SDD-Matcher: a Semantic-Driven Data Matching Framework

Yan Tang\textsuperscript{a} and Gang Zhao\textsuperscript{b}
\textsuperscript{a}VUB STARLab, 10G731, Vrije Universiteit Brussel, Pleinlaan 2, 1050 Elsene, Brussels, Belgium
\textsuperscript{b}Interlartes, BP8 Waterloo, 1410 Belgium

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. Those matching algorithms are the basic building blocks of an SDD-Matcher matching strategy, each of which contains at least a graph matching algorithm. In principle, we can freely choose the matching algorithms when composing a matching strategy. In this article, we focus on a full engineering cycle of SDD-Matcher, which includes five phases – framework design, use case design, implementation, evaluation and strategy enhancement. Lexon Matching Strategy (LexMA) and Controlled, Fully Automated, Ontology-based Matching Strategy (C-FOAM) are revised and refined in SDD-Matcher. We will also illustrate a new SDD-Matcher strategy – Ontology Graph Measuring Strategy (OntoGraM) – in this paper. Semantic decision tables are used as a means to easily configure the matching strategies. In the phases of use case design, evaluation and strategy enhancement, we use real case data from a large enterprise in the fields of human resource management (HRM) and eLearning/training.

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

1. Introduction

Data matching is defined as a 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] provided 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 principle is 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 analyses. The ones shown in [13] emphasize structure analyses. The scope of this paper is essentially to deal with structure analyses.

Our problem is similar to schema matching, which is defined as a process of taking two schemas as the input and producing a mapping between elements of the two schemas that correspond semantically to each other [27] [29]. Rahm and Bernstein provided a survey of approaches to automatic schema matching in [34]. A schema matching technique, no matter whether it is automatic, semi-automatic or manual, can be adapted for ontology matching.

Ontology matching is a process of finding correspondences between semantically related entities of the ontologies [37]. An ontology is a specification of conceptualization [21]. A conceptualization is the intended models, within which a set of logical axioms are designed to account for the intended meaning of a vocabulary [19].

Although we can exploit schema matching approaches for ontology matching, we need to keep in mind that a database schema, by default, is not an ontology model. As discussed in [28], the conceptualisation and the vocabulary of a database schema are
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 \cite{7, 8, 43, 44, 45, 46}, we feel it necessary to provide an integration of SDD-Matcher, which covers a full engineering cycle containing five phases – framework design, use cases design, implementation, evaluation, and strategy enhancement. This is one of the main contributions of this article.

Other contributions are described as follows. Two main ODMF strategies, namely Lexon Matching Strategy (LexMA) and Controlled, Fully Automated, Ontology-based Matching Strategy (C-FOAM), are revised and refined in SDD-Matcher. The Ontology Graph Measuring Strategy (OntoGraM) – a new matching strategy – will be reported. We use semantic decision tables (SDT, \cite{42}) as a means to configure those strategies.

We have developed an evaluation method for justifying the SDD-Matcher matching strategies. In order to get the best matching result, we have designed and implemented an algorithm called Semantic Decision Table Self-Configuring Algorithm (SDT-SCA), which has been used in the phase of strategy enhancement.

This article is organized as follows. Sec. 2 is the related work. We will describe the engineering cycle and the frame overview of SDD-Matcher in Sec. 3. A generic use case and a detailed use case from British Telecom (BT) will be illustrated in Sec. 4. We present the SDD-Matcher matching strategies in Sec. 5. The evaluation method is recorded in Sec. 6. We show the SDD-Matcher implementation, the evaluation results and SDT-SCA in Sec. 7. In Sec. 8, we conclude with discussions and future work.

2. Related Work

One part of the related work has been already discussed in the previous section. As mentioned, the problem of semantic-driven data matching is not exactly 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

not intended a priori to be shared by other applications. An ontology, by definition, contains shared knowledge within the domain and serves as a knowledge base for multiple applications. The level of shareability is the main difference.

There are quite a lot of existing work on ontology matching and integration\cite{37}, e.g., S-Match \cite{20}, Ontima \cite{14} and AgreementMaker \cite{10}. As will be explained in the next section, there is very little related work on semantic-driven data matching (or ontology-based data matching). 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 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\cite{2}. It has been gradually enriched and tested in the EC 3DAH project\cite{1}, EC TASA project\cite{4} and the ITEA 2 DYSE project\cite{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 \cite{43, 45}. The ODMF use case in 3DAH is focused on how to evaluate medical students and provide personalized suggestions of online learning materials to them \cite{7}. In TASA, we have developed a method supported by tools to provide semantic support to process modelers during the design phase of a secure business process model. ODMF was used to discover the user design intent from a dedicated knowledge base \cite{8}. 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 \cite{46}. These use cases cover

\begin{itemize}
\item \cite{7} http://www.ontologymatching.org
\item \cite{2} http://www.prolixproject.org
\item \cite{1} http://3dah.miralab.ch
\item \cite{4} http://www.tas3.eu
\item \cite{5} http://www.dyse.org (last retrieved: July 13th, 2011)
\end{itemize}
and measurements between two data sets using an ontology.

Our problem is similar to schema matching [16]. In particular, we want to compare two sub-schemas in a large schema. This type of schemas is not a simple database schema. Instead, they are ontology models, meaning that they are shared and used by multiple applications within a certain domain.

There exist few approaches to semantic-driven data matching that make good uses of an ontology. TODE [39] uses ontologies (in particular, SUMO 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 [11] embeds ontology-based semantic matching components in their relational database management system. Oracle RDBMS provides four ontology related operators – ONTRELATED, ONTEXPAND, ONTPATH, and ONTDISTANCE. They are implemented based on the 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 [47] 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 the 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 [2], [18], [25], [30], [49] show the related work. Barrett et al. [2] illustrated an algorithm of finding the shortest path between two nodes in a labelled and weighted network. The authors in [25] tried to discover data based on feature distances. The authors in [49] focused on how to find the data objects or the 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 [25], [49]. Compared to their work, we care about neither how the boundary between two sub-graphs is drawn, 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.

