Towards Evaluating GRASIM for Ontology-based Data Matching

Yan Tang

VUB STARLab
10G731, Vrije Universiteit Brussel
Pleinlaan 2, 1050 Elesene
Brussels, Belgium

yan.tang@vub.ac.be

Abstract. The GRASIM (Graph-Aided Similarity calculation) algorithm is designed to solve the problem of ontology-based data matching. We subdivide the matching problem into the ones of restructuring a graph (or a network) and calculating the shortest path between two sub-graphs (or sub-networks). It uses Semantic Decision Tables (SDTs) for storing semantically rich configuration information of the graph. This paper presents an evaluation methodology and the evaluation results while choosing Dijkstra's algorithm to calculate the shortest paths. The tests have been executed with an actual use case of eLearning and training in British Telecom (the Amsterdam branch).

Keywords: GRASIM algorithm, ontology-based data matching, Semantic Decision Table, DOGMA, ontology

1 Introduction and Motivation

In the EC FP6 Prolix project[1], we have developed an ontology-based data matching framework called ODMF to calculate competency gaps between a learning module (such as a learning material or a course), a person's profile (such as his Curriculum Vitae), and the descriptions of a job or a task in the human resource management (HRM) domain. ODMF is a collection of several matching strategies, in which GRASIM has been designed and implemented. Although GRASIM has been designed for matching two data sets that are properly annotated with our competency ontology, it is general enough for any kinds of ontologies.

[1]The EC Prolix (FP6-IST-027905, Process-Oriented Learning and Information Exchange, http://www.prolixproject.org/) is project co-funded by the European Commission under the Sixth Framework Program. It is to align learning with business processes in order to enable organisations to faster improve the competencies of their employees according to continuous changes of business requirements.

This paper focuses on how to evaluate GRASIM and the evaluation results. It is organized as follows. Section 2 is the paper background, which includes Semantic Decision Table (SDT) and GRASIM. We present the evaluation methodology in section 3. Section 4 contains the evaluation results, discussions and lessons learnt. Section 5 is the related work. We conclude and illustrate our future work in section 6.

2 Background

Semantic Decision Table (SDT, [12], e.g., **Table 1**) is a decision table properly annotated with domain ontologies and modelled in the Developing Ontology-Grounded Methods and Applications (DOGMA [10]) framework.

An SDT contains a set of lexons, which are the simple binary fact types, e.g., the lexon in **Table 1** represents a fact that "an Arc has a Weight". It also contains a set of *commitments*, each of which is a rule in a given syntax, e.g., the SDT commitment in **Table 1** contains a *value range* constraint for "Weight".

Table 1. An SDT of deciding the tree arc weights used by GRASIM.

Condition	1*	**2**	**3**	**N**		**1080**
Int(r)	is-a	{define, describe}	…	Has char. of		N/A
Cons(r1,r2)	Uniqueness	Uniqueness	…	Mandatory		N/A
Sub(t,t')	Yes	Yes	…	Yes		No
Action						
Weight = 0				*		
0 < Weight < 50		*				
Weight = 50						
50<Weight<=70						*
70<Weight<100						
Weight = 100 (∞)	*					
SDT lexon	$\langle Arc, has, is\ of, Weight \rangle$					
SDT commitment	$0 \leq Weight \leq 100$					

GRASIM

Let us use G_1 and G_2 to indicate two annotations sets. Each set contains a list of lexons. G for the complete graph (the ontology), $G_1, G_2 \subseteq G$. We decompose the procedure into three steps: 1) study G and label its arcs; 2) reorganize G and use the Dijkstra's algorithm [4] to find the shortest path P between G_1 and G_2; 3) calculate the similarity score based on P.

On step 1, we use SDTs to propose the arc weights and label the arcs. Once a user gets the proposed weights, he could check and update them if unsatisfied. Our system uses these SDTs to check the consistency of the updated weights.

Table 1 shows an SDT that contains the decisions on the ranges of the arc weights based on the interpretations of the role $Int(r)$, the constraints on the role pair $Cons(r_1, r_2)$ and the constraints between the terms, e.g.,$Sub(t, t')$[2]. Users can assign

[2] It is a condition stub indicating whether a concept presented by t is a subtype of the other one presented by t'.

each weight with a number in the given range and label the tree arcs with these weights.

On step 2, we choose Dijkstra's algorithm to calculate the shortest path. Let G_1 be the source graph and G_2 the target graph. $G = \langle T, R \rangle$ where T is a set of graph vertices/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 $G \cdot t \in G \cdot T$. 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 the user specified weight from Step 1. If the weight is 0, then merge the two vertices. If 100, then remove the arc.

