SDD-Matcher: a Semantic-Driven Data Matching Framework

Yan Tang¹ and Gang Zhao²

¹ VUB STARlab, 10G731, Vrije Universiteit Brussel
Pleinlaan 2, 1050 Elsene
Brussels, Belgium

² Interlartes, BP8 Waterloo, 1410 Belgium

¹ yan.tang @vub.ac.be, ² gang.zhao@intelartes.com

Abstract. A generic semantic-driven data matching framework (SDD-Matcher) has been designed and developed for matching data objects across organizations. It contains matching algorithms at three different levels: string, lexical and graph. The level of graph is also called ontological or conceptual level. Each SDD-Matcher matching strategy at least contains a graph matching algorithm. Besides, we can freely choose the matching algorithms at these three levels when composing a matching strategy. A matching strategy can as well be a composition of several matching strategies. In this article, we focus on a full engineering cycle of SDD-Matcher. It includes use cases, comparison to the related work, its design and formalization, an evaluation method, implementation, and evaluation results. We use real case data for the evaluation, which has been validated by large enterprises.

Keywords: semantic decision table (SDT), domain ontology, matching, ontology-based data matching, DOGMA, ODMF

I. INTRODUCTION

Data matching is the process of bringing data from different and heterogeneous data sources and comparing them in order to find out whether they represent the same or similar real-world object [13]. It is a key problem of data management processes, such as data or schema integration, querying across domains, data cleansing, data mining and fuzzy searching. The authors in [13, 16] provide a survey on data matching and mark the importance of data matching in the mentioned fields.

The types of the data sources concerning data matching can be (local or remote) databases, web data (e.g. web pages) or content data in natural language (e.g. textual documents). Although the data source types are different, the underlying principles are rather similar. That is, data matching happens at the levels of either schemas (or structures) or data instances (or values). The approaches discussed in [16] focus on performing data value analysis. The ones shown in [13] emphasize structure analysis. The scope of this paper is essentially to deal with structure analysis.

As will be explained in the next section, there is very little related work on semantic-driven data matching (or ontology-based data matching). There are quite a lot of existing work on ontology matching and integration¹. S-Match [18], Optima [14] and AgreementMaker [9] are some examples. The problem of semantic-driven data matching is not exactly the problem of ontology matching. The goal of ontology matching is to solve the problem of semantic inconsistency while integrating/merging more than two ontologies. Our goal (also the scope of this paper) is to find the

¹ http://www.ontologymatching.org/ (last retrieved on July 14th, 2011)
similarities between two data sets, each of which corresponds to one part in the ontology. There is only one ontology in the particular problem.

SDD-Matcher is designed based on an ontology-based data matching framework (ODMF), which was initially designed by VUB STARLab in 2007 during the EC Prolix project\(^2\). It has been gradually enriched and tested in the EC 3DAH project\(^3\), EC TAS3 project\(^4\) and the ITEA 2 DYSE project\(^5\).

In Prolix, we have used ODMF to calculate competency gaps between employee’s profiles and learning modules (e.g. learning materials and courses) in order to find most suitable learning modules for the employees \([38, 40]\). The ODMF use case in 3DAH is focused on how to evaluate medical students and provide personalized suggestions of online learning materials to them \([6]\). In TAS3, we have developed a method supported by tools to provide semantic support to process modellers during the design phase of a secure business process model. ODMF was used to discover the user design intent from a dedicated knowledge base \([7]\). In DYSE, we have embedded ODMF in a component discoverer and recommender system, which is used to assist amateurs when they want to create their own personalized smart environment (it is also called Do-It-Yourself activities). ODMF has been used to find the most suitable hardware (e.g. sensors and actuators) and software (e.g. pieces of code and CSS feeds) components according to the needs of end users \([41]\). These use cases cover the domains of eLearning, 3D anatomy, human resource management, security and ubiquitous computing. ODMF is not restricted to a particular domain.

Although a few of the SDD-Matcher matching strategies have been discussed in \([6, 7, 38, 40, 41]\), we feel it necessary to provide an integration of SDD-Matcher, which covers a full engineering cycle of a generic use case, design, implementation, evaluation method and evaluation results. This becomes the main contribution of this article. It is organized as follows. Sec. II is the related work. We will describe the overview of SDD-Matcher in Sec. III. A generic use case and a detailed use case from British Telecom will be illustrated in Sec. IV. We present the SDD-Matcher matching algorithms and a few strategy examples in Sec. V. The evaluation method that can be used to evaluate an SDD-Matcher matching strategy is recorded in Sec. VI. We show the SDD-Matcher implementation and a few evaluation results in Sec. VII. In Sec. VIII, we conclude with discussions and future work.

---

\(^2\) http://www.prolixproject.org (last retrieved: July 13\(^{th}\), 2011)

\(^3\) http://3dah.miralab.ch/ (last retrieved: July 13\(^{th}\), 2011)

\(^4\) http://www.tas3.eu (last retrieved: July 13\(^{th}\), 2011)

\(^5\) http://www.dyse.org (last retrieved: July 13\(^{th}\), 2011)
II. RELATED WORK

As mentioned earlier, the problem of semantic-driven data matching is not ontology matching. The problem of ontology matching occurs when more than one ontologies are involved and need to be integrated or merged. Hence the approaches to this problem mainly deal with checking/ensuring consistency between several ontologies. The problem of semantic-driven data matching is a data matching problem. There is only one ontology. The solutions are to find the connections and measurements between two data sets using an ontology.

Data matching happens at the levels of schemas or instances. SDD-Matcher deals with the matching at both the structure level and instance level. Approaches to data matching at the schema level and instance level can be found in the survey papers [13, 16] respectively. Compared to the related work, SDD-Matcher is a more comprehensive approach.

There exist very few approaches to semantic-driven data matching that make good uses of an ontology. TODE [35] uses ontologies (in particular, SUMO6 and WordNet [17]) to categorize web pages. It only deals with matching with very limited structure information. In particular, only the relations of “is-a” (Meronym) and “part-of” (Holonym) are used in TODE. Oracle RDBMS [10] embeds ontology-based semantic matching components in their relational database management system. Oracle RDBMS provides four ontology related operators – ONTRELATED, ONT_EXPAND, ONT_DISTANCE and ONT_PATH. They are implemented based on OWL relations, which are richer than the ones in TODE. However, the matching mechanism used by these database queries is one-to-one matching. The ontology-based resource matching approach illustrated in [44] shows how to share resources in a Grid environment. It uses domain ontologies to describe domain terminology and request queries. The matching rules are embedded in a policy ontology. Similar to many existing ontology-based searching approaches, this work is based on ontological queries, which supports one-to-one exact matching. Compared to their work, SDD-Matcher uses all kinds of ontological relations to process matching. The matching is not only one-to-one matching but also one-to-many matching.

We consider an ontology as a connected graph or network. Our problem then becomes how to find the connections between two sub-graphs. The ideas in [1, 22, 46] show the related work. Barrett et al. [1] illustrate an algorithm of finding the shortest path between two nodes in a labelled and weighted network. The authors in [22] discover data based on feature distances. The authors in [46] focus on how to find data objects or web pages that belong to the same/similar contexts. Although their work deals with sub-graphs in one graph, it is focused more on how to draw a boundary between searching spaces, especially as discussed in [22, 46]. Compared to their work, we care neither about the boundary between two sub-graphs, nor how they are graphically overlapped. Instead, we need to know how each element (arcs and vertices) from one sub-graph is linked with the others from the other one. In addition, our graph is specific. It is an ontology; hence the arcs and vertices are meaningful and semantically rich.

6 http://www.ontologyportal.org (last retrieved: July 18th, 2011)
SDD-Matcher contains algorithms at three levels: string (or alphabetical), lexical and graph. They are the basic components for each SDD-Matcher strategy, which contains at least one graph matching algorithm.

Figure 1 shows a generic semantic-driven data matching design for all SDD-Matcher matching strategies. The solid arrow-tipped bars indicate the execution flow of an SDD-Matcher matching strategy, the entry points of which are illustrated with double line arrows. The dotted arrow-tipped bars explain how final scores are reached.

An SDD-Matcher matching strategy may contain a string-based/morphology-based algorithm (at the level of string matching), a lexical matching algorithm (at the level of lexical matching) and a graph matching algorithm (at the level of graph/conceptual matching). The goal of matching at the levels of string and lexical is to form proper sub-graphs. We find the relations between two sub-graphs based on the matching analysis at the level of graph (or conceptual).

A matching process can start with the matching at the string level followed by the matching at the levels of lexical and graph. It can also start with the matching at the lexical level followed by the matching at the graph level. Or, it starts directly with the matching at the graph level.

An SDD-Matcher matching strategy that contains three steps at the three mentioned levels is described as follows. In each step, SDD-Matcher executes one matching algorithm, which generates a matching score. The scores from a string matching algorithm and a lexical matching algorithm need to be reworked with a penalty value. Or, they need to pass certain thresholds. In the last step, SDD-Matcher generates a final matching score using a graph matching algorithm.

We will discuss SDD-Matcher in detail in Sec. V. In the next section, we will illustrate a generic use case and a detailed use case.
A generic use case and a specific use case described in this section will be used to illustrate the SDD-Matcher strategies and evaluate them.

A. A Generic Use Case

SDD-Matcher is suitable for calculation of the similarities between two sub-graphs in a large ontology graph, given two organizations, which use the same ontology but have different types of data sets (or the documents in different formats). In order to use SDD-Matcher, we need to annotate the data sets using this ontology (or more generically speaking, a knowledge base of data semantics). The two sub-graphs are indeed the annotations of the data sets.

