propose a concrete heuristic that aims at improving precision at the cost of recall. We have tested the heuristic on real as well as synthetic data and anlyze the emricial results.
The novelty of this paper lies in the robust extension of existing methods for schema matching, one that can gracefully accommodate less-than-perfect scenarios in which the exact mapping cannot be identified in a single iteration. Our proposal represents a step forward in achieving fully automated schema matching, which is currently semi-automated at best.

1 Introduction

Matching concepts describing the meaning of data in heterogeneous distributed data sources (*e.g.*, HTML form tags and database and XML schemata) is one of the basic operations of data integration. Due to the cognitive complexity of this matching process [8], it has traditionally been performed by human experts (Web designers, database analysts, and even lay users, depending on the context of the application) [32, 20]. As data integration has been made more automated, the ambiguity in concept interpretation, also known as *semantic heterogeneity*, has become one of the main obstacles to this process. For obvious reasons, manual concept reconciliation in dynamic environments (with or without computer-aided tools) is inefficient to the point of being infeasible, and so cannot provide a general solution. Introduction of the Semantic Web vision [4] and the shift towards machine-understandable Web resources have underscored the importance of automatic matching between sets of elements, also known as *schema matching*.

As a result, several tools for automated schema matching, such as GLUE [11] and OntoBuilder [15], have been developed in recent years. Given two data schemata (*e.g.*, two sets of attributes), these tools output a single *mapping* from elements of one schema to elements of the other. The outputted mapping is considered to be the *best* of all possible mappings between these schemata.

Although these tools comprise a significant step towards fulfilling the vision of automated schema matching, it has become obvious that the user must accept a degree of imperfection in this process [14]. A prime reason for this is the enormous ambiguity and heterogeneity of data description concepts: It is unrealistic to expect a single mapping engine to identify the correct mapping for any possible concept in a set. Another (and probably no less crucial) reason is that "the syntactic representation of schemas and data do not completely convey the semantics of different databases" [26]; *i.e.*, the description of a concept in a schema can be semantically misleading. Therefore, managing uncertainty in schema matching has been recognized as the next issue on the research agenda in the realm of data integration [24].

In this work, we offer an uncertainty management tool, using top-K mappings. We propose to extend current practice in schema matching by using top-K schema mappings rather than a single best mapping. This is a natural extension of existing methods (which can be considered to fall into the top-1 category), taking into account the uncertainty described above. The essence of this method is the **simultaneous** generation of K schema mappings and the use of heuristics on them to improve the matching process. We demonstrate our approach using a heuristic, dubbed *stability analysis*, to analyze top-K mappings. The usefulness of the heuristic is demonstrated through an empirical analysis of real-world schemata as well as synthetic data.

1.1 Motivating example

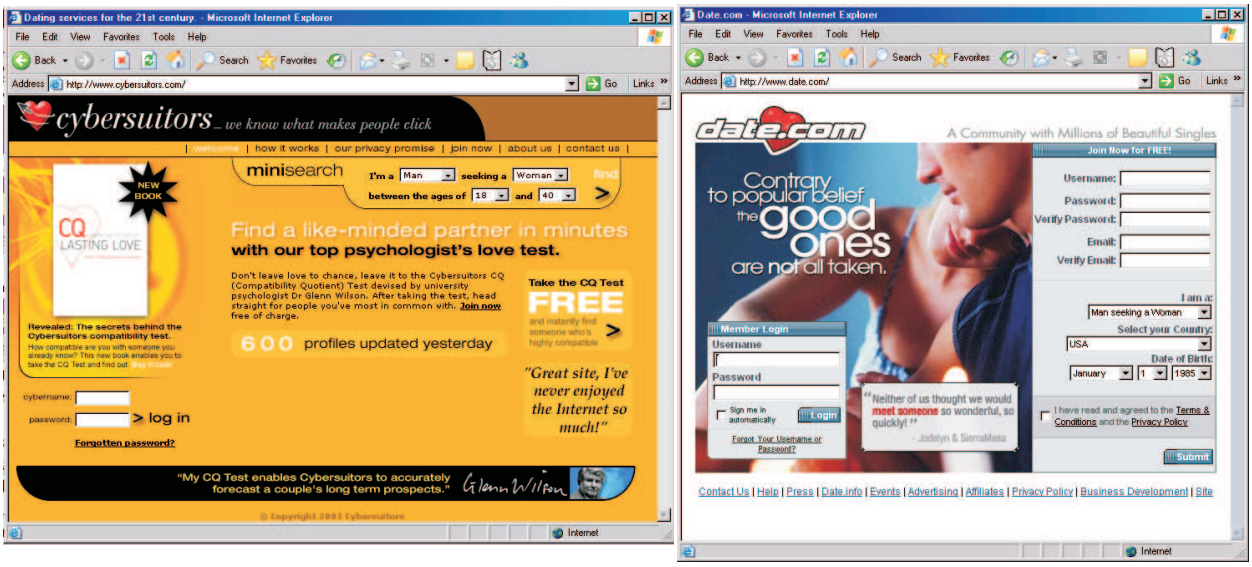

Fig. 1. Motivating example

Figure 1 presents two Web sites that offer matchmaking services. In each of these sites, one has to fill in personal information (*e.g.*, name, country of residence, and birthdate). We have applied a schema matcher called *Combined*, which is part of the toolkit of OntoBuilder [15]. The matcher returned the best mapping, containing a set of possible attribute mappings. We shall present a formal model of the matching process in Section 2.

www.cybersuitors.com	www.date.com
select: Country: (cboCountries)	select: Select your Country (countrycode)
select: Birthday: (cboDays)	select: Date of Birth (dob_day)
select: Birthday: (cboMonths)	select: Date of Birth (dob_month)
select: Birthday: (cboYears)	select: Date of Birth (dob_year)
checkbox: (chkAgreement2)	image: ()
checkbox: (chkAgreement1)	checkbox: Date.com - Join Now for Free! (over18)
select: State (if in USA): (cboUSstates)	select: I am a (i_am)

Table 1. Best mapping of the motivating example

A list of these mappings appear in Table 1. Each column in the table contains information about one field in a registration form in one of the Web sites. The information consists of the type of field (*e.g.*, select field and checkbox), the label as appears at the Web site, and the name of the field, given here in parentheses and hidden from the user. Each row in the table represents an attribute mapping, as proposed by the matcher. The top part of the table contains four correct mappings. The bottom part of the table contains three incorrect mappings.

Schema matchers face two obstacles in providing the best mapping. First, correct mappings should be identified and provided to the user. Second, incorrect mappings should be avoided. These two tasks can be measured by the classical IR metrics of recall and precision. The former judges how many correct mappings the matcher identifies, while the latter measures how many incorrect mappings the matcher has managed to avoid. Separating correct from incorrect mappings is a hard task. One technique that is often used is that of a threshold. Using a threshold, a matcher can discard attribute mappings that do not reach sufficient similarity, assuming that those attribute mappings with low similarity measures are less adequate than those with high similarity measure. By doing so, a schema matcher (hopefully) increases precision, at the expense of recall. Using a threshold, however, works only in clear-cut scenarios. Moreover, tuning the threshold becomes an art in itself. As an example, consider the case study in Figure 1 and Table 1. The four correct attribute mappings received similarity measures in the range $[0.49, 0.7]$ while the other similarity measures ranged from 0 to 0.5. Any arbitrary apriori selection of a threshold may yield false negatives (if the threshold is set above 0.49) or false positives, in case the threshold is set below 0.49.

Consider now an alternative, in which the matcher generates top-10 mappings, that is, the best 10 mappings between the two schemata, such that mapping i differs from mappings $1, 2, ...i-1$ by at least one attribute mapping. For example, the second best mapping maps `checkbox: (chkAgreement2)` with `checkbox: Date.com - Join Now for Free! (over18)` and `checkbox: (chkAgreement1)` is mapped with `image: ()` (this last attribute is actually a button and has no associated label or field name). The method proposed in this paper assumes that such a scenario represents a "shaky" confidence in this mapping to start with and removes it from the set of proposed attribute mappings (see Section 3.2). Simultaneous analysis of the top-10 mappings reveals that the four correct attribute mappings did not change throughout the 10 mappings, while the other attributes were mapped with different attributes in different mappings. Stability analysis, the heuristic proposed in this paper, suggests that the four mappings, for which consistent attribute mappings were observed in the top-10 mappings, should be proposed as the "best mapping," yielding a precision of 100% without adversely affecting recall.

1.2 Related work

Schema matching has been an active field of study for many years now. In this section, we review past research in two areas, heterogeneous databases and ontology design.