The traveral 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 [4] for its detailed explanation. The shortest path P from G_1 to G_2 is denoted as a positive number P_{1-2} where for all $t_i \in G_1 \cdot T$ and $t_j \in G_2 \cdot T$, $P_{1-2} \leq \alpha(G_1 \cdot t_i, G_2 \cdot t_j)$. That is to say, P_{1-2} is equal to the smallest output of $\alpha(G_1 \cdot t_i, G_2 \cdot t_j)$.

Note that an arc in a graph has two directions. We need to cacluate all the shortest paths from G_1 to G_2 and vice versa.

On step 3, Suppose G_1 has in total n_1 vertices and G_2 has n_2 vertices. We use the following formula to calcuate the similarity score.

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

In the above formula, the function $sp(x, y)$ is the shortest path from the vertex x to y. It uses the user defined weights. The function $sp'(x, y)$ is also the shortest path from x to y, which uses the largest Integer within the weight ranges. The parameter $\lambda, 0 \leq \lambda \leq 1$ is used to tune the importance of the tree traveling direction.

3 Evaluation Methodology

Our evaluation methodology adapts the methodological principles of program evaluation [9], purpose oriented evaluation [1], and utilization-focused evaluation [11]. The principles are listed as follows.

- **Enhancement**. It needs to help the engineers to improve the system functions.
- **Usefulness**. It needs to help to ensure that a system is delivering right functions.
- **Transparency**. It helps to determine what information in the process is important.
- **Evolution**. It needs to test and collect continuous feedbacks for revising a system.
- **Accomplishment**. It has a precondition analysis and post-condition analysis in order to determine whether all the functions of a system are well accomplished.
- **Judgment**. The end users must be able to judge the outcome of a system.

The above principles are at a high level. We call them "macro" evaluation items.

At a more detailed level, we design the evaluation criteria for a matching algorithm (e.g., GRASIM). They are the "micro" evaluation items (see below).

- **Performance analysis**. It is to check whether it is expensive to run an algorithm or not in order to evaluate the functional results. The principle of **Usefulness** is applied to this criterion. It also includes the algorithm analysis.
- **Advantages and disadvantages**. It is to explain the situations that an algorithm is applicable and not applicable, which can help to improve it in the future. The principles of **Evolution** and **Enhancement** are applied on this criterion.
- **End users' judgment**. It is to check whether the scores match the end users' expectations or not. It is used for evaluating whether it is delivering complete and good functional results or not. It is designed based on the principles of **Judgment** and **Accomplishment**. In particular, we use *satisfaction levels* as the measurement.
- **Difficulty levels of management**. It includes the following sub-items.
  Managing required knowledge base. It is to check whether it is difficult or not for end users or engineers to manage the knowledge base in order to evaluate whether it is easily used or not. This criterion is based on the principle of **Usefulness**.
  Using an algorithm. It is to check whether it is difficult or not for the engineers to manage the parameters of the algorithm. It complies with the principles of **Usefulness** and **Enhancement**.
  Improving outcomes. It is to find with which factors an algorithm is delivering good functional results in order to improve the function. It is designed based on the principles of **Transparency** and **Improvement**.

Accordingly, our evaluation methodology contains the following six steps:

- Step 1 (preparation step): we scope our test case by designing a general use case.
- Step 2 (preparation step): we specify our test case in a *story*, which contains triggers, actors, scenarios, and precondition/post-condition analysis.
- Step 3: we design the test data to feed GRASIM. In particular, we need to build the ontology and the annotation sets.
- Step 4: we design a test suit for the non-technical end users. A user test suite records the **levels of relevance** between G_1 (e.g., a company value) and G_2(e.g., a learning material). It can be 1, 2, 3, 4, or 5. 1 means that they 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".
- Step 5: we compare the results generated by the algorithm with the expectations from the end users. The outcome is a report containing a list of comparisons.
- Step 6: we analyze the above report and draw the conclusions, which will be used to enhance the algorithm for the next iteration.

We observe that the similarity scores change every time when the *ontology*, *annotation sets*, or the *weights* in the SDTs are updated. In order to correctly interpret the relevance levels before Step 5, we should not change any of the above three items.

How to interpret them is as follows. We first get the maximum score after running a complete test. Then, we equally split it into 5 ranges; e.g., if the maximum similarity score is 0.3225, then its scale is [0, 0.3225]. For the relevance levels 5,4,3,2 and 1, the ranges are: $(0.258,1)$, $(0.193,0.258]$, $(0.129,0.193]$, $(0.0645, 0.129]$ and $(0,0.0645]$.