Figure 2 shows a generic use case of SDD-Matcher. Two data objects from two different sub-domains, enterprises or departments, have data descriptions in natural language. These descriptions can be in a well defined structure or free texts, which the annotation server takes as the input and generates two annotation sets as the output. These two annotation sets correspond to two sub-graphs in the ontology graph. Based on the ontology and the annotation set, SDD-Matcher calculates a similarity score, which is a float number in the range [0, 1].

Figure 2 – An SDD-Matcher Generic Case

Note that the ontology server supports not only querying or reasoning ontologies, but also ontology versioning. A meaning evolution support system (MESS, [11]) has been built at STARLab to support community-based ontology versioning.

We use the Developing Ontology-Grounded Methods and Applications framework (DOGMA, [25, 26, 34]) for modelling our ontologies. An ontology modelled in DOGMA has two layers: a lexon layer and a commitment layer.

A lexon is a simple binary fact type, which contains a context identifier, two terms and two roles. For instance, a lexon \((\gamma, \text{teacher}, \text{teaches}, \text{is taught by}, \text{student})\) presents a fact that “a teacher teaches a student, and a student is taught by a teacher”, where “teacher” and “students” are two terms, “teaches” and “is taught by” are two roles. The context identifier \(\gamma\), which we often use a document
A commitment (also called “ontological commitment”) is an agreement made by a community (also called “group of interests”). A commitment is a rule in a given syntax. For instance, we can use decision commitment language (DECOL, [37]) to write a commitment shown as follows.

\[ P = [\text{Student, takes, is taken by, Exam}] : \text{MAND} (P). \]

The above commitment contains a mandatory constraint, which means that each student takes at least one exam. Annotation is one kind of commitment, with which we select lexons from the ontology.

The generic use case shown in this section can be extended for particular domains. In the next section, we will illustrate a use case from British Telecom (the Amsterdam Branch) in the field of human resource management (HRM) and eLearning.

B. A Use Case in BT (the Amsterdam Branch)

This use case has been worked out during the EC Prolix project. The main idea follows the design of the generic use case in the previous section.

The human resource management (HRM) department of one test bed uses the textual descriptions of “company values” (such as “trustworthy”, “straightforward” and “heart”) to evaluate its employees. The results are recorded in the evaluation forms. The training department uses competency notations of skills and abilities to categorize the learning courses and learning materials. Each department uses its own supporting tools and terms. In order to have a better collaboration between these two departments and to enhance the interoperability between the applications across different departments, we have developed the HRM ontology, with which we annotate the company values and the learning courses/materials. By doing so, each company value or each learning material corresponds to one subgraph in the same ontology.

Figure 1 illustrates the use case by considering an ontology as a graph and an annotation result set as a sub-graph. The sub graph indicated with “Straightforward & Heart” is the union of the annotation sets of “Straightforward” and “Heart” (Table 1). The sub graph denoted with “Course ITIL1” is the annotation set of “ITIL1” (Table 1).

Note that in our problem settings, every item (company values and learning courses) is annotated with the domain ontology. If a knowledge engineer cannot find a properly defined concept, then he/she needs to define this new concept by executing an ontology creation/versioning method, e.g. (MESS, [11]).
Figure 3 - the company values (“straightforward” and “heart”) from the HRM department and a learning course (“ITIL1”) from the training/e-learning department correspond to two sub-graphs of the HRM ontology [40]

Table 1 – the annotation sets for “Straightforward”, “Heart” and “Course ITIL 1”

<table>
<thead>
<tr>
<th>Context</th>
<th>Head term</th>
<th>Role</th>
<th>Co-role</th>
<th>Tail term</th>
</tr>
</thead>
<tbody>
<tr>
<td>Straightforward</td>
<td>person</td>
<td>has characteristics</td>
<td>is characteristics of</td>
<td>simplicity</td>
</tr>
<tr>
<td></td>
<td>person</td>
<td>has characteristics</td>
<td>is characteristics of</td>
<td>clarity</td>
</tr>
<tr>
<td></td>
<td>employee</td>
<td>has characteristics</td>
<td>is characteristics of</td>
<td>simplicity</td>
</tr>
<tr>
<td></td>
<td>employee</td>
<td>has characteristics</td>
<td>is characteristics of</td>
<td>clarity</td>
</tr>
<tr>
<td>Heart</td>
<td>person</td>
<td>has characteristics</td>
<td>is characteristics of</td>
<td>simplicity</td>
</tr>
<tr>
<td></td>
<td>person</td>
<td>has characteristics</td>
<td>is characteristics of</td>
<td>clarity</td>
</tr>
<tr>
<td></td>
<td>employee</td>
<td>has characteristics</td>
<td>is characteristics of</td>
<td>simplicity</td>
</tr>
<tr>
<td></td>
<td>employee</td>
<td>has characteristics</td>
<td>is characteristics of</td>
<td>clarity</td>
</tr>
<tr>
<td></td>
<td>person</td>
<td>has characteristics</td>
<td>is characteristics of</td>
<td>simplicity</td>
</tr>
<tr>
<td></td>
<td>person</td>
<td>has characteristics</td>
<td>is characteristics of</td>
<td>clarity</td>
</tr>
<tr>
<td></td>
<td>employee</td>
<td>has characteristics</td>
<td>is characteristics of</td>
<td>simplicity</td>
</tr>
<tr>
<td></td>
<td>employee</td>
<td>has characteristics</td>
<td>is characteristics of</td>
<td>clarity</td>
</tr>
<tr>
<td></td>
<td>person</td>
<td>has characteristics</td>
<td>is characteristics of</td>
<td>simplicity</td>
</tr>
<tr>
<td></td>
<td>person</td>
<td>has characteristics</td>
<td>is characteristics of</td>
<td>clarity</td>
</tr>
<tr>
<td></td>
<td>employee</td>
<td>has characteristics</td>
<td>is characteristics of</td>
<td>simplicity</td>
</tr>
<tr>
<td></td>
<td>employee</td>
<td>has characteristics</td>
<td>is characteristics of</td>
<td>clarity</td>
</tr>
<tr>
<td>ITIL1</td>
<td>person</td>
<td>describe</td>
<td>is described by</td>
<td>practice</td>
</tr>
<tr>
<td></td>
<td>person</td>
<td>describe</td>
<td>is described by</td>
<td>service</td>
</tr>
<tr>
<td></td>
<td>person</td>
<td>define</td>
<td>is defined by</td>
<td>service</td>
</tr>
<tr>
<td></td>
<td>person</td>
<td>distinguish</td>
<td>is distinguished by</td>
<td>service</td>
</tr>
<tr>
<td></td>
<td>person</td>
<td>define</td>
<td>is defined by</td>
<td>function</td>
</tr>
<tr>
<td></td>
<td>person</td>
<td>define</td>
<td>is defined by</td>
<td>role</td>
</tr>
<tr>
<td></td>
<td>person</td>
<td>describe</td>
<td>is described by</td>
<td>process model</td>
</tr>
<tr>
<td></td>
<td>person</td>
<td>list</td>
<td>is listed by</td>
<td>characteristic of process</td>
</tr>
</tbody>
</table>
V. SDD-MATCHER MATCHING ALGORITHMS AND STRATEGIES

As mentioned in Sec. III, SDD-Matcher contains algorithms at three levels: string, lexical and graph. Algorithms concerning string and lexical matching are mainly used to form proper sub-graphs. And graph matching algorithms are used to find the relations between two sub-graphs.

The SDD-Matcher string matching algorithms include the ones from the SecondString project [8], which contains the implementation of UnsmoothedJS [20, 21, 48], JaroWinklerTFIDF [20, 21, 48] and TFIDF (term frequency–inverse document frequency, [33]).

The lexical matching algorithms are based on the lexical information from WordNet [17] and domain dictionary. In WordNet, synonyms are grouped into unordered sets – Synsets, the smallest entity in the linguistic database. Including the synonyms, mostly used relations are 1) is-a subsumption relationship (also called hypernym and hyponymy in WordNet, e.g. “jet black” is a hyponym of “black” and “achromatic color” is a hypernym of “black”); 2) type-instance relationship (e.g. “Mao Zedong” is an instance of “communist”); 3) part-whole Meronym relationship (e.g. “neck” is a part of “body”); 4) troponym in verb Synsets, which expresses different manner, precision or volume of a verb, for instance, “yawl” is a troponym of “shout”.

As discussed, we use lexons to express concepts and relations. The terms $r_1$ and $r_2$ in a lexon $\langle y, t_1, r_1, r_2, t_2 \rangle$ are often nouns. The roles $r_1$ and $r_2$ are often verbs. It is rare to see adverbs in a lexon. Therefore, the WordNet relations concerning adverbs are not currently used by SDD-Matcher. Note that the relation of antonym for nouns or verbs is not used as well. We consider two terms (or roles) separated (completely has no connection) when they do not connect with each other; and antonym is a specific case of this kind of separation.

Note that it is possible to use user-specific dictionary that contains data dictionaries and equivalences. It is simple at the computation yet important from business point of view.

Graph theory has been studied as a classic researching area for decades in the field of math. West [47] presents a good survey. Current SDD-Matcher takes commonly used graph matching algorithms, such as the ones of finding bipartite matching [47, chapter 3] and Dijkstra’s shortest path [12]. As will be discussed in the following sections, the LexMA matching algorithm is a simplified version of bipartite matching, and the one in the GRASIM strategy is based on the calculation of Dijkstra’s shortest path.