Heterogeneous databases The evolution of organizational computing, from "islands of automation" to enterprise-level systems, has created the need to homogenize heterogeneous databases. More than ever before, companies are seeking integrated data that go well beyond a single organizational unit. In addition, a high percentage of organizational data is now supplied by external resources (*e.g.*, the Web and extranets). Data integration is thus becoming increasingly important for decision support in enterprises [5]. This development also implies that databases with heterogeneous schemata increasingly face the prospect that their data integration process will not effectively manage semantic differences. This may result, at least to some degree, in the mismatching of concepts. Hence, methods for schema matching should take into account a certain level of uncertainty.

Many matchers have been proposed over the past two decades, by researchers in both academia and industry (*e.g.*, [33, 6, 13, 10, 28, 24]) to increase automation of the matching process and reduce semantic

mismatch problems. A useful classification of the various solutions can be found in [31]. A few other systems (MOMIS [3] and Clio [27], to name a couple) aim at resolving semantic heterogeneity in heterogeneous databases using manual intervention. For example, MOMIS input includes manual specification of concept semantic meaning and context as a prerequisite to the matching process.

Several systems offer facilities for iteratively scanning the search space, which can be considered an iterative variation of top-K. Clio presents the user with the best mapping and revises it if the user rejects the mapping. LSD [10] exploits domain constraints to produce the best mapping. A user then examines each concept mapping in the best mapping. If a user specifies a concept mapping as incorrect, it is fed to LSD as an additional constraint, and LSD then produces the next best mapping. Other matchers, such as similarity flooding [25] can be easily adapted to provide such top-K facilities. What is common to all these works is the manual involvement in the iterative process. We aim, in this work, at reducing manual involvement. Therefore, we suggest a heuristic for automatic analysis of information that can be generated from simultaneous (rather than iterative) analysis of top-K mappings.

Another variation of top-K exists [25, 19], in which the user is presented with top-K ($K = 3$ as a default in [25]) concept level mappings. That is, for **each concept** the user is presented with the best top-K concept mappings, out of which it can choose the one that fits best its needs. While top-K here is simultaneous, there are two main differences with our work. First, it ignores concept inter-relationships, and burdens the user with enforcing matching constraints (such as cardinality constraints). Secondly, there is no automatic reasoning involved in deciding which of the top-K is preferred.

There is sparse academic literature on the imprecision of automatic schema matching. A study of representations and reasoning about mappings between domain models was presented in [24]. The paper provides a model representation and inference analysis. It recognizes managing uncertainty as the next step on the research agenda in this area, and leaves this issue open for future research. The research described in [14] fills this gap by providing a model that represents uncertainty (as a measure of imprecision) in the matching process outcome. In the current paper, we build on the results of [14] in extending the current "best mapping" approach into one that considers top-K mappings as an uncertainty management tool.

Ontology design The second body of literature concerned with schema matching is that of ontology design. Ontologies have been widely accepted as the model of choice for modeling heterogeneous data sources by various communities, including the areas of databases [11, 21, 15] and knowledge representation [13], to name just two.

The realm of information science has produced an extensive body of literature and practice in ontology construction, using tools such as thesauri, and in terminology rationalization and matching of different ontologies (*e.g.*, [1, 35]). Elsewhere, as in the DOGMA project [21, 34], an engineering approach to ontology management is taken. Finally, researchers in the area of knowledge representation have studied ontology interoperability, resulting in systems such as Protégé [13].

The body of research aiming at matching schemata by using ontologies has traditionally focused on interactive methods requiring human intervention, massive at times. However, the vision of the semantic Web makes it necessary to minimize human intervention, replacing it with measures of syntactic similarity designed to approximate semantic matching. Recent papers (*e.g.*, [11, 15]) have explored the idea of automatic semantic reconciliation using ontologies. It was observed previously that automatic matching is imperfect [26]. Our approach, focusing on top-K mappings rather than the best mapping, handles this imperfection well.

The QOM (Quick Ontology Mapping) approach [12] is the closest we are aware of to our proposed framework. In this work, an iterative matching process is proposed, in which the mapping of a previous iteration is utilized in determining (the equivalence of our top-K) ontology elements to consider in the next

iteration (note the difference of this approach from other iterative approaches, *e.g.*, similarity flooding, in which there is no selectivity between iterations). The decision is either manual (with the support of a user) or by using a threshold. As already discussed, both methods have disadvantages. In this work, we generalize this approach. One other limitation of QOM is that its top-K is computed per concept, as in [25, 19], ignoring overall ontology constraints (such as $1:1$ mapping).

1.3 Contributions and outline

The specific contributions of this paper are as follows:

- We formalize the notion of top-K schema mappings within the framework of schema matching.
- We provide a classification of top-K matchers according to cardinality constraints.
- We demonstrate that there exists a correlation between patterns in top-K mappings and the correctness of the mapping, which makes the case for the need for top-K mapping analysis.
- We present a heuristic that makes use of simultaneous top-K mappings to improve mapping precision.
- We show experimental results to substantiate the usefulness of top-K mappings in improving precision and evaluate the trade-offs of applying methods based on top-K mappings.

The rest of the paper is organized as follows. Section 2 presents a model for schema matching as a basis for the formal introduction of top-K mappings. Next, we provide a generic heuristic for using top-K mappings (verification) and instantiate it into a concrete heuristic (Section 3). Section 4 outlines our experiments with the verification heuristic. We conclude in Section 5.

2 The model

This section introduces formally the concept of top-K mappings in the context of schema matching. A model for schema matching is presented in Section 2.1, followed by the modeling of top-K in Section 2.2.

2.1 A model for schema matching

As a basis for this work we next layout a model for schema matching and explicitly specify the set of assumptions we shall use throughout the paper. Let S_1 and S_2 be two schemata, defined using some data model (*e.g.*, relational or ontological), with n_1 and n_2 attributes, respectively. Attributes can be joint into *elements*, sets of attributes. The process of schema matching yields schema mapping(s), in which elements of S_1 are mapped onto elements of S_2.

Generally speaking, the process of schema matching is performed in two steps [9]. First, a degree of similarity is computed **automatically** for all element pairs (one element from each schema in each pair), using such methods as name matching, domain matching, and structure (such as XML hierarchical representation) matching. Recall that an element may consist of more than a single attribute. For illustration purposes, consider Table 3 (Section 2.2) to be described in details later. Each entry in the table represents a degree of similarity of a single element pair. The degree of similarity is typically defined on a $[0, 1]$ scale, where 0 represents no similarity and 1 represents fully similar elements.

As a second step, a single mapping is chosen to be the *best mapping*. The best mapping is a mapping that optimizes some target function F, subject to matching constraints. For example, many schema matching

tools aim at maximizing the sum (or average) of pair-wise weights of the selected elements. When deciding on a best mapping, a matcher should decide which elements from one schema are to be mapped with elements of another schema. Also, the matcher may decide that some elements do not satisfy some matching constraints (*e.g.*, minimal degree of similarity) and cannot be mapped. For further illustration, consider Table 3 once more. The bold-face entries in the table (jointly) represent the best mapping.

COMA [9], OntoBuilder, Cupid, and other schema matching tools apply variations of this model in their matching process. Others (such as Prompt and similarity flooding) also apply this two step methodology, yet do not support a mode that provides the user with all pairwise element mappings. However, it can be expected that making their internal representation of attribute similarity measures available for generating top-K mappings is feasible.

A convenient data structure for modeling the matching problem is to view it as an undirected bipartite graph, $G = (X, Y, E)$, with a node set $V = X \bigcup Y$ representing elements, where X and Y denote the sides of the graph (each side representing one schema), and an edge set E. Weights $w : E \rightarrow \mathrm{R}^+$ are assigned with edges, representing the degree of similarity between elements. G does not have to be a complete graph. Threshold constraints may result in the elimination of some edges. Also, some matchers (*e.g.*, similarity flooding) present only partial pairwise similarity measures. Again, such constraints are interpreted as an incomplete graph.

A mapping in G is a subset of pair-wise edges of E. We denote a mapping by $M \subseteq E$. $F(M)$ represents the target function value of M. Each edge $e \in M$ is an *element mapping*.

Using the proposed data structure, the matching problem becomes a problem of selecting an optimal mapping (*i.e.*, a subset of E that optimizes F). Given a matching problem and a set of matching constraints, we denote by A_{best} the best known algorithm for solving the bipartite graph matching problem, given the matching constraints. We denote by $C\left(A_{best}\right)$ the complexity of A_{best}.

Typical classification of matching constraints partitions matching problems into either $1:1$ matching, $1:n$ matching, $n:1$ matching, and general $(n:m)$ matching [31]. We now discuss three special cases within this classification and the methods for solving the matching problem in these cases, using an undirected bipartite graph as the underlying data structure. Henceforth, we shall assume that $F(M) = \sum_{e \in M} k_e w\left(e\right)$, a weighted average where k_e is a parameter that can represent the relative importance of an edge e. The study of top-K using other target functions is left for future research.