If a score falls in the range, then we say it is "completely satisfied". Otherwise, we calculate the smaller bias. For instance, if the score for relevance level 4 is 0.2, then

bias 1 is 0.2-0.1935 = 0.0065 and bias 2 is 0.258-0.2 = 0.058. The bias is the smaller values in {0.0065, 0.058}, therefore, 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 are "completely unsatisfied".

4 Results and Lessons Learnt

Our test data is taken from BT. We need to compare a BT *assessment capacity* with a *learning material* that are annotated with the BT competency ontology. With their help, we have executed the evaluation methodology to evaluate GRASIM.

Performance Analysis

The complete test contains 26 learning materials and 10 assessment capacities. The ontology contains 1365 lexons (382 vertex and 208 arcs).

Table 2. Similarity scores of comparing the assessment capacity „Heart" to 18 learning materials. $\lambda_1 = 0, \lambda_2 = 0.1, \lambda_3 = 0.3, \lambda_4 = 0.5, \lambda_5 = 0.7, \lambda_6 = 0.9, \lambda_7 = 1$.

ID	Learning material	$S_{\lambda 1}$	$S_{\lambda 2}$	$S_{\lambda 3}$	$S_{\lambda 4}$	$S_{\lambda 5}$	$S_{\lambda 6}$	$S_{\lambda 7}$
1	Problem solving and decision making	0.78	0.77	0.75	0.72	0.7	0.68	0.66
2	Flexibility in changing circumstances	0.49	0.48	0.44	0.41	0.38	0.34	0.33
3	Communication	0.56	0.55	0.54	0.53	0.51	0.5	0.49
4	Understanding the market	0.33	0.33	0.33	0.33	0.33	0.33	0.33
5	Self management and professionalism	0.44	0.36	0.2	0.04	0.30	0.35	0.46
6	Identifying customer needs	0.46	0.38	0.22	0.05	0.21	0.28	0.33
7	Technical organization of site	0.33	0.33	0.33	0.33	0.33	0.33	0.33
8	Identifying customers' needs …	0.49	0.48	0.44	0.41	0.38	0.34	0.33
9	Problem solving and group decision…	0.51	0.49	0.46	0.42	0.38	0.34	0.32
10	Decision making: implementation …	0.62	0.63	0.63	0.64	0.65	0.66	0.66
11	Attendance management guidance…	0.54	0.53	0.53	0.53	0.53	0.52	0.52
12	Decision making (HARVARD)	0.63	0.63	0.64	0.65	0.65	0.66	0.66
13	3 day MBA	0.51	0.53	0.56	0.59	0.62	0.65	0.66
14	Cross-selling in customer serv. call	0.77	0.75	0.69	0.63	0.58	0.52	0.49
15	Valuing ability	0.53	0.53	0.52	0.51	0.51	0.5	0.49
16	Keep it simple	0.67	0.61	0.47	0.33	0.19	0.05	-
17	Bright ideas	0.53	0.47	0.36	0.25	0.14	0.03	-
18	Communications skills web seminar	0.71	0.67	0.58	0.5	0.41	0.33	0.29

Table 2 contains a part of our test data and the results. The last columns of tests 16 and 17 contain empty values, which mean that GRASIM does not return any value. It happens when no path can be found between $G_1 \cdot t_i$ and $G_2 \cdot t_j$.

We also observe that a similarity score depends on the value of λ. It does not affect the scores (e.g., tests 4 and 7 in **Table 2**) when the shortest path value

from $G_1$to G_2equals to the one from $G_2$to G_1. It happens when the arc weights assigned by the SDTs are "completely balanced" in the both directions. If the weights stored in the SDTs are "well balanced", then λ does not affect the scores a lot (e.g., tests 10, 11 and 12 in **Table 2**).

Advantages and Disadvantages

The advantages of using GRASIM are as follows:

- It is easily modified by the non-technical end users. It has been proven that a business person can deal with spreadsheets and decision tables more easily than other decision support tools, e.g., decision trees (see [12]).
- It can deal with rich ontological commitments for reasoning.
- It is easy to adjust the weights.

The disadvantage is the cost.The cost in miliseconds is 23657 miliseconds when calculating the similarity between the assessment capacity "trustworthy" and the learning material "Cross-selling in a Customer Service Call". The annotation set of this learning material contains 46 lexons and the one of "trustworthy" has 56 lexons.

End Users' Judgment

We use the means introduced in section 3 to interpret the levels of relevance from the test suite. The total numbers of the scores that are completely satisfied, satisfied, not really satisfied and completely unsatisfied are 32, 81, 55 and 33.