Ferrara et al. [18] illustrated a survey on data linking for Semantic Web. The task is to determine whether two object descriptions refer to the same real-world object, or to discover a relationship between the two respected real-world objects. Another interesting related work that assumes there is only one ontology for two datasets to be linked is the approaches of managing open, linked data [30]. We try to find the commonality between two data sets, which correspond to two different real-world objects. Their relationship is multi-dimensional instead of one-dimensional.

3. Engineering Cycle and Framework Overview

The SDD-Matcher engineering cycle contains five phases – framework design, use case design, implementation, evaluation and strategy enhancement – as illustrated in Fig. 1.

![Fig. 1 SDD-Matcher Engineering Cycle](image)

The arrows in Fig. 1 indicate the execution flow. The initial phase is the phase of framework design. After the phase of strategy enhancement, the flow will reiterate and the framework will be updated. In the rest of this paper, we will explain the method, design and/or results for each phase.

In the phase of framework design, we have surveyed on the approaches to semantic-driven data

---

6 http://www.ontologyportal.org (last retrieved: July 18th, 2011)

7 http://linkeddata.org/
matching. As the result, a generic design of SDD-Matcher is illustrated in Fig. 2. The input is two strings. They point to two different data objects in the real world, and the output is a similarity score. 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.

As shown in Fig. 2, an SDD-Matcher strategy contains algorithms at three levels: string (or alphabetical, morphological), lexical and graph. They are the basic components for an SDD-Matcher strategy. Each strategy contains at least one and at most three algorithms. When a strategy contains one algorithm, this algorithm must be at the level of graph matching. When it contains three algorithms, each algorithm must correspond to a different level. It is not recommended to have a combination of algorithms from the same level.

As visualized in Fig. 2, the entrance points can be a matching at the string level, lexical level or graph level. Suppose we have three algorithms \(a_s, a_l, a_g\) where \(a_s\) is at the string level, \(a_l\) at the lexical level and \(a_g\) at the graph level. All the following combinations are the only possible combinations for an SDD-Matcher matching strategy.

- \(\langle a_s, a_l, a_g\rangle\)
- \(\langle a_s, a_g\rangle\)
- \(\langle a_l, a_g\rangle\)
- \(\langle a_g\rangle\)

The above combinations also show that 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 graph matching.

Note that the goal of the matching at the levels of string and lexical is to form proper sub-graphs. If an SDD-Matcher matching strategy is \(\langle a_s, a_l, a_g\rangle\), then its execution flow is described as follows. First, \(a_s\) is used to check possible typos or similar labels for the two input strings. Then, it uses \(a_l\) to find lexically connected labels. If it cannot find any, then it uses \(a_g\) again in order to find ‘similar’ lexically connected labels. Those labels, together with the ones generated in the first step, correspond to two sets of concepts in an ontology. Two sub-graphs are then formed.

In the last step, it calls \(a_g\) to measure the connectivity between these two sub-graphs and yields a similarity score. Note that each step gives a score. The ones generated by \(a_s\) and \(a_g\) need to be reworked with a penalty. Or, they need to pass a threshold.

Fig. 2 will be further studied with three matching strategies in Sec. 5.

4. A Generic Case and a BT Use Case

In the phase of use case design (see Fig. 1), a generic use case and a specific use case are created. They will be used to illustrate and test the SDD-Matcher strategies in the phase of implementation, and to evaluate these matching strategies in the phase of evaluation.

4.1. A Generic Use Case

Fig. 3 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 sets, SDD-Matcher calculates a similarity score, which is a float number in the range \([0, 1]\).
The ontology server supports not only querying or reasoning ontologies, but also ontology versioning.

Our ontology consists of two separate parts—a set of binary fact types (also called “lexons”) and a set of commitments. A lexon is \((\gamma, t_1, r_1, t_2, t_3)\) where \(t_1\) and \(t_2\) represent two concepts and identified by the context identifier \(\gamma\). \(r_1\) and \(r_2\) are the two roles that these two concepts can possibly play with; \(r_2\) is the inverse role of \(r_1\). We often use a document label or URI as the context identifier to point to a resource where \(t_1\) and \(t_2\) are originally defined, and where \(r_1\) and \(r_2\) become meaningful.

Fig. 4 shows a lexon example, which presents a fact that “a teacher teaches a student, and a student is taught by a teacher” in the context identified by “http://en.wikipedia.org/wiki/Teacher”. “Teacher” and “Student” are the two terms that point to two concepts; “teaches” and “is taught by” are the two roles.

In this example, \(r_2\) (“is taught by”) can be easily deduced from \(r_1\) (“teaches”) using a linguistic thesauri. In specific cases, it is not easy to deduce \(r_2\) from \(r_1\). That is the reason why we sometimes need to keep \(r_2\).

A commitment (or “ontological commitment”) is an agreement made by a community (also called “group of interests”). It often contains the constraint that can be expressed in some languages, such as Description Logic (DL, [1]). An example of a constraint is mandatory, e.g. each teacher teaches at least one student, which can be written in DL as Teacher \(\sqsupseteq\) teaches. Student.

A commitment can also be a query, with which we select data from an ontology-based database. For instance, we can use a SPARQL query for an RDF triple store.

Note that it does not matter which format of our domain ontology will be stored and published at the end. It can be in Web Ontology Language (OWL), RDF Schema (RDF(s)) or Semantic Web Rule Language (SWRL). Hence, we will not focus on those issues. The important consideration here is the content/models in the domain ontology.

The generic use case shown in this subsection can be extended for particular domains. In the next subsection, we will illustrate a use case from the Amsterdam branch of British Telecom (BT) in the fields of human resource management (HRM) and eLearning/training.