$1:1$ **matching** When constraining the mapping to be $1:1$, the node set represents individual attributes, where the X node set contains all attributes of one schema ($|X| = n_1$) and the Y node set contains all attributes of the other schema ($|Y| = n_2$). A mapping in G is a subset of pair-wise **disjoint** edges of E. An efficient algorithm for identifying the best mapping in this case is given as a variation of the weighted bipartite graph matching problem [16]. Such an algorithm has a complexity of $C\left(A_{best}\right) = O\left(n^3\right)$ [23],[1] where $n = \max(n_1, n_2)$.

Before moving on to the next two cases, we would like to offer a refined categorization of matching constraints. Consider, for example, a $1:n$ constraint. Such a constraint may indicate that a single attribute in one schema can be replicated to more than a single attribute in another schema (*e.g.*, a `Password` attribute in one schema vs. `Type Password` and `Retype Password` in another schema). Alternatively, such a constraint may indicate that an attribute in one schema is decomposed into several attributes in another schema (*e.g.*, `Name` in one schema is decomposed into `Given Name` and `Surname` in another schema). We denote the former a *replication* constraint and the latter a *decomposition* constraint.

[1] Here we consider the complexity of the best *sequential* algorithm for finding a maximum weight mapping in a bipartite graph. Likewise, an alternative algorithm for this problem is presented in [17], and its time complexity is $O(n^{2.5} \log\left(nW\right))$, where W stands for the highest edge weight in the graph.

When a replication constraint is applied, matching decisions of individual attributes are independent of one another. Therefore, matching `Password` with `Type Password` is independent of the matching of `Password` with `Retype Password`. However, a decomposition constraint cannot be evaluated by some aggregation of matching of individual attributes. For example, consider the attribute `Name` and the attribute pair `Given Name` and `Surname`. Machine learning techniques are likely to rate the comparison of concatenation of values from `Given Name` and `Surname` against `Name`, higher than comparing each of the attributes independently. Therefore, decomposition constraints require the evaluation of elements as well as individual attributes.

The bipartite graph can support the provision of such comparison by adding feasible attribute sets as elements (nodes) in the graph. Such enhancement entails, in many cases, higher complexity of the matching process. For example, if the matching constraint allows a single attribute in S_1 to be matched with up to 2 attributes in S_2, one needs to consider all possible pairs in a schema. Therefore, n_1 elements of S_1 are matched with $\binom{n_2}{2} = \frac{1}{2} n_2 (n_2 - 1)$ elements of S_2, which increases the number of nodes in G to $|V| = \mathcal{O}\left(n^2\right)$. This computation can be generalized to any (sufficiently small) constant c, constraining the number of attributes in an element. The complexity in this case is of $\mathcal{O}(n^c)$.

$1:n$ **matching with replications** Using the bipartite graph as a data structure, the following simple algorithm can be devised. Consider a matching constraint that allows an attribute in one schema to be replicated several times in another schema. Therefore, nodes on one side of the bipartite graph (say, the Y nodes) cannot have more than a single incidenting edge. Such a constraint does not apply to the other side of the graph (X nodes). Therefore, all one has to do is to identify the best edge incidenting upon each node that requires unique mapping. Let $v \in Y$ be a node in the graph and v_e be the set of all edges incident on v. The following simple algorithm can thus be applied, where $argmax_{v_e} w(e)$ stands for the edge of v_e that maximizes w.

Algorithm 1:
$S_e \leftarrow \emptyset$

For each $v \in Y$ do

$S_e \leftarrow S_e \cup \{argmax_{v_e} w(e)\}$

Return S_e

The complexity of Algorithm 1 is $C(A_{best}) = \mathcal{O}(|E|) = \mathcal{O}\left(\left(n^c\right)^2\right) = \mathcal{O}\left(n^{2c}\right)$.

$n:m$ **matching with decomposition** When replacing single attributes with elements, traditional algorithms for solving matching problems in bipartite graphs can no longer ensure unique attribute selection. Therefore, two elements that contain the same attribute A can be chosen as part of a mapping, which means that A no longer has a unique mapping. Nevertheless, certain $n:m$ constraints can be supported by the bipartite graph data structure. As an example, consider a constraint enforcing each attribute in one schema to be mapped uniquely to a single combination of attributes. Therefore, if `Name` in S_1 is mapped to the combination of `Given Name` and `Surname` in S_2, no other attribute in S_1 can be mapped to this combination (although a different combination of `Surname` and `OfficeNumber` in S_2 may be the appropriate combination for `InitialPassword` in S_1). This special case can fall into the category of $1:1$ global cardinality and $n:m$ local cardinality, according to the classification of [31].
Such a constraint is mapped into the data structure in the following way. The X set of nodes represent individual attributes of one schema. The Y set of nodes represent all legal elements of the other schema. We can then apply an algorithm for solving the weighted bipartite graph matching problem in this graph.

The complexity of this algorithm can be defined in terms of attributes as followed.

$$C\left(A_{best}\right) = \mathcal{O}\left(|V|^3\right) = \mathcal{O}\left((n^c)^3\right) = \mathcal{O}\left(n^{3c}\right)$$

2.2 Modeling top-K mappings

Let $G = (X, Y, E)$ be an undirected bipartite graph with edges representing the degree of similarity between elements. Top-K can be defined recursively as follows. For $K = 1$, the K-th best mapping M_1^* is any maximum weight mapping in G. Let M_i^* denote the i-th best mapping, for any $i > 0$. Then, given the best $i-1$ mappings $M_1^*, M_2^*, \ldots, M_{i-1}^*$, the i-th best mapping M_i^* is defined as a mapping of maximum weight over mappings that differ from each of $M_1^*, M_2^*, \ldots, M_{i-1}^*$. Therefore, given top-$K$ mappings, any mapping $M \subseteq E$ such that $M \notin \{M_1^*, M_2^*, \ldots, M_K^*\}$ satisfies $F(M) \leq \min_{1 \leq j \leq k} F(M_j^*) = F(M_k^*)$.

cybersuitors.com	date.com
101. select: Country: (cboCountries)	201. select: Select your Country (countrycode)
102. select: Birthday: (cboYears)	202. select: Date of Birth (dob_year)
103. select: Birthday: (cboMonths)	203. select: Date of Birth (dob_month)
104. .select: Birthday: (cboDays)	204. select: Date of Birth (dob_day)
105. checkbox: (chkAgreement1)	
106. checkbox: (chkAgreement2)	
107. text: Last name: (txtLastName)	
108. text: First name: (txtFirstName)	
109. text: Use this name instead of my first name: (cboPenName)	
110. text: Your city town village: (txtPlace)	
111. select: State (if in USA): (cboUSstates)	
112. password: Please choose a password This ... (txtPassword)	
113. password: Re-enter password: (txtPassword2)	
114. text: Your email address: (txtEmail)	
115. text: Please retype your email address to confirm it: (txtEmail2)	
	205. image: ()
	206. checkbox: Date.com - Join Now for Free! (over18)
	207. select: I am a (i_am)

Table 2. running example attributes

To illustrate the notion of top-K mappings, consider the following example.

Example 1 (Running example schemata). Table 2 is an extension of Table 1, presenting attributes of two schemata, `cybersuitors.com` and `date.com`. Matching attributes are presented in the same row.

Attribute names were extracted from the labels as appear on the Web site and the field names (in parentheses) of the form entries. Field names are used for matching the values returned by the client to the server's database schema, hence the condensed form. Attribute names are preceded by field type (*e.g.*, select, checkbox, *etc.*), also used to enhance the matching process. The field `image: ()` is an image field, used as a submit button, that is not associated with name or label. We refer the interested reader to [15] for a detailed description of the extraction process. Henceforth, we number vertices rather than using attribute names, for the sake of clarity.

↓ cybersuitors.com/ date.com→	201	202	203	204	205	206	207
101	**0.5937**	0.0597	0.0639	0.0597	0	0.0597	0.0542
102	0.0575	**0.4944**	0.184	0.184	0	0.0618	0.0625
103	0.0620	0.1833	**0.6847**	0.1833	0	0.0611	0.0618
104	0.0620	0.2516	0.2507	**0.7042**	0	0.0627	0.0634
105	0.05774	0.0538	0.0577	0.0538	**0.4**	0.5038	0.0538
106	0.0577	0.0538	0.0577	0.0538	0.4	**0.5038**	0.0538
107	0.0075	0.0101	0.0101	0.0101	0	0.0473	0.0216
108	0.0071	0.0153	0.0153	0.0153	0	0.0469	0.0194
109	0.0085	0.009	0.009	0.009	0	0.034	0.014
110	0.051	0.0089	0.0089	0.0089	0	0.0115	0.0089
111	0.0649	0.0712	0.0712	0.0712	0	0.0629	**0.1101**
112	0.013	0.0052	0.0052	0.0052	0	0.0124	0.0124
113	0.01	0.0108	0.0108	0.0108	0	0.0125	0.0075
114	0.0563	0.0094	0.0094	0.0094	0	0.0118	0.0094
115	0.0599	0.0083	0.0083	0.0083	0	0.0083	0.0319

Table 3. Edge weights in the example

The edge weights of the bipartite graph of the example are given in Table 3, as computed **automatically** by OntoBuilder. We refer the interested reader to [15] for a detailed description of OntoBuilder matching techniques. In this example, we constrain the matching process to result in $1:1$ mappings.