In the following subsections, four SDD-Matcher matching strategies will be illustrated.
A. Lexon Matching Algorithm/Strategy (LexMA)

LexMA is a graph matching algorithm, which is based on LeMaSt [39] and is the only algorithm used in the LexMA matching strategy in SDD-Matcher.

Suppose we have two graphs $G_1$, which contains $n_1$ lexons, and $G_2$, which contains $n_2$ lexons. We use $\cdot$ to indicate the source of a lexon. For example, $G_1 \cdot l_1$ is a lexon from $G_1$. Two lexons can have four different relations.

The first one is equivalence. We consider $l_1$ and $l_2$ equivalent and note it as “$l_1 = l_2$” if it can be deduced using the following formula.

$$(l_1 \cdot l_1 = l_2 \cdot l_1) \cap (l_1 \cdot r_1 = l_2 \cdot r_1) \cap (l_1 \cdot r_2 = l_2 \cdot r_2) \cap (l_1 \cdot t_2 = l_2 \cdot t_2) \cup (l_1 \cdot t_1 = l_2 \cdot t_1) \cap (l_1 \cdot r_2 = l_2 \cdot r_2) \cap (l_1 \cdot t_1 = l_2 \cdot t_2) \rightarrow l_1 = l_2.$$

The second one is inequality. We consider $l_1$ and $l_2$ unequal and note it as “$l_1 \neq l_2$” if it can be deduced using the following formula.

$$(l_1 \cdot t_1 \neq l_2 \cdot t_1) \cap (l_1 \cdot t_2 \neq l_2 \cdot t_1) \cap (l_1 \cdot t_1 \neq l_2 \cdot t_2) \cap (l_1 \cdot r_1 \neq l_2 \cdot r_1) \cap (l_1 \cdot r_2 \neq l_2 \cdot r_2) \cap (l_1 \cdot r_2 \neq l_2 \cdot r_1) \rightarrow l_1 \neq l_2.$$

The third one records the situation of “same vertex, different edges” (very similar). We say $l_1$ is very similar to $l_2$ and note it as $l_1 \cong l_2$ if it can be deduced using the following formula.

$$(l_1 \cdot t_1 = l_2 \cdot t_1) \cap (l_1 \cdot t_2 = l_2 \cdot t_2) \cap (l_1 \cdot r_1 = l_2 \cdot r_1) \cap (l_1 \cdot r_2 = l_2 \cdot r_2) \cup (l_1 \cdot t_1 = l_2 \cdot t_2) \cap (l_1 \cdot r_2 = l_2 \cdot r_2) \rightarrow l_1 \cong l_2.$$

The last relation is to describe the situation of “connected with one vertex” (connected), which we indicated as $l_1 \sim l_2$. The formula is illustrated as: $-(l_1 = l_2) \cap -(l_1 \neq l_2) \cap ((l_1 \cdot t_1 = l_2 \cdot t_1) \cup (l_1 \cdot t_1 = l_2 \cdot t_2) \cup (l_1 \cdot t_2 = l_2 \cdot t_1) \cup (l_1 \cdot t_2 = l_2 \cdot t_2)) \rightarrow l_1 \sim l_2$

The similarity score of comparing $G_1$ and $G_2$ is calculated using the following equation.

$$S_{G_1-G_2} = w_1 \times \frac{m_{G_1 \cdot l=G_2 \cdot l}}{n_{G_1} + n_{G_2} - m_{G_1 \cdot l=G_2 \cdot l}} + w_2 \times \frac{m_{G_1 \cdot l \not= G_2 \cdot l}}{n_{G_1} + n_{G_2} - m_{G_1 \cdot l \not= G_2 \cdot l}} + w_3 \times \frac{m_{G_1 \cdot l \sim G_2 \cdot l}}{n_{G_1}} + w_4 \times \frac{m_{G_1 \cdot l \sim G_2 \cdot l}}{n_{G_2}}$$

The parameters $w_1, w_2, w_3$ and $w_4$ are the Real number weights ($w_1, w_2, w_3, w_4 \in \mathbb{R}$), where $0 \leq w_1, w_2, w_3, w_4 \leq 1$, and $w_1 + w_2 + w_3 + w_4 = 1$. $n_{G_1}$ is the total number of the lexons in $G_1$; $n_{G_2}$ is the total number of the lexons in $G_2$.

The value $m_{G_1 \cdot l \not= G_2 \cdot l}$ is the size of a subset of $G_1$. We denote this subset as $G_{G_1 \cdot l \not= G_2 \cdot l}$, which is defined as $\{ G_{G_1 \cdot l \not= G_2 \cdot l} \cdot l | G_{G_1 \cdot l \not= G_2 \cdot l} \cdot l_i = G_2 \cdot l_j, 1 \leq j \leq n_{G_2}, G_{G_1 \cdot l \not= G_2 \cdot l} \subseteq G_1 \}$. The value $m_{G_1 \cdot l \sim G_2 \cdot l}$ is the size of a subset of $G_1$, which we denote as $G_{G_1 \cdot l \sim G_2 \cdot l}$. It is defined as $\{ G_{G_1 \cdot l \sim G_2 \cdot l} \cdot l | G_{G_1 \cdot l \sim G_2 \cdot l} \subseteq G_1 \}$. The value $m_{G_1 \cdot l \cdot G_2 \cdot l}$ is the size of a subset of $G_1$, which we denote as $G_{G_1 \cdot l \cdot G_2 \cdot l}$, and defined as $\{ G_{G_1 \cdot l \cdot G_2 \cdot l} \cdot l \cdot G_2 \cdot l_j | G_1 \cdot l \cdot G_2 \cdot l_j, 1 \leq j \leq n_{G_2}, G_{G_1 \cdot l \cdot G_2 \cdot l} \subseteq G_1 \}$. 
We use a semantic decision table (SDT, [40]) to monitor the values of $w_1, w_2, w_3$ and $w_4$. The semantic decision table is an extension to decision tables, which is a (set of) decision tables(s) properly annotated with domain ontologies.

An SDT contains a decision table, a set of SDT lexons and SDT commitments. A decision table consists of a set of conditions, actions and decision rules. A condition is a combination of a condition stub and condition entry. An action is a combination of an action stub and an action entry. A decision rule is a decision column in the table.

An example is illustrated in Table 2. “Profile” is a condition stub. “Optimistic” is a condition entry. The pair $\langle \text{Profile}, \text{Optimistic} \rangle$ is a condition. “$w_1$” is an action stub. “1” is an action entry. The pair $\langle w_1, 1 \rangle$ is an action. Columns 1~4 in Table 2 are the four decision rules.

<table>
<thead>
<tr>
<th>Condition</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile</td>
<td>Optimistic</td>
<td>Pessimistic</td>
<td>Balanced</td>
<td>Customized</td>
</tr>
<tr>
<td>Action</td>
<td>$w_1$</td>
<td>1</td>
<td>0.1</td>
<td>0.25</td>
</tr>
<tr>
<td></td>
<td>$w_2$</td>
<td>0</td>
<td>0.1</td>
<td>0.25</td>
</tr>
<tr>
<td></td>
<td>$w_3$</td>
<td>0</td>
<td>0.4</td>
<td>0.25</td>
</tr>
<tr>
<td></td>
<td>$w_4$</td>
<td>0</td>
<td>0.4</td>
<td>0.25</td>
</tr>
</tbody>
</table>

SDT Commitments in DECOL
1. $\langle \text{P1} = \text{[Weight, has, is of, Value]}, \text{P2} = \text{[Weight, has value type, is value type of, Float]} \rangle$: $\text{P1 (Value)} >= 0$, $\text{P1 (Value)} <= 1$.
2. $\langle \text{P1} = \text{[Weight, has, is of, Value]} \rangle$: $\text{P1 (Weight)} = \{w_1, w_2, w_3, w_4\}$, $w_1 + w_2 + w_3 + w_4 = 1$.

The SDT lexons in Table 2 are the following ones.

$\langle y, \text{Weight, has, is of, Value} \rangle$

$\langle y, \text{Weight, has value type, is value type of, Float} \rangle$

The first SDT commitment in Table 2 indicates that the weights in LexMA must be Float values, and their value range is $[0, 1]$. The second one expresses that the total value of the LexMA weights must be 1.

The SDT lexons in Table 2 are the following ones.

$\langle y, \text{Weight, has, is of, Value} \rangle$

$\langle y, \text{Weight, has value type, is value type of, Float} \rangle$

The first SDT commitment in Table 2 indicates that the weights in LexMA must be Float values, and their value range is $[0, 1]$. The second one expresses that the total value of the LexMA weights must be 1.