4.2. A BT Use Case

This use case has been worked out during the EC Prolix project.

At BT, the human resource management (HRM) department uses textual descriptions of “company values” (such as “trustworthy”, “straightforward” and “heart”) to rate its employees. The results are recorded in an assessment form. The training department uses competency notations in terms of skills and abilities to categorize learning materials. A learning material has an ID, e.g., “course ITIL 1”. Each department uses its own supporting tools and terms. In order to have a better collaboration between them, we need to enhance the interoperability for the applications across departments, we have developed an HRM ontology.

A use case scenario is as follows. Inge works for BT and gets ranked periodically. One day, she got her assessment form sent by the HRM department. She realized that her “heart” got a low rate and wanted to train herself with the learning materials that can help to improve her “heart”. She fed “heart” as an input to SDD-Matcher, which generated a sorted list of learning materials. Inge took this list and downloaded the recommended materials from the website of the training department. At the end, Inge trained herself and her “heart” got improved.

What SDD-Matcher does in this use case scenario is as follows. Before the first usage, SDD-Matcher needs to execute an annotation task. It is a pre-process task. We can use the algorithm illustrated in [41] to automate this task, or do it manually or semi-automatically. As it is out of the paper scope, we refer to [41] and its related work for further reading. Note that if a knowledge engineer cannot find a properly defined concept during the annotation task, then he needs to define this new concept by executing an ontology creation/versioning method.

In real-time, SDD-Matcher compares the annotation set of “heart” with the annotation set of each learning material. Each comparison gives a score. After all the comparison tasks are successfully exe-
culated, SDD-Matcher provides a sorted list of learning materials.

We will further enrich this use case with alternative inputs in Sec. 5.3.

5. SDD-Matcher Matching Strategies

In the phase of implementation (see Fig. 1), we need to design the SDD-Matcher matching strategies based on the use cases that are created in the phase of use case design.

The SDD-Matcher string matching algorithms include the ones from the SecondString project [9], which contains the implementation of UnsmoothedJS [23], [24], [51], JaroWinklerTFIDF [23], [24], [51] and TFIDF (term frequency–inverse document frequency, [38]).

The lexical matching algorithms are based on the lexical information from WordNet [17] and a domain dictionary. In WordNet, synonyms are grouped into unordered sets – Synsets, which are 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 hypernym of “black” and “achromatic color” is a hyponym 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, “yaw” is a troponym of “shout”.

As discussed, we use lexons to express concepts and relations. \( t_1 \) and \( t_2 \) are often two nouns; and \( r_1 \) and \( r_2 \) are often two verbs. It is rare to see adverbs in such a lexon. Therefore, the WordNet relations concerning adverbs are currently not used by SDD-Matcher. Note that the relation of antonym for nouns or verbs is not used either. We consider two terms (or roles) separated 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 the user-specific dictionary that contains data dictionaries and equivalences. It is simple from a computational point of view, yet important from a business point of view.

Graph theory has been studied as a classic research area for decades. West [50] presented a survey on its algorithms. Current SDD-Matcher takes commonly used graph matching algorithms, such as the ones of finding bipartite matching [50] and Dijkstra’s shortest path [12]. As will be discussed in the following sections, the matching algorithm in LexMA is a simplified version of bipartite matching. The GRASIM strategy, which we have studied in [43], is based on the calculation of Dijkstra’s shortest path.

In the following subsections, three matching strategies from SDD-Matcher will be illustrated.

5.1. Lexon Matching Strategy (LexMA)

LexMA is a matching strategy that only contains a graph matching algorithm. It is an enhanced version of LeMaSt [44].

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 \( G_1 \).

We identify four kinds of relations between two lexons. 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.

\[
(l_1 \cdot t_1 = l_2 \cdot t_2) \cap (l_1 \cdot r_1 = l_2 \cdot r_2) \cap (l_1 \cdot t_2 = l_2 \cdot t_1) \cap (l_1 \cdot r_2 = l_2 \cdot r_1)
\]

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.

\[
(l_1 \cdot t_1 \neq l_2 \cdot t_2) \cap (l_1 \cdot t_2 
eq l_2 \cdot t_1) \cap (l_1 \cdot r_1 \neq l_2 \cdot r_2) \cap (l_1 \cdot r_2 \neq l_2 \cdot r_1) \cap (l_1 \cdot r_1 
eq l_2 \cdot r_2)
\]

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.

\[
(l_1 \cdot t_1 = l_2 \cdot t_2) \cap (l_1 \cdot r_1 = l_2 \cdot r_2) \cap (l_1 \cdot r_1 = l_2 \cdot r_2) \cap (l_1 \cdot t_2 = l_2 \cdot t_2) \cap (l_1 \cdot r_2 = l_2 \cdot r_2)
\]

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:

\[
-(l_1 \cdot t_1 = l_2 \cdot t_1) \cap -(l_1 \cdot t_2 = l_2 \cdot t_2) \cap (l_1 \cdot t_1 = l_2 \cdot t_2) \cup (l_1 \cdot t_2 = l_2 \cdot t_1)
\]

The similarity score of comparing \( G_1 \) and \( G_2 \) is calculated using the following equation.
\[ S_{G_1-G_2} = w_1 \times \frac{m_{G_1 \leftarrow G_2}}{n_{G_1} + n_{G_2} - m_{G_1 \leftarrow G_2}} + w_2 \times \frac{m_{G_2 \leftarrow G_1}}{n_{G_1} + n_{G_2} - m_{G_2 \leftarrow G_1}} + w_3 \times \frac{m_{G_1 \leftarrow G_2} + m_{G_2 \leftarrow G_1}}{n_{G_1} + n_{G_2}} + w_4 \times \frac{m_{G_2 \leftarrow G_1}}{n_{G_2}} \]

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 ≤ \( w_1, w_2, w_3, w_4 \leq 1 \), and \( w_1 + w_2 + w_3 + w_4 = 1 \). We use \( n_{G_1} \) to indicate the total number of the lexons in \( G_1 \) and \( n_{G_2} \) to indicate the total number of the lexons in \( G_2 \).