The exact mapping (as determined by a human observer) is: $\{e_{101,201}, e_{102,202}, e_{103,203}, e_{104,204}\}$, while the outcome of A_{best} for the bipartite graph is the best mapping
$M^* = \{e_{101,201}, e_{102,202}, e_{103,203}, e_{104,204}, e_{105,205}, e_{106,206}, e_{111,207}\}$. M^* contains the exact mapping, with additional three attribute mappings. As discussed in Section 1.1, the exact mapping cannot be found by setting a threshold. □

To motivate our research, we now offer an intuitive interpretation of top-K mappings. Suppose an edge weight represents the belief of a matcher in the correctness of an element mapping. A higher weight indicates a higher confidence in the element mapping correctness. When switching from the i-th best mapping to the $(i+1)$ best mapping, the matcher is forced to give up at least one element mapping, while maintaining an overall high confidence in a schema mapping. To do so, the matcher cedes an element mapping in which it is less confident. Therefore, generating top-K mappings can be observed as a process in which a matcher iteratively abandons element mappings in which it is less confident.

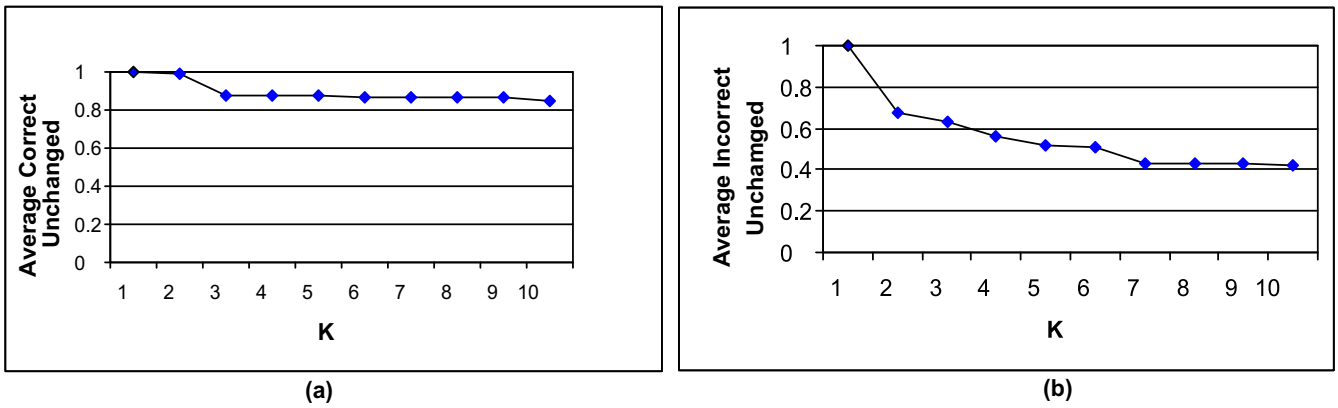

Fig. 2. Average unchaged attribute mappings

We substantiate this intuitive interpretation with an empirical analysis, based on experiments with real world data. The details of the experiments are provided in Section 4, together with a thorough empirical analysis. Here, we provide an initial motivation to our work. Figure 2 provides an analysis of the attribute mapping stability. K is given at the x axis. For each K, we measure the percentage of correct (incorrect) attribute mappings that were not changed throughout the K mappings. That is, those attribute pairs that appear in all top-K mappings. Figure 2(a) illustrates the results for correct attribute mappings. Figure 2(b) illustrates the results for incorrect mappings. It is easy to observe that correct attribute mappings were less subject to change (less than 16%) with K then incorrect attribute mappings (dropping 60%). Therefore, by analyzing top-K mappings, it is likely that correct attribute mappings will remain stable, while incorrect attribute mappings will keep on changing.

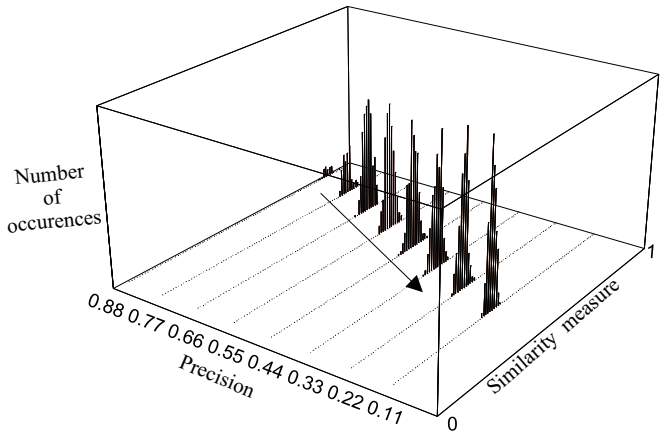

Fig. 3. Similarity measures distribution according to imprecision levels

It can be argued that top-K mappings are not different from any K mappings. Therefore, by choosing **any** K mappings (and not necessarily the top ones), one can derive useful information on the quality of the schema matching process. We justify the decision to use top-K mappings in an earlier research [14]. A class of schema matchers (termed *monotonic*) was defined in [14], for which a higher similarity measure is an indication of a more precise mapping. For completeness sake, we now provide an illustrative example of one form of monotonicity, dubbed *statistical monotonicity* in [14]. Figure 3 presents a pictorial illustration of a distribution of similarity measure values of all possible mappings between two schemata, according to precision levels. These results are based on similarity measure, as assigned by the *Combined* matcher, already mentioned in Section 1.1. At each precision level, similarity measures seem to be normally distributed with a decreasing mean. Therefore, the higher the precision of a mapping is, the higher would be the similarity measure assigned by the matcher. In the absence of any variance, one would expect the top-1 mapping to be the exact mapping. However, as shown in [14] and illustrated here, the variance in each precision level allows mappings within any given precision level to overlap in their similarity measure with other precision levels. Therefore, monotonicity ensures that the top-K mappings are sufficiently "close" to the exact mapping.

The analysis in [14] shows that due to the uncertainty inherent in the matching process, no matcher can be expected to identify the exact mapping as the best mapping at all times. If a matcher were required to iterate over all possible permutations, the search would become infeasible. However, if the top-K mappings contain sufficient information to predict most of the correct mappings and K can be determined to be sufficiently small, precision may be increased at a negligible cost.

Based on the observations of this section, and the empirical analysis we have conducted, a reasonable approach involves a simultaneous analysis of possible mappings to determine the best mapping. This approach

is in contrast to current practice, in which a system seeks the best mapping and resorts to manual intervention for additional input whenever using the best mapping fails. The most prominent drawback of the current approach has to do with the reliance on manual intervention. We aim at minimizing such intervention, and therefore we propose to enhance automatic reasoning on top-K mappings.

3 Schema matching verification

To overcome the uncertainty in mapping results, we propose the following generic methodology. Let S_1 and S_2 be two schemata. The methodology contains five steps as follows:

1. **Computing:** $G = (X, Y, E)$ =Generate similarity graph.
2. **Matching:** $\{M_1^*, M_2^*, \ldots, M_K^*\}$ =Generate top-K mappings, using G.
3. **Verifying:** $A = \{(a, a')\}$ =Analyze $\{M_1^*, M_2^*, \ldots, M_K^*\}$.
4. **Recomputing:** $G' = (X, Y, E')$ =compute $w(a, a')$ for each $(a, a') \in A$.
5. **Rematching:** M^* =Generate top-1 mapping, using G'.

According to this generic methodology, a similarity graph is generated and top-K mappings are generated and analyzed to identify element pairs that are worthy of further consideration (*i.e.*, verification). It then recomputes the similarity measures and generates the best mapping.