The satisfaction rate is **56%** (the total scores of "completely satisfied" and "satisfied"). The 'tolerable' satisfaction rate is **83%** (the ones that are not "completely unsatisfied").

The satisfaction rate is not very high because there are two BT domain experts involved in this evaluation process. Expert A provides the materials for creating the ontology and the annotations for the assessment capacities and the learning materials. Expert B provides the levels of relevance for the test suite. The advantage of having two different experts is: the satisfaction rate is more convincing because the understandings or views of expert B are not biased by the ones from expert A. The disadvantage is: the differences in the understandings may result in a not-very-good satisfaction rate.

Difficulty Levels of Management

We have developed a tool called ODMatcher to help users to modify the knowledge base, e.g., modify the annotation sets of $G_1$and G_2, set the arc weights in SDTs. It also contains a function to support the users with the matching process information, which can be considered as an assessment of end users' judgment.

Concerning the difficulty level of **managing the required knowledge base**, the end users need to know how to use domain ontologies to annotate the assessment capacities and the learning materials. The knowledge engineers need to be the ontology engineers so that they can assign meaningful weights. Therefore, the required management level is **professional**.

With regard to the difficulty level of **using GRASIM**, the knowledge engineers need to know how to construct SDTs. For instance, they need to know how to write the SDT commitments. Hence, the required level is **professional**.

Concerning the difficulty level of **improving the similarity scores**, we judge based on the difficulty levels of the following five tasks. The first one is to improve the *weights on the graph arcs stored in SDTs*, which implies that the knowledge engineers need to know the meaning of the weights.

The second one is to adjust the *value of λ in GRASIM*. As discussed, our ontology graph is a directed graph. λ is used to balance the results calculated from two directions. It does not affect a lot the final similarity score if the weights on the arcs in two directions are well balanced. The knowledge engineers need to know which value the most "suitable" one for λ.

The third one is to update the *structure of the domain ontology*. The knowledge engineers need to understand the domain and have the knowledge in ontology engineering. The scores more likely increase if more arcs (relations between the concepts) are introduced.

The forth one is to modify the *annotation sets*, which requires the expertise of domain experts. When the two annotation sets almost overlap, the similarity score is high. If they are disparate, then the similarity score depends heavily on the shortest distance between these two graphs. If the shortest paths are short, then the score is high. If they are long, then the score is low.

The last one is to improve the *expertise levels of expert A and expert B*. GRASIM will provide more accurate scores when they are improved.

Accordingly, the required level of improving the similarity scores is **professional**.

5 Related Work

A generic evaluation methodology for the problem of ontology-based data matching does not exist. The existing methodologies are trivial and often application specific. *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 [9]. 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". The evaluation methods in [3, 7] are this type of evaluation methods.

Utilization-focused evaluation [11] is an approach to executing evaluations that are practical, ethical and accurate. Examples of such methods are the evaluation methods for non-experimental data [2], which show how to use non-experimental methods to evaluate social programs.

One kind of evaluation methods is called *purpose oriented evaluation methodologies* [1], which contain three subtypes of evaluation methodologies – *formative evaluation*, *pretraining evaluation* and *summative evaluation*. They are used respectively on evaluating process, the value before the implementation, and the outcome of a method/system.

The related work, which is not directly relevant to our work but often used in our life, are product evaluation, personnel evaluation, self evaluation, advocacy evaluation, policy evaluation, organizational evaluation and cluster evaluation.

Our evaluation methodology contains the best practices of the methodologies of program evaluation, utilization-focused evaluation, and purpose-oriented evaluation.

With regard to the related work of GRASIM, we argue that GRASIM is used to solve the problem of *ontology-based data matching*, which is different from the one of *ontology matching*. Ontology matching (e.g., S-Match [5]) is to solve the inconsistency problem when merging several ontologies. In our problem setting, there exists *only one* ontology. GRASIM is to find the similarities between two data sets, each of which corresponds to one part in this ontology.

We solve this problem by transferring it into the problems of finding and calculating the semantic connections between two sub-networks in a graph. In this sense, [8, 13] are the related work of measuring this kind of semantic connections. The main focus of [13] is to find and group the web pages (considered as data objects) that belong to a same or similar context. The goal of [8] is to discover data based on distance. An approach taken by the authors from [8, 13] is to draw a boundary in the search spaces because the total world is unforeseen. This open world problem is out of the scope of GRASIM.

Compared to their solutions, GRASIM has two main innovative contributions – 1) we transfer the shortest path values between two sub-networks into a similarity score; 2) we study the semantics of the arcs and vertices in the graph using SDT.