\( m_{G_1 \leftarrow G_2} \) denotes the size of \( G_{G_1 \leftarrow G_2} \), which is defined as \( \{ l \mid G_{G_1 \leftarrow G_2} \cdot l \} \). \( G_{G_1 \leftarrow G_2} \cdot l \) denotes the size of \( G_{G_1 \leftarrow G_2} \), which is defined as \( \{ l \mid G_{G_1 \leftarrow G_2} \cdot l \} \). \( m_{G_1 \leftarrow G_2} \) indicates the size of \( G_{G_1 \leftarrow G_2} \), which is defined as \( \{ l \mid G_{G_1 \leftarrow G_2} \cdot l \} \). \( l \) is an action stub. \( 1 \) is an action entry. The pair (\( w_1, 1 \)) is an action. Columns 1-8 are the decision rules.

Table 1 decides the weights based on the profile types and the fact whether \( G_1 \) is larger than \( G_2 \) or not. We consider \( G_1 \geq G_2 \) iff \( n_1 \geq n_2 \).

An SDT commitment can be formalized in DL, First-Order Logic, or using mathematical formulas etc., depending on the required modeling feasibility and the type of the reasoning engine that will be used at the end. There are three commitments in Table 1. We use \( SOQ(\mathcal{D}) \) – a DL dialect – for axioms 1 and 2 in order to express the constraints of value range and set. Commitment 3 in Table 1 is a mathematical formula, with which we express that the total of the weights must equal to 1.

Let us use the use case in Sec. 4.2. Suppose the company value “heart” is annotated with the concepts from \( G_1 \) in Fig. 5 and a learning material called “course ITIL 1” is annotated with the ones from \( G_2 \).

![Fig. 5 Annotation graph examples; \( G_1 \) “heart”, \( G_2 \) “course ITIL 1”](image)

We get the following figures by using LexMA:

- \( n_{G_1} = 4 \)
- \( n_{G_2} = 6 \)
- \( m_{G_1 \leftarrow G_2} = 1 \) (see \( \langle \text{Person}, \text{manage}, \text{Emotion} \rangle \)\(^\text{12} \))
- \( m_{G_1 \leftarrow G_2} = 2 \) (\( \langle \text{Person}, \text{interact with}, \text{Person} \rangle \) and \( \langle \text{Person}, \text{trust}, \text{Person} \rangle \) from \( G_1 \) share the same vertex with \( \langle \text{Person}, \text{subordinate}, \text{Person} \rangle \) from \( G_2 \).
- \( m_{G_1 \leftarrow G_2} = 1 \)
- \( m_{G_1 \leftarrow G_2} = 1 \)

If we take the weights defined by a “balanced” profile (see columns 5 and 6 in Table 1), then we get the similarity score \( S_{G_1-G_2} = 0.25 \times \frac{1}{4} + 0.25 \times \frac{2}{4} + 0.25 \times \frac{2}{4} + 0.25 \times \frac{2}{4} = 0.2014 \)

\(^{12}\) We omit the context identifiers and co-roles here.
5.2. Ontology Graph Measuring Strategy (OntoGraM)

The OntoGraM strategy is a composition of the WordNet lexical matching algorithms\(^\text{13}\) (e.g., finding synonyms) and a simple graph algorithm that uses an ontological model to perform the matching.

In OntoGraM, an annotation set need to be a tree. Suppose that we have an ontology model as illustrated in Fig. 6. In the BT use case illustrated in Sec. 4.2, a company value (such as “heart”) is an object of “Qualification”, which contains a set of competencies. A competency can be a task, tool, knowledge, skill or ability. A learning material, e.g., “course ITIL 1”, also contains a set of competencies.

The rest are the concepts from the domain ontology. The relation between a node and its parent node is “part-of”.

The annotation sets that we use here are different from the ones from Sec. 5.1 (see Fig. 5). Fig. 5 contains domain canonical relations (e.g., “interact with”) while Fig. 7 only contains “part-of” merological relation. It is because LexMA deals with text-based domain canonical relations, and, OntoGraM deals with trees, which are a specific kind of graphs.

OntoGraM is described as follows. Given two trees \(G_1\) and \(G_2\), categories \(C_1,C_2,\ldots,C_n\) and annotated concepts \(C_1 \cdot t_1,C_1 \cdot t_2,\ldots,C_1 \cdot t_{n_1},G_2 \cdot t_1,G_2 \cdot t_2,\ldots,G_2 \cdot t_{n_2}\), we compare the 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., “Monitoring” in Fig. 7). The total number of the exactly matched concepts is indicated as \(n_e\).

If \(G_1 \cdot t_i\) is a subtype of \(G_2 \cdot t_j\ (i \in n_1, j \in n_2)\), then we say that \(G_2 \cdot t_j\) subsumes \(G_1 \cdot t_i\) (e.g., “Near Vision” subsumes “Vision” in Fig. 7). The total number of the matched concepts is indicated as \(n_s\).

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 the matched concepts is indicated as \(n_s\).

If \(G_1 \cdot t_i\) is a synonym of \(G_2 \cdot t_j\), then we say that \(G_1 \cdot t_i\) is similar to \(G_2 \cdot t_j\) (e.g., “Comprehension” and “Understanding” in Fig. 7). The total number of similar concepts is indicated as \(n_{\text{syn}}\). Note that in some cases, a relation like “similar” or “similar to”\(^\text{14}\) is defined in the domain ontology. 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 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 (Communication, use, Speaking). We indicate the number of the connected concepts as \(n_c\).

\[^{13}\text{http://lyle.smu.edu/~tspell/jaws/index.html}\]

\[^{14}\text{We call this kind of relations as domain canonical relations. We define a domain canonical relation as a specific relation that must be interpreted and implemented.}\]
are strongly connected if we have (MS Excel, is member of, Spreadsheet Software) in our domain ontology. We indicate the number of the strongly connected concepts as \( n_e \).