![Figure 4 – example of G1 and G2](image)

Suppose we have two sub-graphs as illustrated in Figure 4. We can get the following numbers:
• $n_{G_1} = 4$ (the number of lexons in $G_1$) and $n_{G_2} = 6$ (the number of lexons in $G_2$)

• $m_{G_1 \rightarrow G_2} = 1$ (The equivalent lexon from both $G_1$ and $G_2$ is $\langle y, \text{Person, manage, is managed by, Emotion} \rangle$)

• $m_{G_2 \rightarrow G_1} = 2$ (the lexons $\langle y, \text{Person, interact with, interact with, Person} \rangle$ and $\langle y, \text{Person, trust, is trusted by, Person} \rangle$ from $G_1$ share the same vertex with the lexon $\langle y, \text{Person, subordinate, is subordinated by, Person} \rangle$ from $G_2$. Note that the lexon $\langle y, \text{Person, explain, is explained by, Emotion} \rangle$ from $G_2$ is not compared to any lexons from $G_1$ because the lexon $\langle y, \text{Person, manage, is managed by, Emotion} \rangle$ has the same vertex and it is equivalent to one lexon from $G_1$)

• $m_{G_1 \rightarrow G_2} = 1$ (the number of connected lexons from $G_1$ to $G_2$) and $m_{G_2 \rightarrow G_1} = 1$ (the number of connected lexons from $G_2$ to $G_1$)

• If we take the weights defined by a “balanced” profile in Table 2, then we get the similarity score $S_{G_1 \rightarrow G_2} = 0.25 \times \frac{1}{4+6-1} + 0.25 \times \frac{2}{4+6-2} + 0.25 \times \frac{1}{4} + 0.25 \times \frac{1}{6} \approx 0.2014$
B. Ontology Graph Measuring Strategy (OntoGraM)

The OntoGraM strategy is a composition of the WordNet lexical matching algorithms\(^7\) (e.g. finding synonyms) and a simple graph algorithm that uses an ontological model to perform the matching. For instance we have an ontology model as illustrated in Figure 5. In the BT use case, a company value (such as “Trustworthy”) is an object of “Qualification”, which contains a set of competencies. Each competency contains a set of competencies. A learning object also contains a set of competencies. In such a case, \(G_1\) and \(G_2\) are two trees (specific graphs, e.g. as shown in Figure 6). The relation between a node and its parent node is “part-of”.

\[
\begin{align*}
\text{Person} & \quad -\text{has} \quad 1 \quad -\text{is of} \\
\text{Qualification} & \quad -\text{has} \quad \ast \quad -\text{is of} \\
\text{Learning Material} & \quad -\text{has} \quad 1 \quad -\text{is of} \\
\end{align*}
\]

**Figure 5** – an ontology model (in UML) used for matching

In Figure 5, each qualification is a set of competencies, which can be a task, tool, knowledge, skill or ability. When we annotate a qualification (e.g. Trustworthy), we select the competency items from a competency ontology as shown in Figure 6. “English Language” and “Customer and Personal Service” are selected as the knowledge for “Trustworthy”. “Comprehension”, “Monitoring” and “Speak” are the skills. “Vision” is a required ability. We use the same method to annotate the learning material “Course ITIL 1”. Note that the annotation tree has only two levels. The root indicates the data object. The rest are the annotated concepts from the ontology, which are categorized according to the ontology model for matching.

\[
\begin{align*}
\text{Trustworthy} & \quad \Downarrow \\
\text{English Language} & \quad \Downarrow \\
\text{Comprehension} & \quad \Downarrow \\
\text{Speaking} & \\
\text{Customer and Personal Service} & \quad \Downarrow \\
\text{Monitoring} & \quad \Downarrow \\
\text{Vision} & \\
\text{Course ITIL 1} & \\
\text{Understanding} & \quad \Downarrow \\
\text{Near Vision} & \quad \Downarrow \\
\text{Communication} & \quad \Downarrow \\
\text{Discussion} & \quad \Downarrow \\
\text{Spreadsheet software} & \\
\end{align*}
\]

**Figure 6** – two competency trees

\(^7\) http://lyle.smu.edu/~tspell/jaws/index.html (last retrieved on July 26\(^{th}\), 2011)
When we compare the two trees in Figure 6, we execute OntoGraM, which is described as follows.

Given two trees $G_1$ and $G_2$, categories $C_1, C_2, \ldots, C_n$ and annotated concepts $G_1 \cdot t_1, G_1 \cdot t_2, \ldots, G_1 \cdot t_{n_1}, G_2 \cdot t_1, G_2 \cdot t_2, \ldots, G_2 \cdot t_{n_2}$, we need to compare the annotated concepts (from $G_1$ and $G_2$) that belong to the same category $C_i$ ($i \in n$).

If $G_1 \cdot t_i = G_2 \cdot t_j$ ($i \in n_1, j \in n_2$), then we say that $G_1 \cdot t_i$ and $G_2 \cdot t_j$ exactly match (e.g. “Monitory” in Figure 6). The total number of exactly matched concepts is indicated as $n_e$.

If $G_1 \cdot t_i$ is a subtype of $G_2 \cdot t_j$ ($i \in n_1, j \in n_2$), which is defined in the domain ontology, then we say that $G_2 \cdot t_j$ subsumes $G_1 \cdot t_i$ (e.g. “Near Vision” subsumes “Vision” in Figure 6). The total number of matched concepts is indicated as $n_c$. If $G_1 \cdot t_i$ is a supertype of $G_2 \cdot t_j$, then we say that $G_1 \cdot t_i$ subsumes $G_2 \cdot t_j$ and the total number of matched concepts is indicated as $n_c$.

If $G_1 \cdot t_i$ is a synonym of $G_2 \cdot t_j$, which is defined in WordNet, then we say that $G_1 \cdot t_i$ is similar to $G_2 \cdot t_j$ (e.g. “Comprehension” and “Understanding” in Figure 6). The total number of similar concepts is indicated as $n_{Syn}$. Note that in some cases, a relation like “similar” or “similar to” is defined in the domain ontologies. In this case, we can directly use this ontological relation.

If $G_1 \cdot t_i$ and $G_2 \cdot t_j$ appear in the same lexon $\langle y, G_1 \cdot t_i, r_1, r_2, G_2 \cdot t_j \rangle$ in the domain ontology and none of the above situations occurs, then we say that $G_1 \cdot t_i$ is connected with $G_2 \cdot t_j$. For instance, “Speaking” and “Communication” are connected in the lexon $\langle y, Communication, use, is used by, Speaking \rangle$ in our domain ontology. We indicate the number of connected concepts as $n_c$.

If $G_1 \cdot t_i$ is connected with $G_2 \cdot t_j$ using the domain canonical relations that are not “similar” or “similar to” and none of the above situations occurs, then we say that $G_1 \cdot t_i$ and $G_2 \cdot t_j$ are strongly connected. For example, we have a domain canonical relation “has member”. “MS Excel” and “Spreadsheet Software” are strongly connected if we have a lexon $\langle y, MS\ Excel, is\ member\ of, has\ member, Spreadsheet\ Software \rangle$ in our domain ontology. We indicate the number of strongly connected concepts as $n_c$.

At the end, we use the following equation to calculate the matching score.

$$s_{G_1-G_2} = \frac{w_1 \times n_e + w_2 \times n_c + w_3 \times n_c + w_4 \times n_{Syn} + w_5 \times n_\sim + w_6 \times n_\sim}{n_{G_1}}$$

Where $w_1, \ldots, w_6$ are the weights ($w_1 + w_2 + \cdots + w_6 = 1, 0 \leq w_1, w_2, \ldots, w_6 \leq 1$) and $n_{G_1}$ is the number of the annotated concepts in $G_1$ (the number of vertex without the root). Similar to LexMA, we can use an SDT to configure the weights.

Suppose all the weights are $\frac{1}{6},$ we then get the matching score for Figure 6 as $s_{G_1-G_2} = \frac{1+0+1+1+1+0}{6} \times \frac{1}{6} = 0.1111$.

---

[^8]: We call this kind of relations as domain canonical relations. We define a domain canonical relation as a specific relation that must be interpreted and implemented.
C. Controlled, Fully Automated, Ontology-based Matching Strategy (C-FOAM)

C-FOAM contains two modules: 1) the Interpreter and 2) the Comparator. The interpreter module makes use of the lexical dictionary, domain dictionary, the domain ontology and string matching algorithms to interpret end users’ input. Given a term that denotes either (a) a concept in the domain ontology, or (b) an instance in the ontology, the interpreter will return the correct concept(s) defined in the ontology or lexical dictionary, and an annotation set.

A related work of the C-FOAM interpreter is named entity recognition (NER, [4]) in the fields of text minding and information retrieval. It locates and classifies atomic elements in a text into predefined categories or contexts.

There are two thresholds in the interpreter module (Figure 7). The first one is the threshold for the internal output using string matching. If C-FOAM finds the concept in the ontology, then it will go directly to the step of graph matching. Otherwise, the filtered terms will be the input for the lexical searching components. If the interpreter cannot find any concepts after executing the string matching, then it consults the lexical matching algorithm. The second threshold is to filter the output of the lexical searching components.

There is one threshold in the comparator module (Figure 7). As illustrated in the use case (Sec. IV.B), the final output of SDD-Matcher is a list of learning materials and courses. This threshold is to select the most relevant learning materials and courses after executing graph matching.

Users may select the matching algorithms of string, lexical and graph for C-FOAM. Suppose we select JaroWinklerTFIDF [20, 21, 48] as the string matching algorithm, WordNet [17] synonym finder as the lexical matching algorithm and LexMA (Sec. 0) as the graph matching algorithm to run our example. And, we set all three thresholds as 0.5.

**Situation 1:** If a user enters a string “hearty”, JaroWinklerTFIDF yields 0.97 as the matching score for “hearty” and “Heart”, which is over the threshold (0.97>0.5). The interpreter will feed “Heart” as the input to the comparator, which will find a list of learning materials and courses based on the annotation of “Heart” (see Table 1).