Three main differences exist between the proposed methodology and existing practice in schema matching. First, most of current methods do not use top-K mappings. For those who use top-K mappings (*e.g.*, [25, 12]), it is done locally, on an attribute-based cases. Therefore, it can capture only local similarities, while ignoring global constraints, such as cardinality constraints, and global phenomenon, such as accumulation of elements from one schema around a single element from another schema. The reasoning behind local top-K mappings brings us to the second difference. Current methods assume the assistance of a user in the process, by refuting certain mappings, or setting new constraints. Clearly, cognitive limitations of users make it much harder to compare whole schemata, while attribute-based comparison is a much easier task. In contrast, our approach is aimed at fully-automatic schema matching, recognizing the inherent uncertainty in such a process. With humans outside the loop, there is no reason to simplify the task at the expanse of accuracy, whenever it can be avoided. Finally, the verification step analyzes all top-K mappings simultaneously, rather the iteratively, as was proposed in approaches such as Clio [27]. This difference can also be attributed to the presence of a user in the loop in current practice. In user presence, iterations provide an opportunity for incremental improvement, a process that fits human cognitive capabilities. With automatic matching, on the other hand, no feedback is given, and the use of simultaneous analysis can provide insights that are not necessarily evident with pairwise comparison. One final comment has to do with machine learning methods for schema matching. Such methods allow a training period on annotated data, to be followed by matching schemata, unknown beforehand. Feedback is only given during the learning phase. Therefore, the proposed approach can serve as a complementary method once the learning phase is completed.

The first step is beyond the scope of this paper. As discussed in Section 1.2, many worthy methods and heuristics have been proposed in generating G, and any that satisfy the monotonicity condition, as set in [14], would suffice. The last step has also been extensively discussed in the literature. In Section 2.1 we have discussed the use of bipartite graphs in representing the schema matching problem, and solving the top-1 problem.

In the rest of this paper we focus on steps 3-4. For completeness sake, we next introduce an analysis of existing algorithms for step 2, finding top-2 mappings and top-K mappings.

3.1 Algorithms for finding top-K mappings

The assignment ranking problem involves the enumeration of K assignments with least cost. The first algorithm of $\mathcal{O}(K\,|V|^4)$ for ranking assignments was suggested by Murty in 1968 [29], where $|V|$ is the number of nodes in the assignment graph. In 1985/6, Hamachar and Queyranne proposed an alternative general algorithm for ranking solutions of combinatorial problems [18]. This algorithm was later specialized for bipartite matchings [7], in $\mathcal{O}(K\,|V|^3)$, using flow networks. In [30], another $\mathcal{O}(K\,|V|^3)$ algorithm was presented, using a specific order of analyzing assignments.

In [7] it was shown that finding the second best assignment is equivalent to finding the shortest cycle in a residual network relatively to the best assignment. This demands solving at most n shortest path problems, and therefore solving the top-2 problem is of $\mathcal{O}(|V|^3)$ complexity. Therefore, the top-2 problem can be solved for the case of $1:1$ mapping in $\mathcal{O}(n^3)$. The case of $m:n$ with decomposition is also reduced to $1:1$ mapping (see Section 2.1), yielding a solution to the top-2 problem in $\mathcal{O}(n^{3c})$.

For the case of $1:n$ with replication, an efficient algorithm for generating top-K mappings can be devised as follows, summarized in Algorithm 2 as a pseudo-code. Let $G=(X,Y,E)$ be the matching bipartite graph. First, for each node $v\in Y$, a sorted list of all edges $\{(u,v)\,|u\in X\}$ in a decreasing order of similarity is generated (Line 2). In the next step, we compute for each node $v\in Y$ the weight difference between the edge (u,v) with maximum similarity and the next edge in the list (Line 3). These values are then inserted into a minimum heap (Line 4). Lines 5-7 generate the best mapping. At an iteration i, we remove an edge (u,v) from M^*_{i-1} such that its weight difference is minimal (Line 9) and replace it by an edge (u',v), which precedes it in the sorted list of v (lines 10 and 11). The weight difference between (u',v) and the next edge in the sorted list of v is then inserted into the heap (Line 12).

Algorithm 2:

1 For each $v\in Y$ do

2 $\quad \mathsf{v}^{sorted}$ =create a sorted list of $\{u\in X\}$ in a decreasing order of $w(u,v)$

3 $\quad \Delta v_{\max}=w(u_{(1)},v)-w(u_{(2)},v)$

4 H =Build-Min-Heap$\left(\left\{\left(u_{(1)},v\right),\Delta v_{\max}\,|v\in Y\right\}\right)$

5 $M^*_1=\left\{\left(\max\left\{v^{sorted}\right\},v\right)|v\in Y\right\}$

6 For each $v\in Y$ do

7 $\quad \mathsf{v}^{sorted}=\mathsf{v}^{sorted}\backslash\max\left\{v^{sorted}\right\}$

8 For $i=2$ to k do

9 $\quad (u_{(l)},v)$,Δv=Heap-Extract-Min(H)

10 $\quad M^*_i=M^*_{i-1}\backslash\left\{(u_{(1)},v)\right\}\cup\left(\max\left\{v^{sorted}\right\},v\right)$

11 $\quad \mathsf{v}^{sorted}=\mathsf{v}^{sorted}\backslash\max\left\{v^{sorted}\right\}$

12 $\quad$ Min-Heap-Insert$\left(H,(u_{(l+1)},v),w(u_{(l+1)},v)-w(u_{(l+2)},v)\right)$

13 Return $\{M^*_i\}_{i=1}^k$

Sorting the edge weights for each node takes $\mathcal{O}(n \lg n)$. Therefore, the overall sorting takes $\mathcal{O}\left(n^2 \lg n\right)$. Building the heap is $\mathcal{O}(n)$ and the generation of the best mapping takes $\mathcal{O}(n)$. Each additional mapping requires $\mathcal{O}(\lg n)$, due to the heap operations. Therefore, the overall complexity is $\mathcal{O}\left(n^2 \lg n + K \lg n\right)$.

An adaptation of Murty's algorithm for top-K and Chegireddy's and Hamacher's algorithm for top-2 to the schema matching world is detailed in [2]. We have implemented the algorithms and embedded them in OntoBuilder [15], which was used for experimenting with the heuristic proposed in this paper.

3.2 Stability analysis

We are now ready to introduce a concrete heuristic, instantiating the generic verification methodology presented earlier. Let S_1 and S_2 be two schemata and let $G = (X, Y, E)$ be a bipartite graph, modeling the matching alternatives between S_1 and S_2. Given a set $\{M_1^*, M_2^*, \ldots, M_K^*\}$ of K top mappings, and a user threshold $t \in [0, 1]$, the analysis step of the stability analysis heuristic first computes for each edge $e \in E$ the number of times it appears in $\{M_1^*, M_2^*, \ldots, M_K^*\}$, dubbed $\iota(e)$. It is worth noting that $0 \leq \iota(e) \leq K$. Then, it generates a set of edges $A = \{(a, a')\}$ such that $(a, a') \in A$ iff $\frac{\iota(e)}{K} \leq t$. That is, the set A contains all edges that do not appear a sufficient number of times in the top-K mappings, $\{M_1^*, M_2^*, \ldots, M_K^*\}$. The recomputation phase revises G by setting $w(a, a') = 0$ for each $(a, a') \in A$.

www.cybersuitors.com	www.date.com	$\frac{\iota(e)}{K}$
101	201	1.0
102	202	1.0
103	203	1.0
104	204	1.0
111	207	0.6
105	205	0.5
106	205	0.5
107	206	0.4
108	206	0.4
115	207	0.4
109	206	0.2

Table 4. Stability analysis of the motivating example

Example 2 (Stability analysis example). Consider Example 1. Table 4 provides $\frac{\iota(e)}{K}$ values of all edges whose count in the top-10 mappings is non-zero, in a decreasing order. For $t > 0.6$, the only non-zero edges will be those of the exact mapping. □

For the stability analysis to work, a schema matcher should be monotonic [14]. Therefore, our underlying assumption is that stable attribute mappings represent those mappings that are part of the exact mapping. Our empirical analysis show that this heuristic indeed works better for a matcher which was shown to be monotonic in [14].

4 Experiments

We now present an empirical evaluation of stability analysis, to support our hypothesis that simultaneous analysis of top-K mappings improves the quality of mapping. We report in details on our experimental setup

(Section 4.1), the data that was used (Section 4.2), and the evaluation methodology (Section 4.3). We then present in Section 4.4 the experiment results and provide an empirical analysis of these results.