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

\[
S_{G_1 - G_2} = \left( w_1 \times n_e + w_2 \times n_+ + w_3 \times n_- + w_4 \times n_{\text{syn}} + w_5 \times n_+ + w_6 \times n_- \right) / n_{G_1}
\]

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 the 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 Fig. 7 as

\[
S_{G_1 - G_2} = \frac{1 + 0 + 1 + 1 + 1 + 0}{6} \times \frac{1}{6} = 0.1111.
\]

5.3. Controlled, Fully Automated, Ontology-based Matching Strategy (C-FOAM)

C-FOAM contains two modules: 1) an interpreter and 2) a comparator. The interpreter uses a lexical dictionary, a domain dictionary, a domain ontology and a string matching algorithm to interpret an end user’s 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. Fig. 8 shows the design of C-FOAM processes.

A related work of the C-FOAM interpreter is named entity recognition (NER, [5]) in the fields of text mining and information retrieval. It locates and classifies atomic elements in a text into predefined categories or contexts.

There are two thresholds in the interpreter. 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. As illustrated in the use case (Sec. 4.2), the final output is a list of learning materials. This threshold is to select the most relevant learning materials after executing graph matching.

Suppose we select JaroWinklerTFIDF [23, 24, 51] as the string matching algorithm, WordNet [17] synonym finder as the lexical matching algorithm and LexMA (Sec. 5.1) as the graph matching algorithm to run our example.

Situation 1: If we set the threshold to 0.5 and 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 based on the annotation of “heart”.

Situation 2: If the user enters “warmness”, JaroWinklerTFIDF yields 0.0 when comparing “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 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 “bosomi”, “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). Then, “tenderness” 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 for the interpreter. In that case, the second threshold needs to be 1 because a term can be either a synonym of another or not. In the advanced C-FOAM, we use the SDT (e.g.,
Table 2) to set the scores for different kinds of WordNet relations.

Compared to Table 1, Table 2 has a different outlook. Commitments 2 and 3 in Table 2 contain implications, which mean that if the relation type is antonym, then the score should be 0, and, if the relation type is holonym, then the score is the minimum allowed value. These two commitments are modeled in FOL and not in DL as they are not possible to be modeled in DL.

Table 2 an SDT that decides lexical matching scores based on WordNet relations

<table>
<thead>
<tr>
<th>Condition/Relation type</th>
<th>Action/Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antonym</td>
<td>0</td>
</tr>
<tr>
<td>Synonym</td>
<td>0.9</td>
</tr>
<tr>
<td>Holonym</td>
<td>0.1</td>
</tr>
<tr>
<td>Hypernym</td>
<td>0.9</td>
</tr>
<tr>
<td>Hyponym</td>
<td>0.5</td>
</tr>
<tr>
<td>Instance</td>
<td>0.6</td>
</tr>
<tr>
<td>Meronym</td>
<td>0.2</td>
</tr>
</tbody>
</table>

SDT Commitments

1. \( \text{Score} \triangleq \text{hasValue}_\text{float}[\geq 0 \leq 1] \)
2. \( \text{RelationType}(\text{Antonym}) \Rightarrow \text{hasValue}(\text{Score}, 0) \)
3. \( \text{RelationType}(\text{Holonym}) \Rightarrow \text{hasValue}(\text{Score}, \text{MIN}) \)

According to Table 2, if we use 0.5 as the threshold for the lexical matching, then synonyms, hypernms, hyponyms and instances are the only types used for the lexical matching.

Note that the interpreter 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 because we believe that lexical information is more “semantic” than textual information in many cases.

The final matching score of C-FOAM is calculated using the following formulas.

Situation 1: \( S_{g1} \rightarrow S_{g2} = p_1 \times S_g \)
Situation 2: \( S_{g1} \rightarrow S_g = p_2 \times S_g \)
Situation 3: \( S_{g1} \rightarrow S_g = p_1 \times p_2 \times S_g \)

Where \( p_1 \) and \( p_2 \) are the penalty values for the string matching and lexical matching; \( S_g \) is the graph matching score. For example, if we want to compare “hearty” with “course ITIL 2”, the annotation sets of which are illustrated in Fig. 5, and suppose that \( p_1 = 0.5 \), then we get \( S_{g1} \rightarrow S_g = 0.5 \times 0.2014 \approx 0.1 \) (see Situation 1).

Once C-FOAM gets all the matching scores between the user’s input and all the learning materials, SDD-Matcher will sort the scores and only show a list of learning materials with high scores.

6. An Evaluation Method

After the matching strategies are implemented in the phase of implementation (see Fig. 1), we go to the phase of evaluation.

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 and inform decisions about future programming [32]. 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” [6], [26].

Utilization-focused evaluation [40] 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 [4], which show how to use non-experimental methods to evaluate social programs.


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 [22], [31], [35] for the 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 program evaluation methodologies is that an evaluation methodology needs to help a system to improve their services and/or functions, and, also help to ensure that the system is delivering right services and/or functions. The SDD-Matcher evaluation method needs to help SDD-Matcher to improve the matching results, and, to en-
sure that SDD-Matcher is delivering a correct list of recommended learning materials.

The principles in 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;
- It must have a precondition analysis and a post-condition analysis;
- End users must be able to judge the outcome of a system.

Accordingly, our SDD-Matcher evaluation method needs to determine the information during processes of SDD-Matcher, with which we analyze which matching strategy performs the best within certain contexts. For instance, it must show the internal output from each matching step to our engineers. It needs to continuously analyze the comparison between user’s expectations and the outcome of an SDD-Matcher matching strategy. The method needs to provide a mechanism for end users, with which they can justify the matching results.