**Situation 2:** If the user enters “warmness”, JaroWinklerTFIDF yields 0.0 as the scores when matching “warmness” with all the company values. Then, the interpreter consults the WordNet
Synonym finder, which finds “heart”. Afterwards, C-FOAM will execute the comparator and do the rest as discussed in the previous scenario.

**Situation 3:** If the user provides “tender” as the input, which does not match with any of the company values according to JaroWinklerTFIDF or the WordNet Synonym finder, then the interpreter needs to execute the following steps: 1) load the synonyms of all the company values, e.g. the synonyms of “Heart” is “bosom”, “pump”, “ticker”, “core”, “warmness” and “tenderness” etc.; 2) compare these synonyms with the user’s input using JaroWinklerTFIDF. If the score is more than the threshold (0.5), then the corresponding company value is found; 3) C-FOAM activates the comparator. In this example, the JaroWinklerTFIDF score for “tender” and “tenderness” is 0.92 (0.92>0.5). “Tenderness” is a synonym from “heart”, which will be provided as the input for the comparator. This scenario is also applicable when the user misspells his input.

**Advanced C-FOAM:** In the above scenarios, we only use WordNet Synonym finder in the interpreter module. In that case, threshold 2 is set to 1 because a term can be either a synonym of another one or not. In the advanced C-FOAM, we use SDT (e.g. Table 3) to set the scores for different kinds of WordNet relations. In the first SDT commitment in Table 3, the value range of score is restricted as [0, 1]. In the second commitment, the score must be set 0 if the relation is antonym. The last commitment is a rule: if the relation type is “Holonym”, then the score must be the minimum nonnegative value among all the scores.

Table 3 – an SDT that decides lexical matching scores based on WordNet relations

<table>
<thead>
<tr>
<th>Condition</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relation Type</td>
<td>Antonym</td>
<td>Synonym</td>
<td>Holonym</td>
<td>Hypernym</td>
<td>Hyponym</td>
<td>Instance</td>
<td>Meronym</td>
</tr>
<tr>
<td>Action</td>
<td>P1 = [Score, has, is of, Value], P2 = [Score, has value type, is value type of, Float]: P1 (Value)&gt;=0, P1 (Value) &lt;1. IMPLIES (P2 (Relation Type) = “Antonym”, P1 (Value) =0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Score</td>
<td>0</td>
<td>0.9</td>
<td>0.1</td>
<td>0.9</td>
<td>0.5</td>
<td>0.6</td>
<td>0.2</td>
</tr>
</tbody>
</table>

If we use 0.5 as the threshold for the lexical matching, then synonyms, hypernyms, hyponyms and instances are the only types used for the lexical matching (see Table 3).

Note that the interpreter module carries penalty values for the string matching and lexical matching. Normally, the penalty for the string matching is higher than the one for the lexical matching. The final matching score of C-FOAM is calculated using the following formulas.

**Situation 1:** \( S_{G_1-G_2} = p_1 \times S_G \)

**Situation 2:** \( S_{G_1-G_2} = p_2 \times S_G \)

**Situation 3:** \( S_{G_1-G_2} = p_1 \times p_2 \times S_G \)

Where \( p_1 \) and \( p_2 \) are the penalty values for the string matching and lexical matching; \( S_G \) is the graph matching score.

Once C-FOAM gets all the matching scores between a company value and all the learning materials and course, SDD-Matcher will sort the scores and only show a list of learning materials and courses with high scores.
D. Graph-Aided Similarity calculation strategy (GRASIM)

The Graph-Aided Similarity Calculation Strategy (GRASIM) is a strategy, which adapts Dijkstra’s algorithm [12] for ontology based data matching. It converts the traveling cost of the shortest path in a graph into a similarity score. SDTs are used to freely yet correctly tune the parameters for the calculation.

As mentioned earlier, we use $G_1$ and $G_2$ to indicate two annotation sets, $G$ for the complete graph (the whole ontology, $G_1, G_2 \subseteq G$). GRASIM three steps: 1) study $G$ and label its arcs properly; 2) reorganize $G$ and use Dijkstra’s algorithm to find the shortest path $P$ between $G_1$ and $G_2$; 3) use a $P$-based function, which will be illustrated late in this section, to calculate the similarity score.

**Step 1: study the graph and label its arcs.** We use SDTs to propose weights to an end user for labeling the arcs in the ontology graph. The default weights are calculated based on the decision rules.

Once a user gets the default weights, he can check the decision rules and update the weights if he is not satisfied. Note that we use some semantics in SDTs to restrict the boundary of the weights modified by the users.

<table>
<thead>
<tr>
<th>Condition</th>
<th>1*</th>
<th>2</th>
<th>3</th>
<th>n</th>
<th>1080</th>
</tr>
</thead>
<tbody>
<tr>
<td>Int(r)</td>
<td>is-a</td>
<td>{define, describe}</td>
<td>...</td>
<td>Has char. of</td>
<td>N/A</td>
</tr>
<tr>
<td>Cons(r1,r2)</td>
<td>Uniqueness</td>
<td>Uniqueness</td>
<td>...</td>
<td>mandatory</td>
<td>N/A</td>
</tr>
<tr>
<td>Sub(t,t')</td>
<td>Yes</td>
<td>Yes</td>
<td>...</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

**SDT commitment in DECOL**

1 (P1= [Weight, has, is of, Value], P2 = [Weight, has value type, is value type of, Integer]: P1 (Value)>=0, P1 (Value) <=100.

Table 4 is an SDT example of deciding the default arc weights based on the interpretations of the role $Int(r)$, the constraints on the role pairs $Cons(r_1, r_2)$ and the constraints between the terms, e.g. $Sub(t,t')$. The conditions $\{Sub(t,t'), Yes\}$ and $\{Sub(t,t'), No\}$ indicate whether or not a concept presented by $t$ is a subtype of the other one presented by $t'$.

There are two ways to assign the weights. One is to take the average score shown in Table 4 when the weight is a value range instead of a value. For instance, the default weight for the decision rule Column 2 in Table 4 is 25 (0 < Weight < 50). The default weight for the decision rule in the column n in Table 4 is 0.

**Step 2: organize the graph and use Dijkstra’s algorithm to find shortest paths.** Dijkstra’s algorithm [12] is a well known yet simple algorithm for calculating the shortest path in a graph.

Let $G_1$ be the source graph and $G_2$ be the target graph; $G = \langle T, R \rangle$ where $T$ is a set of graph vertices (or nodes) and $R$ is a set of graph arcs. We use “·” to indicate the source graph of $T$ and $R$. A graph vertex (which is also a lexon term) is denoted as $G \cdot t$ where $G, t \in G \cdot T$. A graph arc (which corresponds to a role/co-role pair) is denoted as $G \cdot r$ where $G \cdot r \in G \cdot R$. 


The weight on $G \cdot r$ is assigned based on the decision rules in the SDT (Table 4). We need to organize the graph before using the Dijkstra’s algorithm. Its function Organize-Graph () is designed as follows.

**FUNCTION: Organize-Graph ()**

BEGIN

Get default weight $w$ in SDT

Get current lexon $\langle y, t_1, r_1, r_2, t_2 \rangle$ in current graph $G$

IF $w == 0$

  THEN merge vertices $t_1$ and $t_2$, remove arc $r_1/r_2$

ELSE IF $w == 100$

  THEN remove arc $r_1/r_2$

ELSE assign $w$ on arc $r_1/r_2$

END

The travel cost from $G_1 \cdot t_i$ to $G_2 \cdot t_j$ is a positive number given by the function of Dijkstra’s shortest path $\alpha(G_1 \cdot t_i, G_2 \cdot t_j)$. We refer to [12] for its detailed explanation. The shortest path from $G_1$ to $G_2$ is denoted as a positive number $P_{1-2}$ where $P_{1-2} \leq \alpha(G_1 \cdot t_i, G_2 \cdot t_j)$ for all $t_i \in G_1 \cdot T$ and $t_j \in G_2 \cdot T$. It means that $P_{1-2}$ is equal to at least one $\alpha(G_1 \cdot t_i, G_2 \cdot t_j)$.

Note that an arch in a graph, in our problem settings, has two directions. The lexon $\langle y, Communication, \text{ has characteristic of, is characteristic of, Clarity} \rangle$ can be illustrated as shown in Figure 8.

![Figure 8](image)

**Figure 8** – an example of transferring a lexon into a directed graph

**Step 3: Similarity Calculation.** Suppose $G_1$ has in total $n_1$ vertices and $G_2$ has $n_2$ vertices. The formula of calculating the similarity between $G_1$ and $G_2$ is designed as follows.

$$S = \lambda \times \frac{\sum_{i=1}^{n_1} \left( 1 - \frac{sp(G_1 \cdot t_i, G_2 \cdot t_j)}{sp(G_1 \cdot t_i, G_2 \cdot t_j)} \right)}{n_1} + (1 - \lambda) \times \frac{\sum_{j=1}^{n_2} \left( 1 - \frac{sp(G_2 \cdot t_j, G_1 \cdot t_i)}{sp(G_2 \cdot t_j, G_1 \cdot t_i)} \right)}{n_2}$$

Where the function $sp(x, y)$ is the shortest path from the vertex $x$ to $y$; the function $sp'(x, y)$ is also the shortest path from $x$ to $y$. The difference between them is the assigned weights on the arcs. The former takes user defined particular weights (e.g. weight = 68) and the latter is calculated based on the biggest number within the range (e.g. 70 for “50<weight<=70”).
Figure 9 - $SP'(x,y)$ is calculated with the biggest Integer within the value ranges of the weights

Figure 9 shows an example of the weights used to calculate $sp(x,y)$ and $sp'(x,y)$. The SDT in Table 4 decides the value ranges of each weight. For instance, the weight 25 is from 0<weight<50. As defined in Table 4, the weight needs to be an Integer (see commitment 1 in Table 4). Therefore, we assign 49 as the biggest number in the value range on this arc.