4.1 Experiment setup

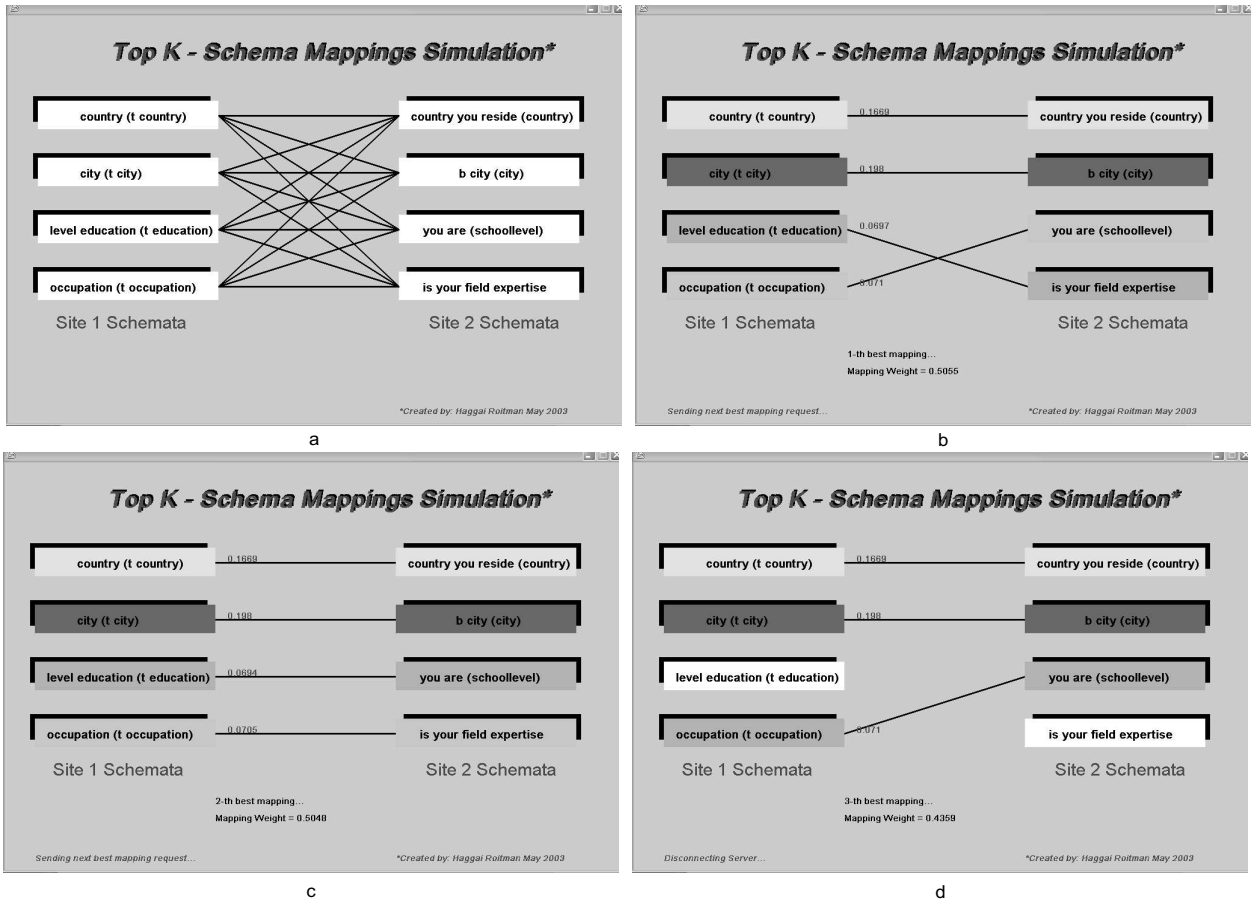

Fig. 4. Demo snapshots

We have implemented several top-K algorithms (including Murty's, Chegireddy and Hamacher's, and our version of $1:n$ mappings with replication) as part of the development of OntoBuilder. OntoBuilder runs under the Java 2 JDK version 1.4 or greater and is downloadable from http://ie.technion.ac.il/OntoBuilder. We chose for A_{best} (whenever $1:1$ mapping is applicable) the maximum weighted bipartite algorithm implementation suggested in LEDA ([22], pp. 132-150). The A_{best} algorithm was implemented in Java, and plugged in for use within our top-K algorithms. We have also generated a demo presentation showing execution of the top-K mappings. Figure 4 provides the visual output of the algorithm, with $K = 3$.

OntoBuilder specializes in extracting ontologies from Web forms, a feature we have used in our experiments. OntoBuilder accepts two ontologies as input, a candidate ontology and a target ontology. It attempts to match each attribute in the target ontology with an attribute in the candidate ontology. OntoBuilder supports an array of matching and filtering algorithms and can be used as a framework for developing new schema matchers which can be plugged-in and used via GUI or as an API. In our experiments we have used the following four matchers (detailed description of which can be found in [15]):

Term: A term is a combination of a label and a name. Term matching compares labels and names to identify syntactically similar terms. To achieve better performance, terms are preprocessed using several techniques originating in IR research. Term matching is based on either complete word or string comparison.

Value: Value matching utilizes domain constraints (*e.g.*, drop lists, check boxes, and radio buttons) to compute similarity measure among terms. The availability of constrained value-sets becomes valuable when comparing two terms that do not exactly match through their labels.

Composition: A composite term is composed of other terms (either atomic or composite). Composition can be translated into a hierarchy. This schema matcher assigns similarity to terms, based on the similarity of their neighbors.

Precedence: The precedence relationship is unique to OntoBuilder and therefore worth of a lengthier discussion. In any interactive process, the order in which data are provided may be important. In particular, data given at an earlier stage may restrict the availability of options for a later entry. For example, a car rental site may determine which car groups are available for a given session, using the information given regarding the pick-up location and time. Therefore, once those entries are filled in, the information is sent back to the server and the next form is brought up. Such precedence relationships can usually be identified by the activation of a script, such as (but not limited to) the one associated with a SUBMIT button. Precedence can be translated into a precedence graph. The matching algorithm is based on a technique we dub *graph pivoting*, as follows. When matching two terms, we consider each of them to be a pivot within its own ontology, thus partitioning the graph into semantically related subgraphs. The semantics of pivoting is taken from the ontological analysis, and in the case of precedence the graph it partitioned into a subgraph of all preceding terms and all succeding terms. By comparing preceding subgraphs and succeding subgraphs, we determine the confidence strength of the pivot terms.

4.2 Data

For our experiments with stability analysis, we have selected 86 Web forms from different domains, such as dating and matchmaking, job hunting, Web mail, hotel reservation, news, and cosmetics. We extracted each Web form ontology using OntoBuilder. We have matched the Web forms in pairs, where pairs were taken from the same domain. The ontologies vary in size, from 5 to 64 attributes with about half of the ontologies have between 10 and 20 attributes. They also vary in the proportion of number of attribute pairs in the exact mapping relative to the target ontology. This proportion ranges from 16.6% to 94.7%; the proportion in about half of the ontologies is more than 60%. Another dimension is the size difference between matched ontologies, ranging from equal size ontologies to about 3 times difference between ontologies. In about half of the pairs, the difference was less than 30% of the target ontology size. Finally, the best mapping precision results range from 5% to 100%, with about half the ontology pairs were mapped by both matchers (see below) with precision of more than 40%.

In addition to the real data, we generated 50 synthetic ontology pairs, as follows. We have selected the ontology pair taut.securesites.com and www1522.boca15-verio.com. Both ontologies are of medium size (18 and 19 attributes, respectively), strongly similar (the exact mapping contains 18 attributes), and close in size. The best mapping of the *Combined* matcher is the exact mapping. Based on the similarity matrix of these ontologies we have generated 50 similarity matrices, for which each value val is replaced with $val + val * f$ and f is randomly taken from $[-v, v]$. $v \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ (10 matrices for each v value).

We ran two schema matchers, namely *Term* and *Combined*, to generate the top-10 mappings. The *Combined* matcher aggregates the results of the four matchers, detailed in Section 4.1, using weighted average. We ran a total of 1860 experiments (93 ontology pairs, 2 schema matchers, and 10 best mappings). Note that in our experiments, generating top-1, top-2, *etc.* can be performed in one pass.

4.3 Evaluation methodology

In order to evaluate the stability analysis heuristic, we measured its performance using two main metrics, namely precision and recall. Precision is computed as the ratio of correct element mappings, with respect to some exact mapping, out of the total number of element mappings suggested by a heuristic. Recall is computed as the ratio of correct element mappings, out of the total number of element mappings in the exact mapping. Both recall and precision are measured on a $[0, 1]$ scale. An optimal schema matching results in both precision and recall equal to 1. Lower precision means more false positives, while lower recall suggests more false negatives.

The independent variables of the experiments were K, the number of simultaneous mappings, and t, the threshold.