The above evaluation principles give an overview of evaluating an SDD-Matcher strategy at a general level. This kind of judgment is called macro judgment. They are the fundamental issues to judge the quality of a matching strategy.

We draw evaluation criteria for the SDD-Matcher matching strategies as shown in Table 3. They are used to evaluate the matching strategy at a detailed level. This kind of judgment is called micro judgment. The reason why we choose these criteria is explained in the column – “motivation/evaluation principle” – in Table 3.

Table 3 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</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>required knowledge resource</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Difficulty of using the strategy</td>
<td>To check whether it is difficult to adjust the parameters of a strategy or not.</td>
<td>In order to continuously analyze (with different parameters) the comparison between users’ expected scores and the scores calculated by this strategy.</td>
</tr>
</tbody>
</table>
from BT. The output of this step is a report containing design issues.

**Step 2 (design a use case story):** We specify the problem in this step. 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 postconditions of the use case. The output of this step is a report containing a detailed SDD-Matcher use case story, e.g., Table 4.

Table 4 a story of a detailed SDD-Matcher use case

<table>
<thead>
<tr>
<th>Title</th>
<th>SDD-Matcher for learning material recommendation</th>
<th>ID</th>
<th>OMMR_V1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scope</td>
<td>e-learning, training, material recommendation, ontology based gap analysis</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Purpose</td>
<td>This story describes a use case of using ontology based gap analysis framework for the recommendation of the learning materials for.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Settings</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S1</td>
<td>An employee has competencies, which can be evaluated by reviewers and stored in.</td>
<td></td>
<td></td>
</tr>
<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>
<td></td>
<td></td>
</tr>
<tr>
<td>C1</td>
<td>Every employee at has one function. His function (level) gets raised when he gets a good evaluation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>The reviewer evaluates the employee. The evaluation result, which is stored in, is the input of the recommendation process.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3</td>
<td>The trainer is responsible for training employees with appropriate materials.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Episodes**

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

| El | An employee gets from a reviewer. The employee’s actual performance rating of the values is recorded in. |
| El-2 | The employee provides input. (annotated) |
| El-2.1 | (mandatory) The employee provides the actual competency scores from the. |
| El-2.2 | (optional) The employee provides the expected competency scores as input 2. |
| El-3 | The SDD-Matcher performs the calculation. |
| El-3.1 | If input 2 is not provided, then capacities with value of NI are collected as the capacity set that needs to be improved. |
| El-3.2 | 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. |
| El-3.3 | The framework collects all the capacities that do not need to be improved. |
| El-3.4 | 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. |
| El-3.4.1 | The framework generates two graphs in the ontology. |
| El-3.4.2 | The framework combines the graphs of the capacities in the first set into graph 1. |
| El-3.4.3 | The framework combines the graphs of the capacities in the second set into graph 2. |

| El-3.4.4 | The framework compares graph 1 and graph 2. |
| El-3.4.5 | 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. |
| El-3.4.6 | The framework finds the learning materials that are annotated with the concepts in gap 1. |
| El-4 | The matching framework generates the output. |
| El-4.1 | Output 1: a set of recommended learning materials. |
| El-4.2 | 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. |
| El-4.3 | Output 3: others, e.g. the steps of graph matching (log) |

The preconditions of the story in Table 4 are as follows: 1) there exists a competency ontology; 2) there exist several matching strategies in SDD-Matcher; and 3) there is at least one good example from the test bed.

The post-conditions of the story (see Table 4) 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 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. For example, for the BT use case in Sec. 4.2, 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.

**Step 4 (Design a user test suite):** In this step, a knowledge engineer will design a user test suite, the data in which will be provided by an evaluator. This evaluator knows well the domains of both HRM and training but has not contributed to modelling the domain ontology. The reason why we do not allow an evaluator to be an ontology modeller is because the evaluation result will be more valuable if he is not influenced by the models in the ontology. The output of this step is a report containing a test suite.

An example of a user test suite is illustrated in Table 5. A level of relevance can be 1, 2, 3, 4, or 5; 1 means that and are completely irrelevant; 2 means “not very relevant (or I don't know)”; 3 means “relevant”; and, 4 means “very relevant” and level 5 means “100% relevant”.

| El-4.3.4.4 | The framework compares graph 1 and graph 2. |
| El-4.3.5 | 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. |
| El-4.3.6 | The framework finds the learning materials that are annotated with the concepts in gap 1. |
| El-4 | The matching framework generates the output. |
| El-4.1 | Output 1: a set of recommended learning materials. |
| El-4.2 | 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. |
| El-4.3 | Output 3: others, e.g. the steps of graph matching (log) |
We use this test suite to measure whether or not a particular SDD-Matcher matching strategy provides "good" similarity scores.

**Step 5 (Analyze 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 (Analyze and conclude)**. In the step, we analyse the comparison report that is produced in step 5 and draw valuable conclusions. We will illustrate such conclusions in the following sections.

### 7. Evaluation Result and Strategy Enhancement

Together with the criteria illustrated in Table 3, we have implemented a tool called ODMatcher as shown in Fig. 10 to test and evaluate the strategies.

<table>
<thead>
<tr>
<th>$G_1$</th>
<th>$G_2$</th>
<th>Level of relevance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heart</td>
<td>Course ITIL1</td>
<td>3</td>
</tr>
<tr>
<td>Helpful</td>
<td>Course ITIL8</td>
<td>2</td>
</tr>
<tr>
<td>Straightforward</td>
<td>PD0236</td>
<td>4</td>
</tr>
<tr>
<td>Inspiring</td>
<td>SKpl_04_a05</td>
<td>5</td>
</tr>
<tr>
<td>Trust worthy</td>
<td>RTAMG001</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>
</tbody>
</table>

Table 5 A BT test suite

7.1. Satisfactory Rates

In order to calculate the satisfactory rate of a matching strategy, we need to get the scale of the similarity scores of a matching strategy. A scale is defined as the range of the scores generated by the matching strategy. For instance, the minimum and maximum scores of LexMA are 0 and 0.3225. Then the scale of LexMA similarity scores is $[0, 0.3225]$. Seeing that there are 5 levels in Table 5, we split this scale into 5 parts.