The parameter $\lambda$ ($0 \leq \lambda \leq 1$) is used to tune the importance of the direction. In our use case, the users tend to allocate larger numbers to the direction that goes from the input to the output. Note that we need to set $sp(x,y)$ a very small positive number (e.g. $0.1$) if $sp(x,y) = 0$.

In this section, we have discussed four matching strategies, namely LexMA, OntoGraM, C-FOAM and GRASIM in SDD-Matcher. In the next section, we will illustrate our evaluation method for evaluating different SDD-Matcher matching strategies.
VI. EVALUATION METHOD

In the previous work in the literatures, a general evaluation method for semantic-driven data matching does not exist. Evaluation methods are often trivial and application specific. Hence we need to design an evaluation method for the SDD-Matcher matching strategies.

*Program evaluation* is the systematic collection of information about the activities, characteristics and outcomes of programs to make judgments about the program, improve program effectiveness, and inform decisions about future programming [28]. It is “the systematic assessment of the operation and/or outcomes of a program or policy, compared to a set of explicit or implicit standards as a means of contributing to the improvement of program or policy” [5, 23].

*Utilization-focused evaluation* [36] is a comprehensive approach to doing evaluations that are *useful, practical, ethical* and *accurate*. Examples of such methods are the evaluation methods for non-experimental data [3], which show how to use non-experimental methods to evaluate social programs.

*Purpose oriented evaluation methodologies* [2] contain three kinds of evaluation methodologies – *formative evaluation, pretraining evaluation* and *summative evaluation*. Formative evaluation focuses on the process. Pretraining evaluation focuses on judgment of the value before the implementation. Summative evaluation focuses on the outcome.

Other evaluation types, which are considered not directly linked to our work, are product evaluation, personnel evaluation, self evaluation, advocacy evaluation, policy evaluation, organizational evaluation, and cluster evaluation etc. We refer to [19, 27, 31] for examples of product evaluation methodologies, self evaluation methodologies and cluster evaluation methodologies.

The SDD-Matcher evaluation method adapts the principles in the methodologies for *program evaluation* and *purpose oriented evaluation*.

The basic principle in the program evaluation methodologies is that the evaluation methodology needs to help a system to improve their services and/or functions, and, also help to ensure that the system is delivering the right services and/or functions. The SDD-Matcher evaluation method needs to help SDD-Matcher to improve the matching results, and, to ensure that SDD-Matcher is delivering the correct list of recommended learning materials and courses.

The principles in the purpose oriented evaluation methodologies are as follows:

- It must be able to determine what information exists in the process of a system, which is important so that the engineers can analyze the processing information;
- We need to use it to test and collect continuous feedback in order to revise the process, which is also important;
- It must have precondition analysis and post-condition analysis of the evaluated systems;
- End users must be able to judge the outcome of a system based on the evaluation methodology.

Accordingly, our SDD-Matcher evaluation method needs to determine the information during the process of SDD-Matcher, with which we analyze which matching strategy performs the best within certain contexts. It needs to continuously analyze the comparison between user’s expectations and the outcome of SDD-Matcher. The evaluation method needs to have precondition analysis and post-condition analysis of SDD-Matcher and its use case. The method needs to provide a mechanism for the end users of SDD-Matcher, with which they can justify the matching results.
The above evaluation principles give an overview of evaluating SDD-Matcher at a general level (*macro judgment*). They are the fundamental issues to judge the quality of SDD-Matcher.

We also draw evaluation criteria for the SDD-Matcher matching strategies as shown in Table 5. These evaluation criteria are used to evaluate the SDD-Matcher at a detailed level (*micro judgment*). The reasons why we choose these criteria to evaluate these strategies are explained in the column of “motivation/evaluation principle” in Table 5.

**Table 5 – Evaluation criteria for the SDD-Matcher evaluation method**

<table>
<thead>
<tr>
<th>Evaluation criteria</th>
<th>Explanation</th>
<th>Motivation/Evaluation principle(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Difficulty of managing the required knowledge resource</td>
<td>To check whether it is difficult to manage the knowledge base of a strategy or not.</td>
<td>To evaluate that this specific strategy are useful and easily used.</td>
</tr>
<tr>
<td>Difficulty of using the strategy</td>
<td>To check whether it is difficult to adjust the parameters of a strategy.</td>
<td>In order to <em>continuously</em> analyze (with different parameters) the comparison between users’ expected similarity scores and the similarity scores calculated by this strategy.</td>
</tr>
<tr>
<td>Results of the matching strategy</td>
<td>To check whether the similarity scores match users’ expectations or not.</td>
<td>To evaluate whether this strategy is delivering the right services and good functional results.</td>
</tr>
<tr>
<td>What affects the matching score</td>
<td>To find with which factors, this strategy is delivering the right services and good functional results</td>
<td></td>
</tr>
<tr>
<td>Advantage and disadvantage</td>
<td>To explain the situations that this strategy is applicable and inapplicable</td>
<td>To evaluate whether this strategy is delivering the right services and good functional results; To help a system that uses this strategy to improve their services and/or functions</td>
</tr>
<tr>
<td>Performance analysis</td>
<td>To check whether it is expensive to run a strategy</td>
<td>To evaluate whether this strategy is delivering the right services and good functional results; In Prolix, SDD-Matcher -CA is required to provide a score within 1 second</td>
</tr>
</tbody>
</table>

**Figure 10 – the SDD-Matcher evaluation method**

The SDD-Matcher evaluation method is illustrated as shown in Figure 10. The arrow-tipped bars indicate the activity flows between the SDD-Matcher evaluation steps.

**Step 1 (Design a generic use case) and Step 2 (design a detailed use case):** The steps of designing a generic use case and design a detailed use case are the preparation steps. We *scope the problem* in the step of designing a generic use case and initialize clear requirements for a viable SDD-Matcher use case. Design terms (e.g. actors9 and triggers10) are gathered and analyzed from the test beds’ materials. The output of this step is a report containing a generic SDD-Matcher use case.

---

9 An *actor* is a person or other entity external to SDD-Matcher being specified who interacts with SDD-Matcher and performs test cases or use cases to accomplish test tasks.

10 A *trigger* is an identifier of the event that initiates the use case or test case.
We specify the problem in the step of designing a detailed use case. We specify the design terms from the previous step by specifying types of information (e.g. process information) used by SDD-Matcher. We also analyze preconditions and post-conditions of the use case. The output of this step is a report containing a detailed SDD-Matcher use case.

Table 6 is an example of a detailed SDD-Matcher use case, which we have developed as a story.

**Table 6 – a story of a detailed SDD-Matcher use case**

<table>
<thead>
<tr>
<th>Purpose</th>
<th>This story describes the use case of using ontology based gap analysis framework for the recommendation of the learning materials for employees.</th>
</tr>
</thead>
</table>

**Settings**

<table>
<thead>
<tr>
<th>S1</th>
<th>An employee has competencies, which can be evaluated by reviewers and stored in an ontology.</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2</td>
<td>Learning materials contains the methods of improving skills of employees. The formats of these learning materials vary from documents to multimedia resources.</td>
</tr>
</tbody>
</table>

**Characters**

<table>
<thead>
<tr>
<th>C1</th>
<th>Every employee has a function. His function (level) gets raised when he gets a good evaluation.</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2</td>
<td>The reviewer evaluates the employee. The evaluation result, which is stored in an ontology.</td>
</tr>
<tr>
<td>C3</td>
<td>The trainer is responsible for training employees with appropriate materials.</td>
</tr>
</tbody>
</table>

**Episodes**

**Episode 1 – use case that involves graph matching algorithm**

<table>
<thead>
<tr>
<th>EI-1</th>
<th>An employee gets performance rating from a reviewer. The employee’s actual performance rating of the values is recorded in the employee’s file.</th>
</tr>
</thead>
<tbody>
<tr>
<td>EI-2</td>
<td>The employee provides input. (annotated)</td>
</tr>
<tr>
<td>EI-2.1</td>
<td>The employee provides the actual competency scores from the review.</td>
</tr>
<tr>
<td>EI-2.2</td>
<td>(optional) The employee provides the expected competency scores as the input 2.</td>
</tr>
<tr>
<td>EI-3</td>
<td>The SDD-Matcher performs the calculation.</td>
</tr>
<tr>
<td>EI-3.1</td>
<td>If input 2 is not provided, then capacities with value of NI are collected as the capacity set that needs to be improved.</td>
</tr>
<tr>
<td>EI-3.2</td>
<td>If input 2 is provided, then capacities with a value that is lower than expected are collected as the capacities that need to be improved.</td>
</tr>
<tr>
<td>EI-3.3</td>
<td>The framework collects all the capacities that do not need to be improved.</td>
</tr>
<tr>
<td>EI-3.4</td>
<td>The framework compares the gap between the capacities in the set of existing capacities and the capacities in the set of capacities that need to be improved.</td>
</tr>
<tr>
<td>EI-3.4.1</td>
<td>The framework generates two networks (two graphs) in the ontology.</td>
</tr>
<tr>
<td>EI-3.4.2</td>
<td>The framework combines the graphs of the capacities in the first set into one graph. Suppose this graph is graph 1.</td>
</tr>
<tr>
<td>EI-3.4.3</td>
<td>The framework combines the graphs of the capacities in the second set into one graph. Suppose this graph is graph 2.</td>
</tr>
<tr>
<td>EI-3.4.4</td>
<td>The framework compares graph 1 and graph 2.</td>
</tr>
<tr>
<td>EI-3.4.5</td>
<td>The framework finds the difference between graph 1 and graph 2, an internal competency gap set is generated. Suppose this gap set is gap 1.</td>
</tr>
<tr>
<td>EI-3.4.6</td>
<td>The framework finds the learning materials that are annotated with the concepts in gap 1.</td>
</tr>
<tr>
<td>EI-4</td>
<td>The matching framework generates the output.</td>
</tr>
<tr>
<td>EI-4.1</td>
<td>Output 1: a set of recommended learning materials.</td>
</tr>
<tr>
<td>EI-4.2</td>
<td>Output 2: reasons of recommendation. E.g. each learning material is illustrated with relevant concepts concerning competency in the ontology. And each capacity is also illustrated with relevant concepts concerning competency in the ontology.</td>
</tr>
<tr>
<td>EI-4.3</td>
<td>Output 3: others, e.g. the steps of graph matching (log)</td>
</tr>
</tbody>
</table>