4.4 Results and analysis

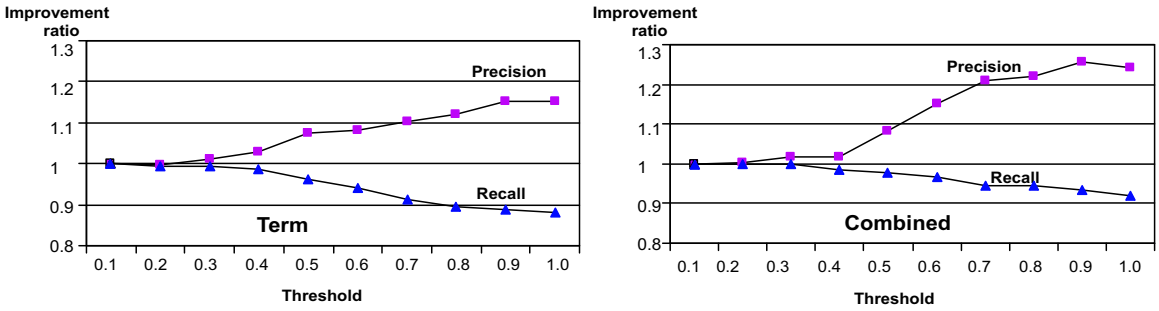

Fig. 5. Precision and Recall for Stability analysis with $K = 10$

In our first experiment we have measured precision and recall for a fixed $K \in \{1, 2, ..., 10\}$, varying the threshold t. Figure 5 presents the average change to precision and recall for different thresholds over all 43 real data pairs. K was set to 10. Figure 5(left) illustrates the results for the *Term* matcher and Figure 5(right) illustrates the results for the *Combined* matcher. In both cases, precision increases (in general) up to $t = 0.9$ with the increased threshold. Recall demonstrates a monotonic decrease with the increased threshold. Such a phenomenon accords with our initial intuition and is expected for monotonic matchers. In [14] we have shown that both the *Term* and the *Combined* matchers are monotonic.

A closer look at the amount of improvements reveals that the precision of the *Term* matcher increases by up to 15.4% (with $t = 0.9$). The *Combined* matcher provides an increase of 25.6% (again with $t = 0.9$). As for recall, it decreases to a maximum of 11.9% for the *Term* matcher and by a smaller 8% for the *Combined* matcher (for $t = 1$). Therefore, stability analysis works better for the *Combined* matcher than for the *Term* matcher. This conclusion can be aligned with the discussion in [14], where the exact mapping was found, on average, at $K = 7$ for the *Combined* matcher and for $K > 100$ for the *Term* matcher.

Looking at the shape of the graphs, it seems that with both matchers, the improvements (in terms of precision) levels off at about $t = 0.9$. We hypothesize that, in general, the increased demand of a higher threshold benefit the heuristic up to a point, from which it will become impossible for the top-K algorithm to keep even its stronger attribute mappings. We believe that the "break-even point" depends to a great extent on the size of the ontology, some evidence to which is given below. The *Term* matcher has a more wiggly precision result, with some decreases in precision for $t = 0.1, 0.6$, and 0.7. Therefore, the performance

Ontology class	Term				Combined			
	precision		recall		precision		recall	
	max increase	t	max decrease	t	max increase	t	max decrease	t
strongly similar	19.8%	1.0	10.1%	1.0	28.5%	1.0	5.8%	1.0
weekly similar	12.1%	0.9	13.7%	1.0	23.3%	0.9	10.3%	1.0
big	7.6%	1.0	8.5%	1.0	16.7%	1.0	3.6%	1.0
small	20.7%	1.0	14.3%	1.0	29.5%	1.0	11.1%	1.0
similar	8.9%	0.9	11.3%	1.0	19.9%	0.9	7%	1.0
disimilar	24.1%	1.0	12.6%	1.0	32.3	0.9	9%	1.0
low initial precision	11.4%	0.9	16.9%	1.0	27.6%	0.9	11.6%	1.0
high initial precision	19.2%	1.0	7.1%	1.0	24.5%	1.0	4.7%	1.0

Table 5. Stability analysis of ontology classes

of the *Term* matcher is less predicted (although the difference is not statistically significant) than that of the *Combined* matcher. This conclusions was also reached in [14].

Next, we have partitioned the ontologies into two groups, based on ontology similarity. We define an ontology pair for which 60% or more of the terms in the target ontology can be matched to terms in the target ontology to be *strongly similar*. 22 pairs out of the 43 pairs were strongly similar, with similarity ranging from 60% to 94.7%. Table 5 summarizes the analysis for this and following partitions. Again, K was set to 10. The results show improvement over the whole group in the precision level, with smaller decrease in the recall level. The precision of the *Term* matcher increases by up to 19.8% (with $t = 1$). The *Combined* matcher provides an increase of 28.5% (again with $t = 1$). As for recall, it decreases by 10.1% for the *Term* matcher and by a smaller 5.8% for the *Combined* matcher. The main conclusion for this experiment is that stability analysis works better for strongly similar ontologies.

We also partitioned the ontologies based on size. We define an ontology to be big (relative to ontology sizes we have in our data set) if it has more than 20 attributes. There were 18 big target ontologies. The results show less improvement in the precision level for bigger ontologies, yet with smaller decrease in the recall level. The precision of the *Term* matcher increases by up to 7.6% (with $t = 1$). The *Combined* matcher provides an increase of 16.7% (again with $t = 1$). As for recall, it decreases by 8.5% for the *Term* matcher and by 3.6% for the *Combined* matcher. The smaller gain in precision and the smaller reduction in recall for bigger ontologies can be justified by the smaller marginal impact a single attribute has on the overall performance. Generally speaking, however, it seems that stability analysis is better suited for smaller ontologies.

We next experimented with ontologies that differ in size. Ontology pairs were considered similar if the difference between the number of attributes of the candidate and target ontologies was less then 30% of the target ontology (there were 23 such pairs). The results show less improvement in the precision level for similar-size ontologies. The precision of the *Term* matcher increases by up to 8.9% (with $t = 0.9$). The *Combined* matcher provides an increase of 19.9% (again with $t = 0.9$). As for recall, it decreases by 11.3% for the *Term* matcher and by 7% for the *Combined* matcher. We found no clear explanation to the phenomenon in which stability analysis is better suited for ontologies that differ in size.

The last partitioning is based on the best mapping precision level. Out of the 43 pairs, 21 pairs (20 pairs for the *Combined* schema matcher) had a precision level of less than 0.4 for the best mapping. Here, our analysis show different results for the *Term* matcher and the *Combined* matcher. The stability analysis heuristic using the *Term* matcher seem to be more effective for pairs which had high initial precision, increasing the precision by an average of 19.2%. Using the *Combined* matcher, the difference between the two groups is much smaller, showing slightly better performance (27.6% vs. 24.5%) for the ontology pairs with low initial precision. Recall, for both matchers, was significantly less affected by for ontologies with high initial precision.

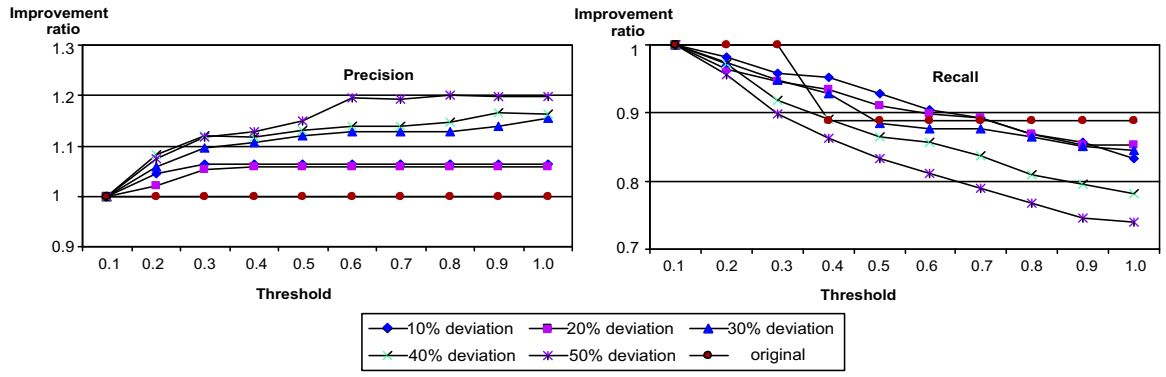

Fig. 6. Stability analysis on synthetic data: Combined algorithm

Our final experiment was aimed at analyzing the impact of noise in schema matcher similarity measures. We have generated top-K mappings for the synthetic matrices (see Section 4.2), applied the stability analysis heuristic and measured precision and recall as before. Figure 6 illustrates the change of precision and recall, partitioned according to the various v (maximum deviation) values. We present the results for the *Combined* matcher only since the results for the *Term* matcher share the same trends. The synthetic data demonstrates the same trends as the real data, increase of precision as the threshold increases, leveling off at around $t = 0.9$. Somewhat surprising, the heuristic becomes more effective, in terms of precision, as v increases. Clearly, the good starting point of the original mapping serves in the good performance, even with increased noise. When we applied the heuristic to randomly generated matrices (in which each attribute pair got a uniformly distributed value in $[0, 1]$) the result was a complete chaos, and the heuristic was practically useless. Recall deterioration (Figure 6(right)) serves as an indication to the impact of noise. With more noise, the heuristic throws out more attribute mappings, including good ones. Therefore, with $v = 0.5$, the matcher lost 26% in its recall level for $t = 10$ (compare this with an average of 12% recall loss for real data).