<table>
<thead>
<tr>
<th>Level</th>
<th>Score Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0 - 0.0645</td>
</tr>
<tr>
<td>2</td>
<td>0.0645 - 0.129</td>
</tr>
<tr>
<td>3</td>
<td>0.129 - 0.1935</td>
</tr>
<tr>
<td>4</td>
<td>0.1935 - 0.258</td>
</tr>
<tr>
<td>5</td>
<td>0.258 - 0.3225</td>
</tr>
</tbody>
</table>

In order to calculate the satisfactory rate of a matching strategy, we need to get the scale of the similarity scores of a matching strategy. A scale is defined as the range of the scores generated by the matching strategy. For instance, the minimum and maximum scores of LexMA are 0 and 0.3225. Then the scale of LexMA similarity scores is $[0, 0.3225]$. Seeing that there are 5 levels in Table 5, we split this scale into 5 parts. If we split it equally, then we will get a mapping as follows.

- Relevance Level 1: $S_{G_1-G_2} > 0.258$
- Relevance level 4: $0.1935 \leq S_{G_1-G_2} \leq 0.258$
- Relevance level 3: $0.129 < S_{G_1-G_2} < 0.1935$
- Relevance level 2: $0.0645 < S_{G_1-G_2} < 0.129$
- Relevance level 1: $S_{G_1-G_2} \leq 0.0645$

A similarity score is “completely satisfied” if it falls in the range, otherwise, we need to calculate the bias. A bias is defined as a quantified difference between an actual score (generated by a matching strategy) and its expected score. It is the minimum value of the low boundary bias and the high boundary bias, which are calculated using the following algorithm.

IF (Similarity Score < Low Boundary)
   THEN Low Boundary Bias = Low Boundary - Similarity Score.
ELSE Low Boundary Bias = Similarity Score - Low Boundary

IF (Similarity Score < High Boundary)
   THEN High Boundary Bias = High Boundary - Similarity Score.
ELSE High Boundary Bias = Similarity Score - 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 0.065.

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 it 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 that perfectly match the user’s expectations, 27 similarity scores satisfy the users’ expectation. Then the satisfactory rate is $78\% \left(\frac{51+27}{100} = 78\%\right)$. 

---

[Fig. 10 The screenshot of ODMatcher]
Fig. 11 shows the result of comparing user’s expectations with the SDD-Matcher Scores generated by LexMA, OntoGraM and C-FOAM. As we can see, LexMA has the highest satisfactory rate. OntoGraM has the highest rate that is “completely satisfied”.

Note that we can increase the satisfactory rate of a matching strategy by modifying its parameters. It is a necessary step in the phase of strategy enhancement (see Fig. 1).

As has been discussed in this article, SDT is used to configure the parameters. In what follows, we will illustrate an algorithm called SDT Self-Configuring Algorithm (SDT-SCA) in order to find the best configuration.

7.2. Semantic Decision Table Self-Configuring Algorithm

The algorithm is illustrated in Fig. 12. In the step of preparation phase, we get the test suit as shown in Table 5 and decide which matching strategy we will use. 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 have been discussed in the previous section. Then, we cluster the scores with satisfactory rates.

![Fig. 12 SDT-SCA](image)

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 \(x_i\) changeable actions in an SDT, and we allow the action stubs to vary from \(y_1\) to \(y_x\) (with the calibration of \(c\)). A complete test of SDT-SCA contains a loop that executes exactly \((y_x - y_1)/c + 1)\times x\) 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 6 shows a part of the results of running SDT-SCA for LexMA.

![Table 6 Test results of running SDT-SCA for LexMA](image)