The preconditions of the story in Table 6 are as follows: 1) there exists a competency ontology in the matching framework; 2) there exist several matching strategies in the matching framework; and 3) there is at least one good example from the test bed.

The post-conditions of the story (see Table 6) are as follows: 1) SDD-Matcher needs to illustrate the definition of a term and the relations of concepts in the ontology base; 2) SDD-Matcher needs to explain matching results from different SDD-Matcher matching strategies.

**Step 3 (Design test and evaluation data):** In this step, we design the test data that are used by SDD-Matcher (not by end users). The output of this step is a report containing a list of test and evaluation data. In Prolix, British Telecom (BT, http://www.bt.com) has been selected as our test bed. The test data set contains 26 learning materials, which are categorized into 10 soft skills. There are in total 10
company values. The ontology contains 1365 lexons, which cover 382 different concepts and 208 different role pairs. All the selected learning materials and 10 company values are properly annotated with the ontology. We refer to Table 1 as the example of annotation.

**Step 4 (Design a user test suite):** In this step, a knowledge engineer designs a user test suite, the data in which need to be provided by an evaluator. This evaluator knows well the domains of both HRM and learning/training. The output of this step is a report containing a test suite that is filled.

An example of a user test suite is illustrated in Table 7. The level of relevance can be 1, 2, 3, 4, or 5. 1 means G1 (e.g. a company value) and G2 (e.g. a learning material) are completely irrelevant. 2 means “not very relevant (or I don’t know)”. Level 3 means “relevant”. Level 4 means “very relevant” and level 5 means “100% relevant”.

<table>
<thead>
<tr>
<th>G1</th>
<th>G2</th>
<th>Level of relevance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heart</td>
<td>ITIL1</td>
<td>3</td>
</tr>
<tr>
<td>Helpful</td>
<td>ITIL8</td>
<td>2</td>
</tr>
<tr>
<td>Straightforward</td>
<td>PD0236</td>
<td>4</td>
</tr>
<tr>
<td>Inspiring</td>
<td>SKpd_04_a05</td>
<td>5</td>
</tr>
<tr>
<td>Trustworthy</td>
<td>BTAMG001</td>
<td>1</td>
</tr>
<tr>
<td>Customer connected</td>
<td>HMM23</td>
<td>2</td>
</tr>
<tr>
<td>Team work</td>
<td>FS-POSTA01</td>
<td>4</td>
</tr>
<tr>
<td>Coaching for performance</td>
<td>SKCUST0154</td>
<td>5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Table 7 needs to be filled in by an expert from the test bed. He should not have contributed to modeling the domain ontology. But he knows well the domain. We use this test suite to measure whether a particular SDD-Matcher matching strategy provides a satisfactory similarity score or not.

**Step 5 (Analyse SDD-Matcher outputs vs. users’ expectations):** The output of this step is a report of a comparison, which can be a figure or a data sheet. We will illustrate such reports in the next section.

**Step 6 (Analyse and conclude):** In the step, we analyse the comparison report that is produced in step 5 and draw valuable conclusions. We will also illustrate such conclusions in the following sections.
VII. IMPLEMENTATION AND EVALUATION RESULT

We have implemented a tool called ODMatcher (see Figure 11) to test and evaluate the SDD-Matcher matching strategies discussed in this article.

![ODMatcher](image)

**Figure 11** – a screenshot of ODMatcher

We use the criteria illustrated in Table 5 to evaluate the matching strategies.

The satisfactory rate is one criterion, which we calculate based on the data in the test suite (e.g. Table 7). How we calculate the satisfactory rates is described in the following subsection.

A. Satisfactory Rates

We first get the scale of the similarity scores of a matching strategy. For instance, the minimum and maximum scores of LexMA are 0 and 0.3225. Then the scale of LexMA similarity scores is $[0, 0.3225]$, which is equally split into 5 score ranges. There are 5 score ranges in total because there are 5 levels in the user test suite (see Table 7). Therefore, we get a mapping as follows.

- Relevance Level 5 – similarity score >0.258
- Relevance level 4 – similarity score >0.1935 and <=0.258
- Relevance level 3 – similarity score >0.129 and <=0.1935
- Relevance level 2 – similarity score >0.0645 and <=0.129
- Relevance level 1 – similarity score <=0.0645
A similarity score is “completely satisfied” if it falls in the range, otherwise, we need to calculate the bias. It is the minimum value of the low boundary bias and the high boundary bias, which are calculated using the following code.

\[
\text{IF (Similarity Score < Low Boundary)} \\
\quad \text{THEN Low Boundary Bias} = \text{Low Boundary} - \text{Similarity Score}.
\]

\[
\text{ELSE Low Boundary Bias} = \text{Similarity Score} - \text{Low Boundary}
\]

\[
\text{IF (Similarity Score < High Boundary)} \\
\quad \text{THEN High Boundary Bias} = \text{High Boundary} - \text{Similarity Score}
\]

\[
\text{ELSE High Boundary Bias} = \text{Similarity Score} - \text{High Boundary}
\]

For instance, if the similarity score for relevance level 4 is 0.2, then the low boundary bias is \(0.2 - 0.1935 = 0.0065\) and the high boundary bias is \(0.258 - 0.2 = 0.058\). The bias is then set to 0.0065.

If the bias is less than 0.0645 (one interval), then we say that this similarity score is “satisfied”. If it is more than 0.0645 and less than 0.129 (two intervals), then we say that is “not really satisfied”. All the rest scores are “completely unsatisfied”.

The satisfactory rate is the total of “completely satisfied” and “satisfied”. For instance we have 100 similarity scores. There are 51 similarity scores are the perfect match. 27 similarity scores satisfy the users’ expectation. Then the satisfactory rate is 78\% \(\left(\frac{51+27}{100} = 78\%\right)\).

Table 8 shows the result of comparing user’s expectations with the SDD-Matcher Scores generated by LexMA, OntoGraM, C-FOAM and GRASIM. In the problem settings, LexMA has the highest satisfactory rate (“completely satisfied” and “satisfied”). OntoGraM has the highest rate that is “completely satisfied”.

Table 8 – SDD-Matcher Scores vs. User’s Expectations

<table>
<thead>
<tr>
<th></th>
<th>completely satisfied</th>
<th>satisfied</th>
<th>not really satisfied</th>
<th>completely unsatisfied</th>
</tr>
</thead>
<tbody>
<tr>
<td>LexMA</td>
<td>20%</td>
<td>58%</td>
<td>20%</td>
<td>2%</td>
</tr>
<tr>
<td>OntoGraM</td>
<td>47%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-FOAM</td>
<td>27%</td>
<td>24%</td>
<td>26%</td>
<td>23%</td>
</tr>
<tr>
<td>GRASIM ((\lambda=0.5))</td>
<td>16%</td>
<td>40%</td>
<td>27%</td>
<td>17%</td>
</tr>
</tbody>
</table>

Note that we can increase the satisfactory rate of a particular matching strategy by modifying its parameters. As has been discussed in this article, SDT is used to configure the parameters of LexMA, OntoGraM and GRASIM. In what follows, we will illustrate SDT Self-Configuring Algorithm (SDT-SCA).

B. Semantic Decision Table Self-Configuring Algorithm

The processes of the semantic decision table self-configuration algorithm are illustrated as in Figure 12. Similar to the evaluation method illustrated in the previous section, in the step of preparation phase, we ask the expert to fill in a test suit (e.g., Table 7). In the steps of “run a complete test and calculate score ranges” and “calculate bias”, we get a score range and bias, the processing details of which has been as well discussed in the previous section. Then, we cluster scores with satisfactory rates (e.g. as shown in Table 8).
In the step of “Increase/decrease parameters in SDT”, we increase/decrease the parameters by assigning any possible floats that are accurate to a certain decimal places. For example, the action stubs for the changeable actions may be one value in the set \(\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}\) if we use one decimal place and allow them to vary from 0.1 to 0.9.

We repeat step 2 ~ step 5 until we cannot increase or decrease the parameters anymore. Suppose we have \(x\) changeable actions in an SDT, and we allow the action stubs to vary from \(y_1\) to \(y_2\) (with the calibration of \(c\)). A complete test of SDT-SCA contains a loop that executes exactly \((y_2 - y_1)/c + 1)^x\) times.