5 Discussion and conclusion

In this paper, we have investigated a major shortcoming of standard methods for automatic semantic reconciliation: namely, that they commit to the best mapping, typically chosen as that which maximizes the sum (or average) of pair-wise similarities under certain constraints (*e.g.*, 1:1 mapping). The problem is that, due to uncertainty in concept interpretation, the best mapping chosen by the matcher can actually be an unsuccessful choice. To alleviate this shortcoming, we propose that instead of using just the best mapping, a set of top-K mappings should be generated and examined iteratively until a good mapping is found. Using this approach, the exact mapping is likely to be identified if the matcher ranks it sufficiently high (but not necessarily as the best).

We have proposed a generic framework for the simultaneous utilization of top-K mapping and provide a concrete heuristic, termed stability analysis, to utilize top-K mappings in improving mapping precision (at the cost of recall). Stability analysis was shown empirically to provide good results for monotonic schema matchers.

An ongoing research involves the investigation of efficient methods for utilizing input from multiple schema matchers. Choosing among the current variety of schema matchers is far from being trivial. First, the number of heuristics is continuously growing, and this diversity by itself complicates our decision making. Second, as one would expect, recent empirical analysis shows that there is no (and may never be) a single dominant schema matcher that performs best, regardless of the data model and application domain [14]. Bearing these observations in mind, we believe that customers of schema matching would expect some degree

of robustness, despite the biases and shortcomings of individual heuristics. Therefore, tools for determining the best "cocktail" of schema matchers would seem to be the next natural step in schema matching research. Future work involves the identification of additional heuristics for top-K utilization. Such heuristics may be based on the separation of matchers into groups. For example, matchers can be partitioned into costly vs. cheap matchers. Cheap matchers are utilized in identifying potential points of failures and more costly matchers will be applied to this problem subset, thus reducing the overall cost of the matching process (hopefully) without hurting its accuracy.

Acknowledgement

The work of Gal was partially supported by Technion V.P.R. Fund - New York Metropolitan Research Fund, The Fund for the Promotion of Research at the Technion, and the IBM Faculty Award for 2003/2004 on "Self-Configuration in Autonomic Computing using Knowledge Management." We thank Haggai Roitman for his assistance in implementing the algorithms and Amir Taller for his assistance in running experiments.

References

1. J. Aitchison, A. Gilchrist, and D. Bawden. *Thesaurus construction and use: a practical manual*. Aslib, London, third edition, 1997.
2. A. Anaby-Tavor. Enhancing the formal similarity based matching model. Master's thesis, Technion-Israel Institute of Technology, May 2003.
3. S. Bergamaschi, S. Castano, M. Vincini, and D. Beneventano. Semantic integration of heterogeneous information sources. *Data & Knowledge Engineering*, 36(3), 2001.
4. T. Berners-Lee, J. Hendler, and O. Lassila. The semantic Web. *Scientific American*, May 2001.
5. M. Brodie. The grand challenge in information technology and the illusion of validity. Keynote lecture at the International Federated Conference 'On the Move to Meaningful Internet Systems and Ubiquitous Computing', 2002.
6. S. Castano, V. De Antonellis, M.G. Fugini, and B. Pernici. Conceptual schema analysis: Techniques and applications. *ACM Transactions on Database Systems (TODS)*, 23(3):286–332, 1998.
7. C.R. Chegireddy and H.W. Hamacher. Algorithms for finding k-best perfect matchings. *Discrete Applied Mathematics*, 18:155–165, 1987.
8. B. Convent. Unsolvable problems related to the view integration approach. In *Proceedings of the International Conference on Database Theory (ICDT)*, Rome, Italy, September 1986. In *Computer Science*, Vol. 243, G. Goos and J. Hartmanis, Eds. Springer-Verlag, New York, pp. 141-156.
9. H.H. Do and E. Rahm. COMA - a system for flexible combination of schema matching approaches. In *Proceedings of the International conference on very Large Data Bases (VLDB)*, pages 610–621, 2002.
10. A. Doan, P. Domingos, and A.Y. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In Walid G. Aref, editor, *Proceedings of the ACM-SIGMOD conference on Management of Data (SIGMOD)*, Santa Barbara, California, May 2001. ACM Press.
11. A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In *Proceedings of the eleventh international conference on World Wide Web*, pages 662–673. ACM Press, 2002.
12. M. Ehrig and S. Staab. Qom quick ontology mapping. In *Proceedings of the Third International Semantic Web Conference (ISWC'2004)*, pages 683 – 697, October 2004. Lecture Notes in Computer Science, Volume 3298.
13. N. Fridman Noy and M.A. Musen. PROMPT: Algorithm and tool for automated ontology merging and alignment. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, pages 450–455, Austin, TX, 2000.
14. A. Gal, A. Anaby-Tavor, A. Trombetta, and D. Montesi. A framework for modeling and evaluating automatic semantic reconciliation. *VLDB Journal*, 14(1):50–67, 2005.
15. A. Gal, G. Modica, H.M. Jamil, and A. Eyal. Automatic ontology matching using application semantics. *AI Magazine*, 26(1), 2005.
16. Z. Galil. Efficient algorithms for finding maximum matching in graphs. *ACM Computing Surveys*, 18(1):23–38, March 1986.

17. U. Güntzer, W-T. Balke, and W. Kießling. Optimizing multi-feature queries in image databases. In *Proceedings of the Twenty Sixth Very Large Databases (VLDB) Conference*, pages 419–428, Las Vegas, 2001.
18. H.W. Hamacher and M. Queyranne. K-best solutions to combinatorial optimization problems. *Annals of Operations Research*, 4:123–143, 1985/6.
19. A. Heß and N. Kushmerick. Learning to attach semantic metadata to web services. In *Proceedings of the Second Semantic Web Conference*, 2003.
20. R. Hull. Managing semantic heterogeneity in databases: A theoretical perspective. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 51–61. ACM Press, 1997.
21. M. Jarrar and R. Meersman. Formal ontology engineering in the DOGMA approach. In *Proceedings International Federated Conference 'On the Move to Meaningful Internet Systems and Ubiquitous Computing'*, pages 1238–1254, October 2002.
22. K.Mehlhorn and S.Naher, editors. *LEDA, A platform for combinatorial and geometric computing*. Cambridge University Press, 1999.
23. B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer, second edition, 2002.
24. J. Madhavan, P.A. Bernstein, P. Domingos, and A.Y. Halevy. Representing and reasoning about mappings between domain models. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI)*, pages 80–86, 2002.
25. S. Melnik, E. Rahm, and P.A. Bernstein. Rondo: A programming platform for generic model management. In *Proceedings of the ACM-SIGMOD conference on Management of Data (SIGMOD)*, pages 193–204, San Diego, California, 2003. ACM Press.
26. R.J. Miller, L.M. Haas, and M.A. Hernández. Schema mapping as query discovery. In A. El Abbadi, M.L. Brodie, S. Chakravarthy, U. Dayal, N. Kamel, G. Schlageter, and K.-Y. Whang, editors, *Proceedings of the International conference on very Large Data Bases (VLDB)*, pages 77–88. Morgan Kaufmann, 2000.
27. R.J. Miller, M.A. Hernàndez, L.M. Haas, L.-L. Yan, C.T.H. Ho, R. Fagin, and L. Popa. The Clio project: Managing heterogeneity. *SIGMOD Record*, 30(1):78–83, 2001.
28. G. Modica, A. Gal, and H. Jamil. The use of machine-generated ontologies in dynamic information seeking. In C. Batini, F. Giunchiglia, P. Giorgini, and M. Mecella, editors, *Cooperative Information Systems, 9th International Conference, CoopIS 2001, Trento, Italy, September 5-7, 2001, Proceedings*, volume 2172 of *Lecture Notes in Computer Science*, pages 433–448. Springer, 2001.
29. K.G. Murty. An algorithm for ranking all the assignments in order of increasing cost. *Operations Research*, 16:682–687, 1968.
30. M. Pascoal, M.E. Captivo, and J. Cl'imaco. A note on a new variant of Murty's ranking assignments algorithm. *4OR: Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 1(3):243–255, 2003.
31. E. Rahm and P.A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.
32. A. Sheth and J. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, 1990.
33. A.P. Sheth, S.K. Gala, and S.B. Navathe. On automatic reasoning for schema integration. *Intenational Journal on Intelligent Cooperative Information Systems (IJICIS)*, 2(1):23–50, June 1993.
34. P. Spyns, R. Meersman, and M. Jarrar. Data modelling versus ontology engineering. *ACM SIGMOD Record*, 31(4), 2002.
35. B.C. Vickery. *Faceted classification schemes*. Graduate School of Library Service, Rutgers, the State University, New Brunswick, N.J., 1966.