<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>0.1</td>
<td>43</td>
<td>65</td>
<td>72</td>
<td>54</td>
</tr>
<tr>
<td>2</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.2</td>
<td>0.1</td>
<td>43</td>
<td>65</td>
<td>72</td>
<td>54</td>
</tr>
<tr>
<td>3</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.3</td>
<td>0.3</td>
<td>43</td>
<td>65</td>
<td>72</td>
<td>54</td>
</tr>
<tr>
<td>4</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.4</td>
<td>0.4</td>
<td>43</td>
<td>65</td>
<td>72</td>
<td>54</td>
</tr>
<tr>
<td>5</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.5</td>
<td>0.5</td>
<td>44</td>
<td>65</td>
<td>69</td>
<td>56</td>
</tr>
<tr>
<td>6</td>
<td>0.1</td>
<td>0.1</td>
<td>0.6</td>
<td>0.6</td>
<td>0.6</td>
<td>43</td>
<td>65</td>
<td>71</td>
<td>54</td>
</tr>
<tr>
<td>7</td>
<td>0.1</td>
<td>0.1</td>
<td>0.7</td>
<td>0.7</td>
<td>0.7</td>
<td>43</td>
<td>65</td>
<td>72</td>
<td>54</td>
</tr>
<tr>
<td>8</td>
<td>0.1</td>
<td>0.1</td>
<td>0.8</td>
<td>0.8</td>
<td>0.8</td>
<td>43</td>
<td>65</td>
<td>72</td>
<td>54</td>
</tr>
<tr>
<td>9</td>
<td>0.1</td>
<td>0.1</td>
<td>0.9</td>
<td>0.9</td>
<td>0.9</td>
<td>43</td>
<td>65</td>
<td>72</td>
<td>54</td>
</tr>
<tr>
<td>10</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>44</td>
<td>80</td>
<td>73</td>
<td>36</td>
</tr>
<tr>
<td>11</td>
<td>0.1</td>
<td>0.1</td>
<td>0.8</td>
<td>0.8</td>
<td>0.8</td>
<td>65</td>
<td>109</td>
<td>53</td>
<td>6</td>
</tr>
<tr>
<td>12</td>
<td>0.1</td>
<td>0.1</td>
<td>0.8</td>
<td>0.7</td>
<td>0.7</td>
<td>66</td>
<td>110</td>
<td>52</td>
<td>6</td>
</tr>
<tr>
<td>13</td>
<td>0.1</td>
<td>0.1</td>
<td>0.8</td>
<td>0.7</td>
<td>0.7</td>
<td>66</td>
<td>111</td>
<td>51</td>
<td>6</td>
</tr>
<tr>
<td>14</td>
<td>0.1</td>
<td>0.1</td>
<td>0.8</td>
<td>0.9</td>
<td>0.9</td>
<td>66</td>
<td>112</td>
<td>51</td>
<td>6</td>
</tr>
<tr>
<td>15</td>
<td>0.1</td>
<td>0.1</td>
<td>0.9</td>
<td>0.1</td>
<td>0.1</td>
<td>67</td>
<td>119</td>
<td>43</td>
<td>5</td>
</tr>
<tr>
<td>16</td>
<td>0.1</td>
<td>0.1</td>
<td>0.9</td>
<td>0.2</td>
<td>0.2</td>
<td>67</td>
<td>120</td>
<td>43</td>
<td>5</td>
</tr>
<tr>
<td>17</td>
<td>0.1</td>
<td>0.1</td>
<td>0.9</td>
<td>0.3</td>
<td>0.3</td>
<td>67</td>
<td>119</td>
<td>43</td>
<td>5</td>
</tr>
<tr>
<td>18</td>
<td>0.1</td>
<td>0.1</td>
<td>0.9</td>
<td>0.4</td>
<td>0.4</td>
<td>67</td>
<td>119</td>
<td>43</td>
<td>5</td>
</tr>
<tr>
<td>19</td>
<td>0.1</td>
<td>0.1</td>
<td>0.9</td>
<td>0.5</td>
<td>0.5</td>
<td>66</td>
<td>112</td>
<td>51</td>
<td>6</td>
</tr>
<tr>
<td>20</td>
<td>0.1</td>
<td>0.1</td>
<td>0.9</td>
<td>0.6</td>
<td>0.6</td>
<td>66</td>
<td>111</td>
<td>51</td>
<td>6</td>
</tr>
</tbody>
</table>

The meaning of the numbers in columns 1 ~ 9 from Table 6 is explained as follows.

- **1**: parameter for one term match (suppose we indicate it as \(x_1\))
- **2**: parameter for two terms match (suppose we indicate it as \(x_2\))
- **3**: parameter for one term one role match (suppose we indicate it as \(x_3\))
- **4**: parameter for two terms one role match (suppose we indicate it as \(x_4\))
8. Conclusion, Discussion and Future Work

We have presented a generic semantic-driven data matching framework called SDD-Matcher. It is used to find similarities between two objects using their annotations. We have illustrated a full engineering cycle of SDD-Matcher, which contains five phases – framework design, uses case design, implementation, evaluation and strategy enhancement.

The framework is a computational framework using semantic tokens. It covers the computation of semantic tokens (e.g., string, label and words) and types (e.g., structural or contextual information, generic semantics from the ontology). An SDD-Matcher contains matching algorithms at the levels of string, lexical and graph. A matching strategy from SDD-Matcher is a composition of those algorithms. Each 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, and form proper sub-graphs in an ontology. Each sub-graph corresponds to an annotation set of a data object. In this article, we have shown the LexMA, enhanced C-FOAM and OntoGraM matching strategies. Together with BT, we have validated our use cases, evaluated the three matching strategies and enhanced the strategies. We have used semantic decision tables to configure matching strategies in SDD-Matcher. In the phase of strategy enhancement, in order to have the most satisfied matching scores, we have designed the SDT-SCA algorithm for automatically adjust the semantic decision tables.

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 and natural-language structure for unstructured data. The future matching strategies can be dictionary-based, rule-based or decision table-based.

SDD-Matcher uses an ontology, the format of which is not restricted. For example, we use UML diagrams and simple trees for OntoGraM, and directed graphs for LexMA and C-FOAM.

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 equivalence between those ontologies. The problem can be solved by ontology alignment techniques.

The SDT-SCA algorithm is used to enhance the matching strategies. Note that it is not the only way to enhance a strategy. We can try other means, e.g., introducing more parameters or modifying the formulas, which will be one of our future works.

We have designed a graph matching algorithm for LexMA. Other graph matching algorithms, which are usually used but not yet implemented in SDD-Matcher, are maximum flow algorithms [36], Edmonds’s non-bipartite matching algorithm [15], Tutte’s 1-factor Theorem [48] and the algorithm of finding minimum spanning tree [33]. How to use those algorithms in constructing SDD-Matcher matching strategies is one of our future works.

Another interesting future work is to apply the matching strategies from SDD-Matcher on linking data for the Semantic Web. Although the requirements are not exactly the same, some principles and works illustrated in [18] can be considered as useful approaches to applying the strategies and designing other SDD-Matcher matching strategies in the future.

9. Acknowledgement

The data are taken from the EC FP6 Prolix project. It is authors’ pleasure to thank our ex-colleagues – Peter De Baer – for the implementation of ODMatcher. With regard to the use case illustrated in this article, we are very grateful to work collaboratively with Hans Dirkzwager, Joke de Laaf and Johanneke Stam from BT. And, we shall thank Prof. Tharam Dillon from Curtin University for his valuation suggestions and ideas in the paper.