We collect the satisfaction rates for all the possible parameter combinations, and select the one that has the best satisfaction rate.

For instance, Table 9 shows part of the results of running SDT-SCA for LexMA. The meanings of the numbers in columns 1 ~ 9 (Table 9) are explained as follows.

- 1: parameter for one term match \((x_1)\)
- 2: parameter for two terms match \((x_2)\)
- 3: parameter for one term one role match \((x_3)\)
- 4: parameter for two terms one role match \((x_4)\)
- 5: number of the scores that are completely satisfied. We use \(n_{complet}\) to denote the value.
- 6: number of the scores that are satisfied. We use \(n_{satisfy}\) to denote the value.
- 7: number of the score that are not really satisfied. We use \(n_{notrealty}\) to denote the value.
- 8: number of the scores that are completely unsatisfied. We use \(n_{not}\) to denote the value.
9: satisfaction rate. It is calculated using the formula: \( \frac{n_{\text{complete}} + n_{\text{satisfy}}}{n_{\text{complete}} + n_{\text{satisfy}} + n_{\text{notreal}} + n_{\text{not}}} \)

Table 9 – Test results of running SDT-SCA for LexMA

<table>
<thead>
<tr>
<th>ID</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>43</td>
<td>65</td>
<td>72</td>
<td>54</td>
<td>0.462</td>
</tr>
<tr>
<td>2</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.2</td>
<td>43</td>
<td>65</td>
<td>72</td>
<td>54</td>
<td>0.462</td>
</tr>
<tr>
<td>3</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.3</td>
<td>43</td>
<td>65</td>
<td>72</td>
<td>54</td>
<td>0.462</td>
</tr>
<tr>
<td>4</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.4</td>
<td>43</td>
<td>65</td>
<td>72</td>
<td>54</td>
<td>0.462</td>
</tr>
<tr>
<td>5</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.5</td>
<td>44</td>
<td>65</td>
<td>69</td>
<td>56</td>
<td>0.464</td>
</tr>
<tr>
<td>6</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.6</td>
<td>43</td>
<td>65</td>
<td>71</td>
<td>54</td>
<td>0.463</td>
</tr>
<tr>
<td>7</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.7</td>
<td>43</td>
<td>65</td>
<td>72</td>
<td>54</td>
<td>0.461</td>
</tr>
<tr>
<td>8</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.8</td>
<td>43</td>
<td>65</td>
<td>71</td>
<td>54</td>
<td>0.463</td>
</tr>
<tr>
<td>9</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.9</td>
<td>43</td>
<td>65</td>
<td>72</td>
<td>54</td>
<td>0.462</td>
</tr>
<tr>
<td>10</td>
<td>0.1</td>
<td>0.1</td>
<td>0.2</td>
<td>0.1</td>
<td>44</td>
<td>80</td>
<td>73</td>
<td>36</td>
<td>0.533</td>
</tr>
</tbody>
</table>

In Table 9, (0.1, 0.1, 0.9, 0.1) is the best parameter set for the SDT. Accordingly, we adjust the SDT shown in Table 2 by updating the action stubs in column 4 (see Table 10).

Table 10 – Resultant SDT (tabular view only) that uses the “best” parameters according to Table 9

<table>
<thead>
<tr>
<th>Condition</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile</td>
<td>Optimistic</td>
<td>Pessimistic</td>
<td>Balanced</td>
<td>Customized</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Action</td>
<td>( w_1 )</td>
<td>1</td>
<td>0.1</td>
<td>0.25</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( w_2 )</td>
<td>0</td>
<td>0.1</td>
<td>0.25</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( w_3 )</td>
<td>0</td>
<td>0.4</td>
<td>0.25</td>
<td>0.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>( w_4 )</td>
<td>0</td>
<td>0.4</td>
<td>0.25</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

C. Other Results

Including the discussions in Section II, we would like to compare our SDD-Matcher to other approaches by considering the matching category (e.g., data mining or resource matching), applied domain (e.g. web information retrieval or web resource querying), used technologies (e.g. lexical matching algorithms or graph matching algorithms), and ontology graph type (e.g. directed graph or is-a tree). They are presented in the following table.
<table>
<thead>
<tr>
<th>Name</th>
<th>Matching Category</th>
<th>Applied Domain</th>
<th>Used Techniques</th>
<th>Ontology</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>TODE [35]</td>
<td>Data matching</td>
<td>Web information retrieval</td>
<td>Lexical Chain, SUMO, WordNet, MultiWordNet</td>
<td>Hierarchical tree</td>
<td>Is-a and Part-of, Subtype and</td>
</tr>
<tr>
<td>Raising [49]</td>
<td>Data mining</td>
<td>Market interests discovery; association rule mining</td>
<td>Exact match, Not applied</td>
<td>Raising operation method, Directed Acyclic Graph</td>
<td>Is-a only, Subtype only</td>
</tr>
<tr>
<td>Oracle RDBMS [10]</td>
<td>Semantic matching</td>
<td>Data cluster, query and data table cluster</td>
<td>Not applied, Not available</td>
<td>Based on distances and paths of concept nodes</td>
<td>Not specified, OWL relations, OWL constraints</td>
</tr>
<tr>
<td>Ontology-based matchmaker [44]</td>
<td>Resource matching</td>
<td>Data querying</td>
<td>Not applied, No</td>
<td>The matching rules are embedded in a policy ontology</td>
<td>Not specified, RDF(s) relations, User defined</td>
</tr>
<tr>
<td>DAML+OIL based Matching [24]</td>
<td>Resource matching</td>
<td>Information discovery</td>
<td>Not applied, No</td>
<td>Algorithm is written in description logic, Hierarchical tree, DAML-OIL relations</td>
<td>DAML-OIL constraints</td>
</tr>
<tr>
<td>CG for SW [49]</td>
<td>Semantic searching</td>
<td>Information discovery</td>
<td>Exact match, Not applied</td>
<td>Thematic similarity graph matching algorithm, Relations and concepts are in hierarchical trees, Ontology is in conceptual graphs, Domain relations in natural language</td>
<td>Domain relations in natural language, Subtype only</td>
</tr>
<tr>
<td>S-Match [18]</td>
<td>Schema and data matching</td>
<td>Ontology integration, matching and mapping</td>
<td>Irrelevant, WordNet, Manually</td>
<td>Hierarchical trees, Is-a only</td>
<td>Subtype only</td>
</tr>
<tr>
<td>SDD-Matcher</td>
<td>Data matching and semantic searching</td>
<td>Information discovery</td>
<td>SecondStringing algorithms, WordNet, GRASIM, LexMA and simple graph matching algorithms</td>
<td>Directed graph or hierarchical trees, Domain canonical relations and formal ontological relations</td>
<td>Any kinds of Ontological constraints</td>
</tr>
</tbody>
</table>
We have presented the design and implementation of a semantic-driven data matching framework (SDD-Matcher), which is used to find similarities between two objects using their annotations. In this paper, we have covered a generic and a detailed use cases, its comparison to the related work, its design and formalization, an evaluation method, implementation, and evaluation results.

The framework is a computational framework using semantic terms. It covers the computation of semantic tokens (e.g., strings, labels and words) and types (e.g., structural or contextual information, generic semantics from the ontology).

It is important to know that there is only one ontology in the settings. A multiple ontology scenario is also possible, which we need a preprocessing module to establish the equivalency between two ontologies. The problem can be solved by ontology alignment techniques.

SDD-Matcher contains several matching strategies, each of which is a composition of matching algorithms at the level of string, lexical and graph. Each matching strategy contains at least one graph matching algorithm. String and lexical matching algorithms are used to align different labels in a natural language into unified concepts.

Currently, the graph matching algorithms in SDD-Matcher are based on bipartite matching algorithm and Dijkstra’s shortest path algorithm. We can as well use other shortest path algorithms, such as described in [43]. Other graph matching algorithms, which are usually used but not yet implemented in SDD-Matcher, are maximum flow algorithms [32], Edmonds's non-bipartite matching algorithm [15], Tutte’s 1-factor Theorem [45] and the algorithm of finding minimum spanning tree [29]. How to use those algorithms in constructing SDD-Matcher matching strategies is one of our future works.

SDD-Matcher covers the computation of semantic tokens and types. A future direction may be twofold: levels of data pragmatics and data syntax. With regard to the level of data pragmatics, we will need to interpret and compare specific semantics in the data context, i.e. context-specific and personalized data matching. Concerning the level of data syntax, we will study data schema for structured data and natural-language structure for unstructured data. The future matching strategies can be dictionary-based, rule-based or decision table-based.

We have used semantic decision tables (SDTs) to configure matching strategies in SDD-Matcher. In order to have the most satisfied matching scores, the Semantic Decision Table Self-Configuration Algorithm (SDT-SCA) has been designed to find the “best” parameters in an SDT.

The data is taken from the EC FP6 Prolix project. It is authors’ pleasure to thank our ex-colleagues – Peter De Baer – for the implementation of ODMatcher. We shall thank Prof. Tharam Dillon from Curtin University for his valuation suggestions and ideas in the paper.
X. REFERENCES


24. Li, L. and Horrocks, I. (2004): A software framework for matchmaking based on semantic web technology, Special Issue on Semantic Web Services and Their Role in Enterprise Application Integration and E-Commerce, Michael Wellman and John Riedl, editors, Int. Journal of Electronic Commerce (IJE), Summer 2004 (8, 4)