6 Conclusion and Future Work

In this paper, we have discussed the evaluation methodology for GRASIM, which is an ontology-based data matching algorithm using SDTs. It is general enough for evaluating any kinds of ontology-based data matching algorithms. It has been tested with a real-life use case in BT and confirmed to be useful for the particular enterprise.

Currently, we are applying GRASIM to some use case scenarios in the ongoing ITEA-2 DIYSE project (http://dyse.org:8080). One use case is to search for similar software components.

Note that GRASIM is not restricted to the Dijkstra's algorithm. It can also use other graph algorithms, such as *A* algorithm* (also called *heuristic search* algorithm [6]). In the future, we will study how to use SDTs to pipeline matching tasks, propose the combinations of several shortest path algorithms, take the feedbacks from the users, and automatically adjust the SDTs.

Acknowledgments. The work has been supported by the EU ITEA-2 Project 2008005 "Do-it-Yourself Smart Experiences", founded by IWT 459. It is the author's pleasure to thank the colleagues who have been involved in the EC Prolix project, especially Ellen Leenarts (mentioned as Expert A in section 4) and Hans Dirkzwager (mentioned as Expert B in section 4) from British Telecom (the Amsterdam branch). I

shall also thank Peter De Baer and dr. Gang Zhao for their valuable feedbacks of GRASIM.

References

1. Bhola, H. S. (1990): Evaluating "Literacy for development" projects, programs and campaigns: Evaluation planning, design and implementation, and utilization of evaluation results. Hamburg, Germany: UNESCO Institute for Education; DSE [German Foundation for International Development], 306 pages
2. Blundell, R., and Costa Dias, M. (2000): *Evaluation methods for non-experimental data*, Fiscal Studies, Volume 21, Number 4, December 2000, pp. 427-468
3. CDC (2009): Developing Process Evaluation Questions, At the National Center for Chronic Disease Prevention and Health Promotion, Healthy Youth, Program Evaluation Resources, http://www.cdc.gov/healthyyouth/evaluation/resources.html
4. Dijkstra, E.W. (1959): *A note on two problems in connexion with graphs*. Numerische Mathematik 1, 269–271
5. Giunchiglia, F., Shvaiko, P., and Yatskevich, M. (2004): *S-Match: an Algorithm and an Implementation of Semantic Matching*, in book of "The Semantic Web: Research and Applications", Springer Berlin/Heidelberg, LNCS 3053/2004, ISBN 978-3-540-21999-6, pp. 61-75, September 2004
6. Hart, P.E., Nilsson, N.J., and Raphael, B. (1968): *A Formal Basis for the Heuristic Determination of Minimum Cost Paths*, IEEE Transactions on System Science and Cybernetics, SSC-4(2)
7. Johnson-Laird, P. N., Byrne, R. M. J., Schaeken, W. (1992): *Prepositional Reasoning by Model*. In: Psychological Review (ISSN: 0033-295X), volume: 99, issue: 3, start page: 418.
8. Korn, F., Sidiropoulos, N., Faloutsos, C., Siegel, E. and Protopapas, Z. (1996): *Fast Nearest Neighbor Search in Medical Databases*. In International Conference on Very Large Databases (VLDB), pages 215–226, India
9. Patton, M. Q. (2002): *Qualitative Research and Evaluation Methods*, 3rd edition, Sage Publications, Inc, London, UK, ISBN 0-7619-1971-6
10. Spyns, P., Tang, Y., and Meersman, R. (2008): *An Ontology Engineering Methodology for DOGMA*, Journal of Applied Ontology, special issue on "Ontological Foundations for Conceptual Modeling", Giancarlo Guizzardi and Terry Halpin (eds.), Volume 3, Issue 1-2, p.13-39 (2008)
11. Stufflebeam, D. L., Madaus, G. F., and Kellaghan, T. (2006): *Utilization-Focused Evaluation*, in book "Evaluation in Education and Human Services", Volume 49, Second edition, Springer, Netherlands, pp. 425~438
12. Tang, Y. (2010): *Semantic Decision Tables - A New, Promising and Practical Way of Organizing Your Business Semantics with Existing Decision Making Tools*, ISBN 978-3-8383-3791-3, LAP LAMBERT Academic Publishing AG & Co. KG, Saarbrucken, Germany
13. Venkateswaran, J., Kahveci1,T. and Camoglu, O. (2006) : *Finding Data Broadness Via Generalized Nearest Neighbors*, in proc. Of Advances in Database Technology - EDBT 2006, Springer Berlin/Heidelberg, ISSN 0302-9743, Volume 3896/2006